Soodar User Manual

Release 23.10

Soodar

Jan 21, 2025

CONTENTS:

CHAPTER

ONE

SOODAR USER GUIDE

1.1 Introduction

1.1.1 Overview

Soodar, new generation of high-capacity, enterprise, core routers, is a recent product in network's industry. Using the latest technologies and improvements in network's domain, make it a robust and reliable choice for being employed in network designs. Implementing a Cisco-wise CLI in control plane and providing a wide range of monitoring tools, ease network administrators getting familiar with product and make them more comfortable with it. The data plane, is the beating heart of Soodar. Equipping a fully-optimized software based data plane with Soodar assures high throughput on router.

Soodar can be used in vast different networks, but it is highly optimized to be used as a router in:

- MPLS core networks
- IPv4/6 core networks
- Data centers

The heart of Soodar, is its operator system. SoodarOS.

SoodarOS

SoodarOS is a routing operating system based on linux, that provides a reliable control-plane and a fast, software based data-plane with all state-of-the-art technologies.

To acheive this, SoodarOS leverages two known software suites:

- FRR for control-plane
- VPP for data-plane

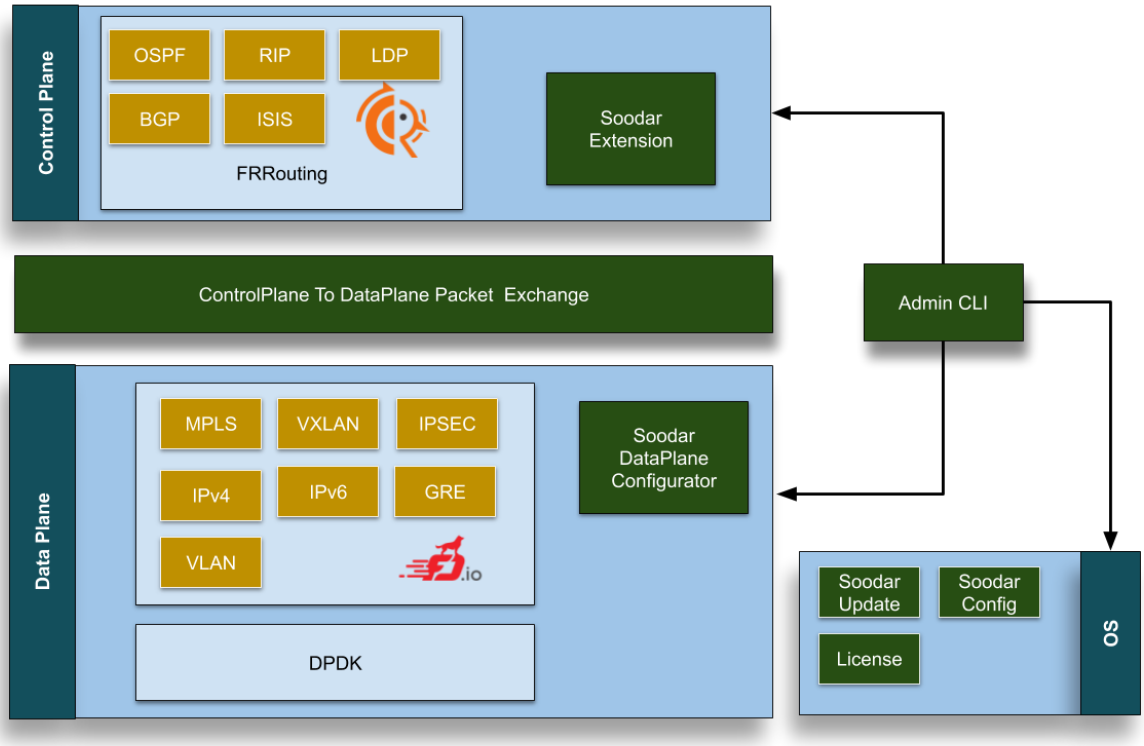
FRR is a software suite that provides TCP/IP based routing services with routing protocols support such as BGP, RIP, OSPF, IS-IS and more . FRR also supports special BGP Route Reflector and Route Server behavior. In addition to traditional IPv4 routing protocols, FRR also supports IPv6 routing protocols.

VPP is an extensible framework that provides out-of-the-box production quality switch/router functionality. It is a high performance, packet-processing stack that can run on commodity CPUs with a rich feature set.

SoodarOS uses an advanced software architecture to provide you with a high quality router. SoodarOS has an interactive user interface and supports common client commands.

Architecture

The following figure, shows SoodarOS components and their relationship



1.2 Basics

1.2.1 Modes and user's configurations

Connect to SoodarOS

There are 3 ways of connecting to router for configuring:

- **Physical connection:**
    1. Direct connection(via monitor and keyboard)
    2. Console connection(RS-232)
- **Remote connection:**
    1. SSH connection

Remote connection

Using well-known *SSH Protocol*, enabled router remote access.

Example : Having a management interface with address `192.168.1.1/24`:

```
m@m-pc:~$ ssh admin@192.168.1.1
admin@192.168.1.1's password:
```

Users

Currently, only one *admin* user is available, named *admin*. It is the username that is used with *SSH* connection.

Modes

- **View mode** Admin has access to some `show` commands to view the router's state.
- **Enable mode**: Admin can't change the router's configs. But he can enable *debug* commands and some more privileged commands than *view mode*
- **Config mode**: Full access to the router.

Passwords

SoodarOS is protected by 3 levels of passwords:

1. Access password
2. Enable password
3. Config password

Access password

It's the primary password to log in with the user. Without having the access password, a person can't have any access to the router. An admin with knowing only *access password* is an admin with just *view mode* privilege.

`password`

Change access password

Enable password

Put an admin in *enable mode*. It is asked when the admin issues the `enable` command.

`enable password PASSWORD`

Set enable password

`no enable password PASSWORD`

Disable enable password.

Config password

Is asked when the admin inputs `configure` in the command line to enter *config mode*.

`enable config password PASSWORD`

Set config password

`no enable config password PASSWORD`

Disable config password

Reset access password

In case access password is forgotten, connect to soodar via *console* and enter `user password`

`user password`

Reset access password. enabled when connected through physical access.

Password length

To force users to set strong passwords, admin can set a minimum length for passwords.

`security passwords min-length`

Apply a minimum password length policy to the system. Default of 8 characters is set as passwords' minimum length.

```
soodar(config)# security password min-length 8
```

`no security passwords min-length`

Remove all restrictions about password length.

Login Failures

Admin can ask for details of failed logins. These details are:

User name: The user who was tried to log in to(currently just admin) **Medium**: Whether it was through SSH or Console **Address**: In case of the remote login attempt, IP address of the initiator machine. Else it's *0.0.0.0*. **Date**: Attempting date

`show login failures`

Example:

```
soodar# show login failures
admin   ssh:notty       192.168.1.13    Thu Sep 17 09:18
admin   ssh:notty       192.168.1.13    Thu Sep 17 09:18
admin   ssh:notty       192.168.1.13    Thu Sep 17 09:18
```

Note: Login logs are stored only for 1 month.

Session Management

SoodarOS' admin can protect the router from DoS attacks and prevent network exhaustion by limiting the SSH authentication tries in a period and blocking the abuser's IP. Also, he can see currently established sessions and terminate them.

show users

Show current running sessions. Includes line number, session type(console or SSH), session ID, and IP address of the remote user

clear line (0-530)

Clear a TTY line and make it usable by terminating the session on that line.

Note: Clearing a line causes all sessions with the same session ID as the cleared session to terminate. In a normal situation, each line has its session ID. But if multiple sessions are run on a single SSH connection, they share the same session ID

---

login block-for TIME attempts ATTEMPT within PERIOD

Set SSH jailing parameters. If someone tries `ATTEMPT`(a number in 1 to 10 range) unsuccessful login attempts within `PERIOD`([30-600]) seconds, his IP address will be limited for next `TIME`([10-7200]) seconds. Default values are 600 seconds of jail time for 5 attempts in 30 seconds.

show login blocked-ips

Show in jail IPs.

login unblock <A.B.C.D|X:X::X:X|all>

Unblock an IP and release it from jail. Admin can unblock all blocked IPs with `all` as command input.

MOTD

Sometimes system administrator needs to set a message, so every user attempting to log in can see it. This could be done by setting a MOTD banner.

banner motd line LINE

Set motd string from an input.

no banner motd

No motd banner string will be printed.

SSH

Soodar serves as a client for the SSH and as an SSH server. Therefore, key management options are provided to users.

SSH Server

ip ssh pubkey-chain

Enter SSH server authorized keys management node.

username USER

Enter authorized public key management node for a user. Any SSH connection attempt to the user with an authorized public key is accepted.

key LINE ..

Add a public key to the user's authorized keys.

no key HASH

Remove a public key from the user's authorized keys by its hash.

no key (1-65535)

Remove a public key from the user's authorized keys by its index in the keys list.

show ip ssh pubkey-chain [verbose] [USER]

Show current authorized keys database for USER(if USER is not provided, show database of all users). if `verbose` option is activated, output complete keys instead of keys' hashes.

ip ssh port (2000-10000)

The purpose of the command is to change the default SSH listen port from 22 to a specified port number within the specified range.

- `(2000-10000)`: Specifies the port number to be used for SSH server. Valid values are integers from 2000 to 10000.

Example:

```
soodar# show ip ssh pubkey-chain
List is empty
soodar# conf ter
soodar(config)# ip ssh pubkey-chain
soodar(conf-ssh-pubkey)# username admin
soodar(conf-ssh-pubkey-user)# key ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQChX8nvRsv/
↪nmZE8r+ljuVjiwe8riTt+kmSilS44/Wr+EFWbncx/E39QugQba+0I21/wn17bHbQitMMnXjINUITzqwTnnYQ
ekwSFjBuZKWKe4i0fYoYH2cqySHiecGJHaRD40Jw/
↪6+FTDK4c0PdBIg1Vd3hF8H+bCyberpEzaJKwN2WBV4Pp2QQSU4hcIag0CB/5uk2NbO8/Ewa/
↪cVG3uPURzDWA2RRh5SI320clRyYDkmrcPv6zcZ81tFx1t6F12N0/U12n/
↪XQw+5YEL8HlbGEeQVG+p4eHuOBjP4Ta1Pz75F1Os/
↪bylGQzTGlsrH4tAz7nj011XdAVAJ4ZuQ35KIwh0sVzEKVwZ9ZRFvOH4P0ijL59f/
↪VRD878v7kVrRSKmKyZYUoJH4TBSkGEASGUXGYF+zzTI0RAa3+␣
↪4j9yFaUMJJ1j1OaMq+FshykuX+3DpBKYQ3of3KWNfLHRCGYao7Eh3QOCxUCN5DuAtYhAd/
↪vzF3DkyanO6LnnbCYkg7SFzWE= temp@test
soodar# show ip ssh pubkey-chain
admin:
   1: W7tjsK1S4C+CfMfjQSQzjiRQHPnHNMhFjbmMyOE02wU temp@test (ssh-rsa)
soodar# show ip ssh pubkey-chain verbose
admin:
   1: AAAAB3NzaC1yc2EAAAADAQABAAABgQChX8nvRsv/nmZE8r+ljuVjiwe8riTt+kmSilS44/Wr+EFWbncx/
↪E39QugQba+0I21/wn17bHbQitMMnXjINUITzqwTnnYQekwSFjBuZKWKe4i0fYoYH2cqySHiecGJHaRD4
0Jw/6+FTDK4c0PdBIg1Vd3hF8H+bCyberpEzaJKwN2WBV4Pp2QQSU4hcIag0CB/5uk2NbO8/Ewa/
↪cVG3uPURzDWA2RRh5SI320clRyYDkmrcPv6zcZ81tFx1t6F12N0/U12n/
```

(continues on next page)

(continued from previous page)

```
↪XQw+5YEL8HlbGEeQVG+p4eHuOBjP4Ta1P
z75F1Os/bylGQzTGlsrH4tAz7nj011XdAVAJ4ZuQ35KIwh0sVzEKVwZ9ZRFvOH4P0ijL59f/
↪VRD878v7kVrRSKmKyZYUoJH4TBSkGEASGUXGYF+zzTI0RAa3+4j9yFaUMJJ1j1OaMq+␣
↪FshykuX+3DpBKYQ3of3KWNfLHRC
GYao7Eh3QOCxUCN5DuAtYhAd/vzF3DkyanO6LnnbCYkg7SFzWE= temp@test (ssh-rsa)
```

SSH Client

ip ssh client

Enter SSH client known host management node.

known-host <A.B.C.D|X:X::X:X|HOST>

Add a server's public key(s) (provided by its IP or hostname) to the known hosts' list of current users.

show ip ssh client known-host <A.B.C.D|X:X::X:X|HOST>

Show public keys(if any) of a server stored in the known hosts' list.

Example:

```
soodar# show ip ssh client known-host 192.168.30.50
soodar# conf ter
soodar(config)# ip ssh client
soodar(conf-ssh-client)# known-host 192.168.30.50 ssh-rsa␣
↪AAAAB3NzaC1yc2EAAAADAQABAAABgQCwOU2O2nNJGXIN5VT1Q0j7+H9kQQ9FnE0sl9aPQbOg/
↪Sw1ryZyuUmApUFFABL7MDNZTKzWd3BfYsOB
sXOsKOHiGTZCPLbS93tvHAYlkeIcYDR9JJEi4A67nN/
↪zXSoT+Ew78iUADjWH6rQSy4dtg+ScHFAj3Z9P7TQpK8zWJDLgA28d+zyYSwNd/
↪MkF+EPmAH7mPoKkg2EGCpr889pR5mcBiXPVq69yUNFUG7U0D2aqDaGbaXk9Tcfq␣
↪CrktVmjGVF8rY91TaLMJBngVaYYsnT+xdYp8i8nicxbJoYDvvde057so␣
↪X6mcTLNXI0opUV9K5TPY7Idp6AWCAxhgJ11IN2z+HZGw56xKDVXL0VXNMngxxIC␣
↪qMV5CxhYHraGkyCha1KXnU2rPi8PbYJkJMIlsXZ+hW9oC␣
↪Zs9x6gzvHHdadi3Ox9JZ6KEqLI7OKf8KNd2alZrGUNjlDIlG/jZhWtYdB4W/
↪oFPAWa5YFqDRfu+VJdVnrGqIzr8GWRlPOjAjwOsBcQk= HOST-KEY
soodar# show ip ssh client known-host 192.168.30.50
192.168.30.50 RSA SHA256:bYisVirAvDxXqwbmYIn7IEj6Grdkf6BeTYCJ7LS11s0 HOST-KEY
soodar# ssh test@192.168.30.50
test@192.168.30.50's password:
```

1.2.2 Basic Config Commands

hostname HOSTNAME

The `hostname` command is used to assign a name to the router.

** `HOSTNAME`: the name to be assigned to the device.

The hostname can be any alphanumeric string of up to 64 characters, but it cannot contain spaces or special characters. The hostname is used in the system prompt, which appears on the command-line interface (CLI) of the device.

For example, the following command sets the hostname of a device to "RouterA":

```
soodar(config)# hostname RouterA
```

After running this command, the system prompt of the device will change to:

```
RouterA(config)#
```

Note that the hostname command only sets the name of the device in the configuration, it does not change the device's IP address or DNS name.

ip host NAME A.B.C.D

The `ip host` command is used to create an alias for a specific IP address. It allows you to assign a name to an IP address for easier configuration and management.

- `NAME`: The alias or name you want to assign to the IP address.
- `A.B.C.D`: The IP address you want to assign the name to.

For example, if you want to create an alias for the IP address 192.168.1.10 and name it "router1", you would enter the following command:

```
soodar(config)# ip host router1 192.168.1.10
```

Once you have created the alias, you can use it instead of the IP address in other commands, making it easier to manage and configure your network devices.

ip name-server A.B.C.D

The command is used to configure the IP address of a DNS (Domain Name System) server on a Cisco device. The DNS server is used to resolve domain names to their corresponding IP addresses.

- `A.B.C.D`: is the IP address of the DNS server.

Multiple DNS servers can be specified using multiple instances of the `ip name-server` command.

show clock [json]

The command displays the current date and time on the device. This command is used to verify the time settings on a device.

- `json`: specifies to display the output of the command in JSON format, which can be useful for programmatic access to the output of the command.

Example:

```
soodar# show clock
               Local time: Thu 2020-09-24 10:15:37 +0330
           Universal time: Thu 2020-09-24 06:45:37 UTC
                 RTC time: Thu 2020-09-24 06:45:37
                Time zone: Asia/Tehran (+0330, +0330)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
soodar(config)# do show clock json
{
  "timezone":"Asia/Tehran",
  "local_rtc":"no",
  "can_ntp":"yes",
  "ntp":"yes",
  "ntp_synchronized":"yes",
```

(continues on next page)

(continued from previous page)

```
  "time_usec":"Thu 2020-09-24 10:15:37 +0330",
  "rtc_time_usec":"Thu 2020-09-24 06:45:37"
}
```

clock timezone TIMEZONE

Set system timezone.

- TIMEZONE: Specifies timezone's long name based on IANA TZDatabase.

For example, to set the time zone to Paris time, the command would be:

```
soodar(config)# clock timezone Europe/Paris
```

This command affects how the device interprets and displays the time. The device's clock uses Coordinated Universal Time (UTC) as its reference point, but the timezone setting is used to determine the time offset for the local time zone.

show daemons status

Show all daemons status on startup. Indicate whether they are enabled or disabled.

service cputime-stats

Collect CPU usage statistics for individual FRR event handlers and CLI commands. This is enabled by default and can be disabled if the extra overhead causes a noticeable slowdown on your system.

Disabling these statistics will also make the *service cputime-warning (1-4294967295)* limit non-functional.

service cputime-warning (1-4294967295)

Warn if the CPU usage of an event handler or CLI command exceeds the specified limit (in milliseconds.) Such warnings are generally indicative of some routine in FRR mistakenly blocking/hogging the processing loop and should be reported as a FRR bug.

Note: The default limit is 5 seconds (i.e. 5000).

This command has no effect if *service cputime-stats* is disabled.

service walltime-warning (1-4294967295)

Warn if the total wallclock time spent handling an event or executing a CLI command exceeds the specified limit (in milliseconds.) This includes time spent waiting for I/O or other tasks executing and may produce excessive warnings if the system is overloaded. (This may still be useful to provide an immediate sign that FRR is not operating correctly due to externally caused starvation.)

Note: The default limit is 5 seconds as above.

---

debug routemap [detail]

This command turns on debugging of routemaps. When detail is specified more data is provided to the operator about the reasoning about what is going on in the routemap code.

service password-encryption

Encrypt password.

Note: Enabled by default.

`line vty`

Enter vty configuration mode.

`exec-timeout MINUTE [SECOND]`

The command is used to configure the maximum amount of time a user session can be inactive before being disconnected from the device.

- `MINUTE`: Specifies the number of minutes of inactivity before the session is terminated. The value can be from 0 to 35791.
- `SECOND` (optional): Specifies the number of seconds of inactivity before the session is terminated. The value can be from 0 to 2147483647.

For example, to set the maximum timeout for a user session to 30 minutes, you would use the following command:

```
soodar(config)# exec-timeout 30
```

If you wanted to set a timeout of 10 minutes and 30 seconds, you would use the following command:

```
soodar(config)# exec-timeout 10 30
```

Note that if you omit the `SECOND` argument, it defaults to 0. Also, if you set the timeout to 0 minutes and 0 seconds, the session will never time out due to inactivity.

Note: Default timeout vaue is 0(timeout is disabled).

`no exec-timeout`

Do not perform timeout at all. This command is as same as `exec-timeout 0 0`.

`allow-reserved-ranges`

Allow using IPv4 reserved (Class E) IP ranges for daemons. E.g.: setting IPv4 addresses for interfaces or allowing reserved ranges in BGP next-hops.

If you need multiple FRR instances (or FRR + any other daemon) running in a single router and peering via 127.0.0.0/8, it's also possible to use this knob if turned on.

Default: off.

`show diagnostic`

The command is used to show a brief overview of the system status. This overview consists of:

- Interfaces states as seen by control-plane and data-plane.
- System services status
- Data-plane errors and stats
- Subsystem crashes

1.2.3 Sample Configuration

Below is a sample configuration file .

```
n1(config)# hostname soodar
soodar(config)# enable password admin
soodar(config)# enable config password configadmin
```

Note: ! and # are comment characters. If the first character of the word is one of the comment characters then from the rest of the line forward will be ignored as a comment.

```
soodar(config)# enable password admin!password
```

If a comment character is not the first character of the word, it's a normal character. So in the above example ! will not be regarded as a comment and the password is set to `admin!password`.

Terminal Mode Commands

`write terminal`

The `write terminal` command is used to display the configuration of a device on the terminal screen. This command is often used to check the current configuration of a device, troubleshoot configuration issues, or to copy the configuration to another device.

This command is equivalent to the `show running-config` command, which displays the current configuration of the device similarly. The write terminal command can be used interchangeably with the `show running-config` command in most cases.

`write file`

Write current configuration to configuration file on storage.

`write erase [A.B.C.D/M A.B.C.D]`

Erase the startup configurations file and replace the default one or the provided one.

- `A.B.C.D/M`: Specifies the IP address to set on *ge0* interface.
- `A.B.C.D`: Specifies the default gateway address.

Note: The default configuration is setting IP address of *192.168.1.55/24* on interface *ge0*.

`configure [terminal]`

Change to configuration mode. This command is the first step to configuration.

`terminal colorize`

Enable/disable color output for terminal

`terminal length (0-4294967295)`

Set terminal display length to `(0-4294967295)`. If length is 0, no display control is performed.

`list`

List all available commands.

show version

Show the current version of SoodarOS and its host information.

```
soodar# show version
OS information
--------------
  System            : Linux
  Node Name         : soodar
  Kernel Version    : 5.4.209-intel-pk-standard
  Systemd Version   : 244.5-r4
  Processor         : x86_64
  Boot Time         : 2023/3/5 14:47:55
  Uptime            : 0:00:44.244257
  Release           : soo-23.04

Packages information
--------------------
  FRR Version       : 8.1+git0+46428baf74-r4
  VPP Version       : 22.06+git0+803ac2c2b0-r4
  StrongSwan Version: 5.9.8+git0+5b0e9486e9-r4
  Mender Version    : 2.6.1-r4
```

show command history

Show entered commands. The history is kept between sessions and is not cleared until an explicit demand of removing history

clear command history [(0-200)]

Clear history command and(if provided) keep the last N commands in history. If N is not provided or it is 0, all history is erased.

- `(0-200)`: Specifies the number of most recent commands to keep in the history buffer. If this parameter is not specified or is 0, all commands in the history buffer will be cleared.

show processes

Show current processes running on the router, their PIDs, statuses, and used memory.

Example:

```
soodar# show processes
   PID     LWP    PPID Status       Size Name
     1       1       0 S           22655 systemd
     2       2       0 S               0 kthreadd
     3       3       2 I               0 rcu_gp
     4       4       2 I               0 rcu_par_gp
     6       6       2 I               0 kworker/0:0H-kblockd
     8       8       2 I               0 mm_percpu_wq
     9       9       2 S               0 ksoftirqd/0
    10      10       2 I               0 rcu_preempt
    11      11       2 S               0 migration/0
    12      12       2 S               0 cpuhp/0
    13      13       2 S               0 cpuhp/1
    14      14       2 S               0 migration/1
    15      15       2 S               0 ksoftirqd/1
    17      17       2 I               0 kworker/1:0H-kblockd
    18      18       2 S               0 kdevtmpfs
```

(continues on next page)

(continued from previous page)

```
     19      19       2 I              0 netns
     20      20       2 S              0 rcu_tasks_kthre
     21      21       2 S              0 kauditd
     23      23       2 I              0 kworker/0:1-events
     24      24       2 S              0 oom_reaper
     25      25       2 I              0 writeback
     26      26       2 S              0 kcompactd0
     27      27       2 S              0 khugepaged
     40      40       2 I              0 cryptd
     59      59       2 I              0 kblockd
     60      60       2 I              0 blkcg_punt_bio
     61      61       2 I              0 tpm_dev_wq
     62      62       2 I              0 ata_sff
     63      63       2 I              0 md
     64      64       2 S              0 watchdogd
     65      65       2 S              0 kswapd0
     67      67       2 I              0 acpi_thermal_pm
     69      69       2 I              0 tpm-vtpm
     70      70       2 I              0 nvme-wq
     71      71       2 I              0 nvme-reset-wq
     72      72       2 I              0 nvme-delete-wq
     73      73       2 S              0 scsi_eh_0
     74      74       2 I              0 scsi_tmf_0
     75      75       2 S              0 scsi_eh_1
     76      76       2 I              0 scsi_tmf_1
     77      77       2 S              0 scsi_eh_2
     78      78       2 I              0 scsi_tmf_2
     79      79       2 S              0 scsi_eh_3
     80      80       2 I              0 scsi_tmf_3
     81      81       2 S              0 scsi_eh_4
     82      82       2 I              0 scsi_tmf_4
     83      83       2 S              0 scsi_eh_5
     84      84       2 I              0 scsi_tmf_5
     89      89       2 I              0 kworker/u4:6-events_unbound
     91      91       2 I              0 kworker/0:1H-kblockd
     92      92       2 I              0 kworker/1:2-rcu_gp
     93      93       2 I              0 raid5wq
     94      94       2 I              0 dm_bufio_cache
     95      95       2 I              0 ipv6_addrconf
     96      96       2 I              0 kworker/u5:0
    101     101       2 S              0 jbd2/sda2-8
    102     102       2 I              0 ext4-rsv-conver
    103     103       2 I              0 kworker/1:1H-events_highpri
    108     108       2 S              0 jbd2/sda4-8
    109     109       2 I              0 ext4-rsv-conver
    150     150       1 S           3299 systemd-udevd
    179     179       2 S              0 scsi_eh_6
    180     180       2 I              0 scsi_tmf_6
    181     181       2 S              0 scsi_eh_7
    182     182       2 I              0 scsi_tmf_7
    183     183       2 I              0 kworker/1:3-dm_bufio_cache
    199     199       1 S          37847 rngd
```

(continues on next page)

(continued from previous page)

```
    199     212       1 S          37847 rngd
    199     213       1 S          37847 rngd
    307     307       1 S           3163 soosys
    311     311       1 S          12305 systemd-journal
    318     318       1 S            605 atd
    320     320       1 S            730 crond
    321     321       1 S           1082 dbus-daemon
    332     332       1 S           3521 snmpd
    334     334       1 S           2793 snmptrapd
    335     335       1 S           1091 chronyd
    340     340       1 R       21304346 vpp_main
    340     359       1 S       21304346 eal-intr-thread
    348     348       1 S           1520 systemd-logind
    350     350       1 S            592 agetty
    351     351       1 S            914 login
    352     352       1 S         324366 charon-systemd
    352     369       1 S         324366 charon-systemd
    352     370       1 S         324366 charon-systemd
    352     371       1 S         324366 charon-systemd
    352     372       1 S         324366 charon-systemd
    352     373       1 S         324366 charon-systemd
    352     374       1 S         324366 charon-systemd
    352     375       1 S         324366 charon-systemd
    352     376       1 S         324366 charon-systemd
    352     377       1 S         324366 charon-systemd
    352     378       1 S         324366 charon-systemd
    352     379       1 S         324366 charon-systemd
    352     380       1 S         324366 charon-systemd
    352     381       1 S         324366 charon-systemd
    352     382       1 S         324366 charon-systemd
    352     383       1 S         324366 charon-systemd
    352     384       1 S         324366 charon-systemd
    354     354       1 S         110108 f2b/server
    354     361       1 S         110108 f2b/observer
    354     364       1 S         110108 f2b/f.sshd
    354     365       1 S         110108 f2b/a.sshd
    354    6750       1 S         110108 f2b/observer
    418     418       1 S           3963 watchfrr
    433     433       1 S         106344 zebra
    433     434       1 S         106344 RCU sweeper
    433     435       1 S         106344 zebra_dplane
    433     436       1 S         106344 zebra_opaque
    433     440       1 S         106344 zebra_apic
    438     438       1 S           2330 staticd
    458     458       2 I              0 kworker/u4:0-events_unbound
    657     657       1 S           1946 systemd
    658     658     657 S          22868 (sd-pam)
    663     663     351 R          11757 vtysh
   5763    5763       2 I              0 kworker/0:2-mm_percpu_wq
   7634    7634     433 R           1092 ps
```

show processes detailed process-id (0-1000000)

Show details of a PID.

Example:

```
soodar# show processes detailed process-id 433
zebra
  Process ID        : 433
  Parent process ID : 1
  Group ID          : 433
  Status            : S
  Session ID        : 433
  User time         : 31
  Kernel time       : 17
  Priority          : 19
  Virtual bytes     : 435585024
  Resident pages    : 4598
  Resident limit    : 18446744073709551615
  Minor page faults : 851
  Major page faults : 2
  Threads           : 5
  Allowed CPUs      : 0-1
  CPU usage         : 0.0%
  Memory usage      : 0.6%
```

show processes memory

Show data-plane's main heap usage, data-plane's stats heap usage, and processes' memory usage.

Example:

```
soodar# show processes memory
Dataplane memory heap:
----------------------
Thread 0 vpp_main
  base 0x7fffb692a000, size 1g, locked, unmap-on-destroy, name 'main heap'
    page stats: page-size 4K, total 262144, mapped 24279, not-mapped 237865
      numa 0: 24279 pages, 94.84m bytes
    total: 1023.99M, used: 90.17M, free: 933.83M, trimmable: 933.81M

Dataplane stats heap:
---------------------
Stats segment
base 0x7fffb0371000, size 31.99m, locked, name 'stat segment'
  total: 31.99M, used: 758.05K, free: 31.26M, trimmable: 30.30M
    free chunks 16 free fastbin blks 0
    max total allocated 31.99M

System processes memory status:
-------------------------------
    PID Text       Data        RSS      Total Name
      1    0      90620       7460      22655 systemd
      2    0          0          0          0 kthreadd
      3    0          0          0          0 rcu_gp
      4    0          0          0          0 rcu_par_gp
      6    0          0          0          0 kworker/0:0H-kblockd
```

(continues on next page)

(continued from previous page)

```
      8    0          0          0          0 mm_percpu_wq
      9    0          0          0          0 ksoftirqd/0
     10    0          0          0          0 rcu_preempt
     11    0          0          0          0 migration/0
     12    0          0          0          0 cpuhp/0
     13    0          0          0          0 cpuhp/1
     14    0          0          0          0 migration/1
     15    0          0          0          0 ksoftirqd/1
     17    0          0          0          0 kworker/1:0H-kblockd
     18    0          0          0          0 kdevtmpfs
     19    0          0          0          0 netns
     20    0          0          0          0 rcu_tasks_kthre
     21    0          0          0          0 kauditd
     23    0          0          0          0 kworker/0:1-events
     24    0          0          0          0 oom_reaper
     25    0          0          0          0 writeback
     26    0          0          0          0 kcompactd0
     27    0          0          0          0 khugepaged
     40    0          0          0          0 cryptd
     59    0          0          0          0 kblockd
     60    0          0          0          0 blkcg_punt_bio
     61    0          0          0          0 tpm_dev_wq
     62    0          0          0          0 ata_sff
     63    0          0          0          0 md
     64    0          0          0          0 watchdogd
     65    0          0          0          0 kswapd0
     67    0          0          0          0 acpi_thermal_pm
     69    0          0          0          0 tpm-vtpm
     70    0          0          0          0 nvme-wq
     71    0          0          0          0 nvme-reset-wq
     72    0          0          0          0 nvme-delete-wq
     73    0          0          0          0 scsi_eh_0
     74    0          0          0          0 scsi_tmf_0
     75    0          0          0          0 scsi_eh_1
     76    0          0          0          0 scsi_tmf_1
     77    0          0          0          0 scsi_eh_2
     78    0          0          0          0 scsi_tmf_2
     79    0          0          0          0 scsi_eh_3
     80    0          0          0          0 scsi_tmf_3
     81    0          0          0          0 scsi_eh_4
     82    0          0          0          0 scsi_tmf_4
     83    0          0          0          0 scsi_eh_5
     84    0          0          0          0 scsi_tmf_5
     89    0          0          0          0 kworker/u4:6-events_unbound
     91    0          0          0          0 kworker/0:1H-kblockd
     92    0          0          0          0 kworker/1:2-rcu_gp
     93    0          0          0          0 raid5wq
     94    0          0          0          0 dm_bufio_cache
     95    0          0          0          0 ipv6_addrconf
     96    0          0          0          0 kworker/u5:0
    101    0          0          0          0 jbd2/sda2-8
    102    0          0          0          0 ext4-rsv-conver
```

(continues on next page)

(continued from previous page)

```
    103    0          0          0          0 kworker/1:1H-events_highpri
    108    0          0          0          0 jbd2/sda4-8
    109    0          0          0          0 ext4-rsv-conver
    150    0      13196       3960       3299 systemd-udevd
    179    0          0          0          0 scsi_eh_6
    180    0          0          0          0 scsi_tmf_6
    181    0          0          0          0 scsi_eh_7
    182    0          0          0          0 scsi_tmf_7
    183    0          0          0          0 kworker/1:3-events_power_efficient
    199    0     151388       1028      37847 rngd
    307    0      12652       5172       3163 soosys
    311    0      49220      16556      12305 systemd-journal
    318    0       2420       1628        605 atd
    320    0       2920       1844        730 crond
    321    0       4328       3448       1082 dbus-daemon
    332    0      14084      11056       3521 snmpd
    334    0      11172       6800       2793 snmptrapd
    335    0       4364       2400       1091 chronyd
    340    0   85217384     135968   21304346 vpp_main
    348    0       6080       4348       1520 systemd-logind
    350    0       2368       1812        592 agetty
    351    0       3656       3096        914 login
    352    0    1297464      12912     324366 charon-systemd
    354    0     440432      21636     110108 f2b/server
    418    0      16364      11720       4091 watchfrr
    433    0     425376      18492     106344 zebra
    438    0       9320       4756       2330 staticd
    458    0          0          0          0 kworker/u4:0-events_unbound
    657    0       7784       6296       1946 systemd
    658    0      91472       2068      22868 (sd-pam)
    663    0      48612      43816      12153 vtysh
   5763    0          0          0          0 kworker/0:2-mm_percpu_wq
   8135   84       4283       2472       1092 ps
```

show hardware {cpu | disk | memory}

Show information about the router's hardware.

Example:

```
n1# show hardware cpu disk memory
CPU information
---------------
  Architecture      : X86_64
  Name              : Intel(R) Celeron(R) CPU  J1900  @ 1.99GHz
  Physical cores    : 4
  Total cores       : 4
  Max Frequency     : 2415.70Mhz
  Min Frequency     : 1332.80Mhz
  Current Frequency : 1876.04Mhz
  NUMA Nodes        : 1
  Total CPU Usage   : 51.7%
  Per Core Information:
```

(continues on next page)

(continued from previous page)

```
    Core 0:
      Type         : Physical
      Physical Core: 0
      NUMA Node    : 0
      Usage        : 0.0%
    Core 1:
      Type         : Physical
      Physical Core: 1
      NUMA Node    : 0
      Usage        : 100.0%
    Core 2:
      Type         : Physical
      Physical Core: 2
      NUMA Node    : 0
      Usage        : 68.8%
    Core 3:
      Type         : Physical
      Physical Core: 3
      NUMA Node    : 0
      Usage        : 37.9%

Memory information
------------------
  Total            : 3.74G
  Available        : 1.33G
  Used             : 2.21G
  Percentage       : 64.6%

  NUMA 0 Total     : 3.65G
  NUMA 0 Available : 1.31G
  NUMA 0 Used      : 2.35G
  NUMA 0 Percentage: 64.2%


Partitions and Usage
--------------------
  Device: /dev/root
    Mountpoint      : /
    File system type: ext4
    Total Size      : 3.44G
    Used            : 1.11G
    Free            : 2.13G
    Percentage      : 34.2%
  Device: /dev/sda4
    Mountpoint      : /data
    File system type: ext4
    Total Size      : 21.96G
    Used            : 38.30M
    Free            : 20.79G
    Percentage      : 0.2%
  Device: /dev/sda1
    Mountpoint      : /boot/efi
```

(continues on next page)

(continued from previous page)

```
    File system type: vfat
    Total Size      : 15.95M
    Used            : 856064
    Free            : 15.13M
    Percentage      : 5.1%
```

show memory control-plane

Show information on how much memory is used by control-plane's processes:

Example:

```
soodar# show memory control-plane
top - 11:26:57 up  2:31,  0 users,  load average: 1.64, 0.76, 0.56
Tasks:  13 total,   0 running,  13 sleeping,   0 stopped,   0 zombie
%Cpu(s):  3.7 us,  1.2 sy,  0.1 ni, 91.4 id,  3.1 wa,  0.0 hi,  0.5 si,  0.0 st
KiB Mem : 14322432 total,  5440116 free,  4352300 used,  4530016 buff/cache
KiB Swap:  2097148 total,  2097148 free,        0 used.  9377520 avail Mem

PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
164 frr       20   0  311388   7792   2224 S   0.0  0.1   0:00.00 bgpd
297 frr       20   0   85136   5416   3136 S   0.0  0.0   0:00.00 eigrpd
288 frr       20   0   85556   5960   3436 S   0.0  0.0   0:00.00 isisd
273 frr       20   0   85736   5824   3384 S   0.0  0.0   0:00.00 ldpd
217 frr       20   0   84248   5072   4152 S   0.0  0.0   0:00.00 ldpd
216 frr       20   0   84096   5052   4140 S   0.0  0.0   0:00.00 ldpd
266 frr       20   0   85432   5628   3172 S   0.0  0.0   0:00.00 ospf6d
192 frr       20   0   86036   6456   3740 S   0.0  0.0   0:00.03 ospfd
176 frr       20   0   85124   5684   3416 S   0.0  0.0   0:00.00 ripd
184 frr       20   0   84812   5488   3372 S   0.0  0.0   0:00.00 ripngd
281 frr       20   0   84628   4028   2168 S   0.0  0.0   0:00.00 staticd
100 root      20   0   83924   3676   2432 S   0.0  0.0   0:00.04 watchfrr
154 frr       20   0 2689096  27420   5592 S   0.0  0.2   0:00.02 zebra
```

show memory control-plane details

Show information on how much memory is used by control-plane's processes in details

Example:

```
soodar# show memory control-plane details
System allocator statistics:
  Total heap allocated:  1584 KiB
  Holding block headers: 0 bytes
  Used small blocks:     0 bytes
  Used ordinary blocks:  1484 KiB
  Free small blocks:     2096 bytes
  Free ordinary blocks:  100 KiB
  Ordinary blocks:       2
  Small blocks:          60
  Holding blocks:        0
(see system documentation for 'mallinfo' for meaning)
--- qmem libfrr ---
Buffer                         :          3      24                  72
Buffer data                    :          1    4120                4120
```

(continues on next page)

(continued from previous page)

```
Host config                    :          3  (variably sized)        72
Command Tokens                 :       3427      72               247160
Command Token Text             :       2555  (variably sized)     83720
Command Token Help             :       2555  (variably sized)     61720
Command Argument               :          2  (variably sized)        48
Command Argument Name          :        641  (variably sized)     15672
[...]
--- qmem Label Manager ---
--- qmem zebra ---
ZEBRA VRF                      :          1     912                 920
Route Entry                    :         11      80                 968
Static route                   :          1     192                 200
RIB destination                :          8      48                 448
RIB table info                 :          4      16                  96
Nexthop tracking object        :          1     200                 200
Zebra Name Space               :          1     312                 312
--- qmem Table Manager ---
```

Below these statistics, statistics on individual memory allocation types in SoodarOS (so-called *MTYPEs*) is printed:

- the first column of numbers is the current count of allocations made for the type (the number decreases when items are freed.)
- the second column is the size of each item. This is only available if allocations on a type are always made with the same size.
- the third column is the total amount of memory allocated for the particular type, including padding applied by malloc. This means that the number may be larger than the first column multiplied by the second. Overhead incurred by malloc's bookkeeping is not included in this, and the column may be missing if system support is not available.

When executing this command from `vtysh`, each of the daemons' memory usage is printed sequentially. You can specify the daemon's name to print only its memory usage.

show history

Dump the vtysh cli history.

logmsg LEVEL MESSAGE

Send a message to all logging destinations that are enabled for messages of the given severity.

find REGEX...

This command performs a regex search across all defined commands in all modes. As an example, suppose you're in enable mode and can't remember where the command to turn OSPF segment routing on is:

```
frr# find router-id
  (config)  router-id A.B.C.D [vrf NAME]
```

The CLI mode is displayed next to each command. In this example, `router-id` is under the *config* mode.

Similarly, suppose you want a listing of all commands that contain "l2vpn" and "neighbor":

```
frr# find l2vpn.*neighbor
  (view)  show [ip] bgp l2vpn evpn neighbors <A.B.C.D|X:X::X:X|WORD> advertised-
↪routes [json]
  (view)  show [ip] bgp l2vpn evpn neighbors <A.B.C.D|X:X::X:X|WORD> routes [json]
```

(continues on next page)

(continued from previous page)

```
  (view)  show [ip] bgp l2vpn evpn rd ASN:NN_OR_IP-ADDRESS:NN neighbors <A.B.C.
↪D|X:X::X:X|WORD> advertised-routes [json]
  (view)  show [ip] bgp l2vpn evpn rd ASN:NN_OR_IP-ADDRESS:NN neighbors <A.B.C.
↪D|X:X::X:X|WORD> routes [json]
  ...
```

Note that when entering spaces as part of a regex specification, repeated spaces will be compressed into a single space for matching purposes. This is a consequence of spaces being used to delimit CLI tokens. If you need to match more than one space, use the `\s` escape.

POSIX Extended Regular Expressions are supported.

show thread cpu control-plane [details [r|w|t|e|x]]

This command displays control-plane run statistics for all the different event types. If no options is specified all different run types are displayed together. Additionally you can ask to look at (r)ead, (w)rite, (t)imer, (e)vent and e(x)ecute thread event types.

Pipe Actions

CLI supports optional modifiers at the end of commands that perform postprocessing on command output or modify the action of commands. These do not show up in the `?` or `TAB` suggestion lists.

... | include REGEX

Filters the output of the preceding command, including only lines which match the POSIX Extended Regular Expression `REGEX`. Do not put the regex in quotes.

Examples:

```
Soodar# show ip bgp sum json | include remoteAs
      "remoteAs":0,
      "remoteAs":455,
      "remoteAs":99,
```

```
Soodar# show run | include neigh.*[0-9]{2}\.0\.[2-4]\.[0-9]*
 neighbor 10.0.2.106 remote-as 99
 neighbor 10.0.2.107 remote-as 99
 neighbor 10.0.2.108 remote-as 99
 neighbor 10.0.2.109 remote-as 99
 neighbor 10.0.2.110 remote-as 99
 neighbor 10.0.3.111 remote-as 111
```

... | exclude REGEX

Filters the output of the preceding command, including only lines which **don't** match the POSIX Extended Regular Expression `REGEX`. Do not put the regex in quotes.

... | section REGEX

Filters the output of the preceding command, including only sections which match the POSIX Extended Regular Expression `REGEX`. Do not put the regex in quotes.

Example:

```
n2# show running-config | section interface\swireguard[1-3]0
interface wireguard10
 bridge-group 100 split-horizon group 0
```

(continues on next page)

(continued from previous page)

```
 wireguard source 200.2.3.2
 wireguard private-key n2key1
 wireguard port 51820
 wireguard peer n3
  public-key D3309A5B6BF9FEC26710852AB0D6F6E5783F9343478933788D6C0BBB204FED4A
  endpoint 200.2.3.3 port 51820
  allowed-ip 200.4.4.4/32
 no shutdown
 ip address 10.200.200.1/32
interface wireguard20
 wireguard source 222.2.3.2
 wireguard private-key n2key2
 wireguard port 51821
 wireguard peer n3
  public-key 3B73F9AFBBDC9C7C14C4F1108381F704050137990418C500B1F8465A13EDD637
  allowed-ip 10.0.1.2/32
  allowed-ip 10.0.3.2/32
  allowed-ip 222.4.4.4/32
 no shutdown
 ip address 10.200.200.2/32
interface wireguard30
 wireguard source 222.2.3.2
 wireguard private-key n2key3
 wireguard port 51822
 wireguard peer n3
  public-key 2F12ACA8B029112BA405286239D38CD43210AA713C7D7E73362C28A25AA04439
  allowed-ip 203.4.4.4/32
 no shutdown
 ip address 10.200.200.3/32
```

... | section-exclude REGEX

Filters the output of the preceding command, including only sections which **don't** match the POSIX Extended Regular Expression `REGEX`. Do not put the regex in quotes.

Example:

```
soodar# show running-config | section-exclude interface
Building configuration...

Current configuration:
!
hostname soodar
no ipv6 forwarding
no zebra nexthop kernel enable
security passwords min-length 8
log syslog errors
log monitor
no banner motd
!
ip name-server 4.2.2.4
ntp server ir.pool.ntp.org iburst burst
!
ip route 0.0.0.0/0 192.168.1.1
```

(continues on next page)

(continued from previous page)

```
line vty
!
end
```

1.2.4 Filtering

FRR provides many very flexible filtering features. Filtering is used for both input and output of the routing information. Once filtering is defined, it can be applied in any direction.

IP Prefix List

ip prefix-list provides the most powerful prefix based filtering mechanism. In addition to *access-list* functionality, *ip prefix-list* has prefix length range specification and sequential number specification. You can add or delete prefix based filters to arbitrary points of prefix-list using sequential number specification.

If no ip prefix-list is specified, it acts as permit. If *ip prefix-list* is defined, and no match is found, default deny is applied.

ip prefix-list NAME (permit|deny) PREFIX [le LEN] [ge LEN]

ip prefix-list NAME seq NUMBER (permit|deny) PREFIX [le LEN] [ge LEN]

You can create *ip prefix-list* using above commands.

seq
: seq *number* can be set either automatically or manually. In the case that sequential numbers are set manually, the user may pick any number less than 4294967295. In the case that sequential number are set automatically, the sequential number will increase by a unit of five (5) per list. If a list with no specified sequential number is created after a list with a specified sequential number, the list will automatically pick the next multiple of five (5) as the list number. For example, if a list with number 2 already exists and a new list with no specified number is created, the next list will be numbered 5. If lists 2 and 7 already exist and a new list with no specified number is created, the new list will be numbered 10.

le
: Specifies prefix length. The prefix list will be applied if the prefix length is less than or equal to the le prefix length.

ge
: Specifies prefix length. The prefix list will be applied if the prefix length is greater than or equal to the ge prefix length.

Less than or equal to prefix numbers and greater than or equal to prefix numbers can be used together. The order of the le and ge commands does not matter.

If a prefix list with a different sequential number but with the exact same rules as a previous list is created, an error will result. However, in the case that the sequential number and the rules are exactly similar, no error will result.

If a list with the same sequential number as a previous list is created, the new list will overwrite the old list.

Matching of IP Prefix is performed from the smaller sequential number to the larger. The matching will stop once any rule has been applied.

In the case of no le or ge command, the prefix length must match exactly the length specified in the prefix list.

ip prefix-list description

ip prefix-list NAME description DESC

Descriptions may be added to prefix lists. This command adds a description to the prefix list.

Showing ip prefix-list

show ip prefix-list [json]

Display all IP prefix lists.

If the `json` option is specified, output is displayed in JSON format.

show ip prefix-list NAME [json]

Show IP prefix list can be used with a prefix list name.

If the `json` option is specified, output is displayed in JSON format.

show ip prefix-list NAME seq NUM [json]

Show IP prefix list can be used with a prefix list name and sequential number.

If the `json` option is specified, output is displayed in JSON format.

show ip prefix-list NAME A.B.C.D/M

If the command longer is used, all prefix lists with prefix lengths equal to or longer than the specified length will be displayed. If the command first match is used, the first prefix length match will be displayed.

show ip prefix-list NAME A.B.C.D/M longer

show ip prefix-list NAME A.B.C.D/M first-match

show ip prefix-list summary [json]

show ip prefix-list summary NAME [json]

show ip prefix-list detail [json]

show ip prefix-list detail NAME [json]

debug prefix-list NAME match <A.B.C.D/M|X:X::X:X/M> [address-mode]

Execute the prefix list matching code for the specified list and prefix. Shows which entry matched, if any. (`address-mode` is used for PIM RP lookups and skips prefix length checks.)

The return value from this command is success only if the prefix-list result is to permit the prefix, so the command can be used in scripting.

Clear counter of ip prefix-list

clear ip prefix-list [NAME [A.B.C.D/M]]

Clears the counters of all IP prefix lists. Clear IP Prefix List can be used with a specified NAME or NAME and prefix.

1.2.5 Route Maps

Route maps provide a means to both filter and/or apply actions to route, hence allowing policy to be applied to routes.

For a route reflector to apply a `route-map` to reflected routes, be sure to include `bgp route-reflector allow-outbound-policy` in `router bgp` mode.

Route maps are an ordered list of route map entries. Each entry may specify up to four distinct sets of clauses:

Matching Conditions
A route-map entry may, optionally, specify one or more conditions which must be matched if the entry is to be considered further, as governed by the Match Policy. If a route-map entry does not explicitly specify any matching conditions, then it always matches.

Set Actions
A route-map entry may, optionally, specify one or more Set Actions to set or modify attributes of the route.

Matching Policy
This specifies the policy implied if the *Matching Conditions* are met or not met, and which actions of the route-map are to be taken, if any. The two possibilities are:

- *permit*: If the entry matches, then carry out the *Set Actions*. Then finish processing the route-map, permitting the route, unless an *Exit Policy* action indicates otherwise.
- *deny*: If the entry matches, then finish processing the route-map and deny the route (return *deny*).

The *Matching Policy* is specified as part of the command which defines the ordered entry in the route-map. See below.

Call Action
Call to another route-map, after any *Set Actions* have been carried out. If the route-map called returns *deny* then processing of the route-map finishes and the route is denied, regardless of the *Matching Policy* or the *Exit Policy*. If the called route-map returns *permit*, then *Matching Policy* and *Exit Policy* govern further behaviour, as normal.

Exit Policy
An entry may, optionally, specify an alternative *Exit Policy* to take if the entry matched, rather than the normal policy of exiting the route-map and permitting the route. The two possibilities are:

- *next*: Continue on with processing of the route-map entries.
- *goto N*: Jump ahead to the first route-map entry whose order in the route-map is >= N. Jumping to a previous entry is not permitted.

The default action of a route-map, if no entries match, is to deny. I.e. a route-map essentially has as its last entry an empty *deny* entry, which matches all routes. To change this behaviour, one must specify an empty *permit* entry as the last entry in the route-map.

To summarise the above:

	Match	No Match
Permit	action	cont
Deny	deny	cont

action

- Apply *set* statements
- If *call* is present, call given route-map. If that returns a `deny`, finish processing and return `deny`.
- If *Exit Policy* is *next*, goto next route-map entry
- If *Exit Policy* is *goto*, goto first entry whose order in the list is >= the given order.

- Finish processing the route-map and permit the route.

deny
The route is denied by the route-map (return `deny`).

cont
goto next route-map entry

`show route-map [WORD] [json]`
Display data about each daemons knowledge of individual route-maps. If WORD is supplied narrow choice to that particular route-map.

If the `json` option is specified, output is displayed in JSON format.

`clear route-map counter [WORD]`
Clear counters that are being stored about the route-map utilization so that subsuquent show commands will indicate since the last clear. If WORD is specified clear just that particular route-map's counters.

Route Map Command

`route-map ROUTE-MAP-NAME (permit|deny) ORDER`
Configure the *order*'th entry in *route-map-name* with `Match Policy` of either *permit* or *deny*.

Route Map Match Command

`match ip address ACCESS_LIST`
Matches the specified *access_list*

`match ip address prefix-list PREFIX_LIST`
Matches the specified *PREFIX_LIST*

`match ip address prefix-len 0-32`
Matches the specified *prefix-len*. This is a Zebra specific command.

`match ipv6 address ACCESS_LIST`
Matches the specified *access_list*

`match ipv6 address prefix-list PREFIX_LIST`
Matches the specified *PREFIX_LIST*

`match ipv6 address prefix-len 0-128`
Matches the specified *prefix-len*. This is a Zebra specific command.

`match ip next-hop ACCESS_LIST`
Match the next-hop according to the given access-list.

`match ip next-hop address IPV4_ADDR`
This is a BGP specific match command. Matches the specified *ipv4_addr*.

`match ip next-hop prefix-list PREFIX_LIST`
Match the next-hop according to the given prefix-list.

`match ipv6 next-hop ACCESS_LIST`
Match the next-hop according to the given access-list.

match ipv6 next-hop address IPV6_ADDR

This is a BGP specific match command. Matches the specified *ipv6_addr*.

match ipv6 next-hop prefix-list PREFIX_LIST

Match the next-hop according to the given prefix-list.

match as-path AS_PATH

Matches the specified *as_path*.

match metric METRIC

Matches the specified *metric*.

match tag TAG

Matches the specified tag value associated with the route. This tag value can be in the range of (1-4294967295).

match local-preference METRIC

Matches the specified *local-preference*.

match community COMMUNITY_LIST

Matches the specified *community_list*

match peer IPV4_ADDR

This is a BGP specific match command. Matches the peer ip address if the neighbor was specified in this manner.

match peer IPV6_ADDR

This is a BGP specific match command. Matches the peer ipv6 address if the neighbor was specified in this manner.

match peer INTERFACE_NAME

This is a BGP specific match command. Matches the peer interface name specified if the neighbor was specified in this manner.

match peer PEER_GROUP_NAME

This is a BGP specific match command. Matches the peer group name specified for the peer in question.

match source-protocol PROTOCOL_NAME

This is a ZEBRA and BGP specific match command. Matches the originating protocol specified.

match source-instance NUMBER

This is a ZEBRA specific match command. The number is a range from (0-255). Matches the originating protocols instance specified.

Route Map Set Command

set tag TAG

Set a tag on the matched route. This tag value can be from (1-4294967295).

set ip next-hop IPV4_ADDRESS

Set the BGP/PBR nexthop address to the specified IPV4_ADDRESS. For both incoming and outgoing route-maps.

set ip next-hop vrf VRF_NAME IPV4_ADDRESS

Set the PBR nexthop address to the specified IPV4_ADDRESS. This address is looked up from the specified VRF.

set ip next-hop peer-address

Set the BGP nexthop address to the address of the peer. For an incoming route-map this means the ip address of our peer is used. For an outgoing route-map this means the ip address of our self is used to establish the peering with our neighbor.

set ip next-hop unchanged

Set the route-map as unchanged. Pass the route-map through without changing it's value.

set ipv6 next-hop peer-address

Set the BGP nexthop address to the address of the peer. For an incoming route-map this means the ipv6 address of our peer is used. For an outgoing route-map this means the ip address of our self is used to establish the peering with our neighbor.

set ipv6 next-hop prefer-global

For Incoming and Import Route-maps if we receive a v6 global and v6 LL address for the route, then prefer to use the global address as the nexthop.

set ipv6 next-hop global IPV6_ADDRESS

Set the next-hop to the specified IPV6_ADDRESS for both incoming and outgoing route-maps.

set local-preference LOCAL_PREF

Set the BGP local preference to *local_pref*.

set local-preference +LOCAL_PREF

Add the BGP local preference to an existing *local_pref*.

set local-preference -LOCAL_PREF

Subtract the BGP local preference from an existing *local_pref*.

set distance DISTANCE

Set the Administrative distance to DISTANCE to use for the route. This is only locally significant and will not be dispersed to peers.

set weight WEIGHT

Set the route's weight.

set metric <[+|-](1-4294967295)|rtt| +rtt |-rtt>

Set the route metric. When used with BGP, set the BGP attribute MED to a specific value. Use +/- to add or subtract the specified value to/from the existing/MED. Use *rtt* to set the MED to the round trip time or +*rtt*/-*rtt* to add/subtract the round trip time to/from the MED.

set min-metric <(0-4294967295)>

Set the minimum meric for the route.

set max-metric <(0-4294967295)>

Set the maximum meric for the route.

set aigp-metric <igp-metric |(1-4294967295)>

Set the BGP attribute AIGP to a specific value. If `igp-metric` is specified, then the value is taken from the IGP protocol, otherwise an arbitrary value.

set as-path prepend AS_PATH

Set the BGP AS path to prepend.

set as-path exclude AS-NUMBER...

Drop AS-NUMBER from the BGP AS path.

set community COMMUNITY

Set the BGP community attribute.

set ipv6 next-hop local IPV6_ADDRESS

Set the BGP-4+ link local IPv6 nexthop address.

set origin ORIGIN <egp|igp|incomplete>

Set BGP route origin.

set table (1-4294967295)

Set the BGP table to a given table identifier

Route Map Call Command

call NAME

Call route-map *name*. If it returns deny, deny the route and finish processing the route-map.

Route Map Exit Action Command

on-match next

continue

Proceed on to the next entry in the route-map.

on-match goto N

continue N

Proceed processing the route-map at the first entry whose order is >= N

Route Map Optimization Command

route-map ROUTE-MAP-NAME optimization

Enable route-map processing optimization for *route-map-name*. The optimization is enabled by default. Instead of sequentially passing through all the route-map indexes until a match is found, the search for the best-match index will be based on a look-up in a prefix-tree. A per-route-map prefix-tree will be constructed for this purpose. The prefix-tree will compose of all the prefixes in all the prefix-lists that are included in the match rule of all the sequences of a route-map.

Route Map Examples

A simple example of a route-map:

```
route-map test permit 10
 match ip address 10
 set local-preference 200
```

This means that if a route matches ip access-list number 10 it's local-preference value is set to 200.

See *Miscellaneous Configuration Examples* for examples of more sophisticated usage of route-maps, including of the `call` action.

1.2.6 System

System Logging

SoodarOS uses `systemd-journald` as the central logging solution.

debug service snmp

Enable logging for SNMP service. All SNMP logs appear in journald.

debug service mender

The command is used to enable debug logging for the Mender service on a network device. Mender is an over-the-air (OTA) software updater for devices. When the debug logging is enabled using this command, it will display detailed information about the communication between the Mender client and server on journald.

debug service ntpd

Enable logging for NTP service. All NTP logs appear in journald.

debug service dhcp4

Enable logging for DHCP4 server service. All logs appear in journald.

debug dplane fib

Enable data plane(VPP) FIB logs.

debug dplane ipsec

Enable data plane(VPP) IPSec logs.

log rotate max-file-size SIZE

set the limit of how sizeable individual journal files may grow at most. When a limit is reached, it rotates to the next journal file.

- `SIZE`: specifies the maximum size of the log file, and can be specified in bytes, kilobytes (K), megabytes (M), or gigabytes (G).

Note: Default value for the maximum file size is 10M.

---

log rotate max-files (1-1000)

The command is used in devices to set the maximum number of archived log files.

- `(1-1000)`: This specifies the maximum number of log files that can be created. The value must be within the range of 1 to 1000.

For example, if you want to limit the number of log files to 10, you can use the following command:

```
soodar(config)# log rotate max-files 10
```

This command will ensure that only 10 log files are kept at any given time, and older log files will be overwritten as new ones are created.

Note: Default value for the maximum number of log files is 40.

---

log rotate max-use <SIZE>

Control how much disk space the journal may use up at most. The size is capped at 4G. After reaching the limit, it starts removing elder journal files.

- SIZE: specifies the maximum amount of disk space that can be used by log files in kilobytes, megabytes, or gigabytes. The format for specifying size is [number][unit], where the unit can be K (kilobytes), M (megabytes), or G (gigabytes).

For example, to configure the maximum size of log files to 500 MB, the following command can be used:

```
soodar(config)# log rotate max-use 500M
```

Once the specified maximum size is reached, the router automatically rotates the log files, starting with the oldest file. This helps to prevent log files from consuming too much disk space and causing issues with router performance.

Note: Default value for the maximum disk space usage of log files is 30% of the disk.

---

log rotate max-file-life (1-1000)

The maximum time(in days) to store entries in a single journal file before rotating to the next one.

- (1-1000): is the maximum number of days that a log file will be kept before it is rotated.

For example, the following command sets the maximum file life for log files to 7 days:

```
soodar(config)# log rotate max-file-life 7
```

This means that log files will be rotated and archived after 7 days of being created, even if they have not reached their maximum file size.

Note: Default value for log files life is 30 days.

---

log rotate max-retention (1-1000)

The maximum time(in days) to store journal entries. This controls whether journal files containing entries older than the specified period are deleted.

- (1-1000): is the number of days to retain log files for.

log syslog [LEVEL]

Enable logging output to syslog. If the optional second argument specifying the logging level is not present, the default logging level (typically debugging, but can be changed using the deprecated `log trap` command) is used. The `no` form of the command disables logging to syslog. Default log level for syslog is set to `error` level.

log syslog [X:X::X:X|A.B.C.D|HOST] tcp [tls [skip-host-verify]] [port (100-65535)]

The command is used to configure the router or switch to send system log messages to a remote syslog server over a TCP connection with optional Transport Layer Security (TLS) encryption.

- X:X::X:X|A.B.C.D|HOST: This parameter specifies the IP address or hostname of the remote syslog server.
- tcp: This parameter specifies that the log messages should be sent over a TCP connection.
- tls: This optional parameter specifies that the connection should be secured with TLS encryption.
- skip-host-verify: This optional parameter specifies that the hostname of the remote syslog server should not be verified when using TLS.
- port (100-65535): This parameter specifies the TCP port number on which the remote syslog server is listening for syslog messages. The default port number for syslog over TCP is 514.

Example:

```
soodar(config)# ip host logServer 1.1.1.1
soodar(config)# log syslog logServer tcp tls port 6514
```

This command configures the router to send syslog messages over a TCP connection with TLS encryption to the remote syslog server at IP address 192.168.1.100 on port number 6514.

log syslog [HOST] loki [skip-host-verify] [port (100-65535)]

The command is used to configure the router or switch to send system log messages to a remote Loki server.

- X:X::X:X|A.B.C.D|HOST: This parameter specifies the IP address or hostname of the Loki server.
- skip-host-verify: This optional parameter specifies that the hostname of the Loki server should not be verified when using TLS.
- port (100-65535): This parameter specifies the TCP port number on which the Loki server is listening for syslog messages. The default port number for syslog over TCP is 514.

Note: Loki connection uses http or https protocols to communicate. User **must** provide the http or https in address.

Note: Port is a different option. User **must not** provide a port in an address like http://temp.ir:3100. It's wrong!

Example:

```
soodar(config)# log syslog https://192.168.1.1 loki skip-host-verify port␣
↪3100
```

log monitor [LEVEL]

Enable logging output to terminal shell. By default, monitor logging is enabled at the informational level, but this command can be used to change the monitor logging level. If the optional second argument specifying the logging level is not present, the default logging level (typically informational) is used. The no form of the command disables logging to terminal monitors.

log facility [FACILITY]

This command changes the facility used in syslog messages. The default facility is daemon. The no form of the command resets the facility to the default daemon facility.

log record-priority

To include the severity in all messages logged to a file. Use the log record-priority global configuration command. To disable this option, use the no form of the command. By default, the severity level is not included in logged messages.

log timestamp precision [(0-6)]

This command sets the precision of log message timestamps to the given number of digits after the decimal point. Currently, the value must be 0 to 6 (i.e., the maximum precision is microseconds). To restore the default behavior (1-second accuracy), use the no form of the command, or set the precision explicitly to 0.

```
log timestamp precision 3
```

In this example, the precision is set to provide timestamps with millisecond accuracy.

`log commands`
: This command enables the logging of all commands typed by a user to all enabled log destinations.

`show log all [follow]`
: Show all journals logs. If the `follow` mode is enabled, it follows the updates.

`show log mender [follow]`
: Show mender update service logs. If the `follow` mode is enabled, it follows the updates.

`show log ssh [follow]`
: Show SSH service logs. If the `follow` mode is enabled, it follows the updates.

`show log soolog [follow]`
: Show Soodar service logs. We are using *vector* for logging. If the `follow` mode is enabled, it follows the updates.

`show log snmpd [follow]`
: Show SNMP service logs. If the `follow` mode is enabled, it follows the updates.

`show log ntpd [follow]`
: Show NTP service logs. If the `follow` mode is enabled, it follows the updates.

`show log vpp [follow]`
: Show VPP service(data plane) logs. If the `follow` mode is enabled, it follows the updates.

`show log frr [follow]`
: Show FRR service(control plane) logs. If the `follow` mode is enabled, it follows the updates.

`show log ipsec [follow]`
: Show IPSec service logs. If the `follow` mode is enabled, it follows the updates.

`show log kernel [follow]`
: Show kernel and boot logs. If the `follow` mode is enabled, it follows the updates.

`show log ppp [follow]`
: Show PPD process logs. If the `follow` mode is enabled, it follows the updates.

`clear log [syslog]`
: Clear all generated logs. Using the `syslog` keyword makes the journald logs vacuumed; otherwise, the log file is truncated.

System update

SoodarOS uses `mender` as its system update solution. It supports both online and offline updates, and in case of failure, it can roll back to the previous version

Online update

Update the system from a server. It is disabled by default. When an online update is enabled, the system automatically checks the server for available updates and installs if any are present.

Configuration

system update enable

Enable/Disable online update

system update server-url URL

The command is used to configure the URL of the update server on the device.

- URL: represents the URL of the update server where the software image is located. This URL should be reachable from the device for the update to be successful. The update server may be located on a local network or on the internet.

Note: Update server address, should be a URL, and an IP address can't be set.

---

system update update-poll-interval (5-2147483647)

The command is used to configure the interval at which a device polls the software update server for available software updates.

- (5-2147483647): specifies the number of seconds between each poll.

When this command is configured, the device will periodically contact the software update server to check for available software updates. If an update is available, the device will download and install the update.

Note: Default update poll interval is 120 seconds.

---

system update inventory-poll-interval (5-2147483647)

Inventory polling is a process that retrieves and updates the device inventory information such as software and hardware components, serial numbers, firmware versions, and other details and sends them to the update server. By configuring the inventory-poll-interval command, administrators can customize the frequency of inventory polling based on their requirements.

- (5-2147483647): specifies the number of seconds between each poll.

Setting a shorter polling interval can provide more up-to-date inventory information but may consume more device resources, while setting a longer interval can conserve resources but may result in outdated inventory information.

Note: Default inventory poll interval is 150 seconds.

Example:

```
soodar(config)# system update enable
soodar(config)# system update server-url https://update.soodar.ir
soodar(config)# system update update-poll-interval 300
soodar(config)# system update inventory-poll-interval 400
```

Offlline update

Update system from removable storage. The procedure to offline update is simple. One need to:

1. Install an update
2. Reboot
3. Commit the update(to make it persistent) or rollback the update(in case of failure. Reboot without a commit to rollback)

Note: To use offline update, the online update should be disabled

Configuration

system update offline list

List available updates on removable storage

Example:

```
n1(config)# system update offline list
    1   rls-20
    2   rls-21
    3   rls-21.1
```

system update offline install ARTIFACT

Install update from removable storage. ARTFICAT is the relative path of update file from removable storage root, without `.mender` postfix

system update offline commit

Commit latest installed update.

Warning: During the system's booting, no removable storage should be plugged into the router device, or the boot fails.

System backup and restore

The router is equipped with a set of backup/restore tools. Users can choose to create snapshots from *running-config* or *startup-config*. But backups can only be restored to *startup-config*.

A backup snapshot contains: * The router configuration(including tuning profiles and current tune profile). This configuration could be based on running-config or startup-config. * Private keys and imported CAs/Certificates. There's an option to exclude this data. * Router's licensing materials(license, license request and router license keys).

Each snapshot is saved with a unique user-provided tag. The same tag is used to restore the snapshot. The snapshots could be stored in two ways:

1. To remote host and via SFTP
2. To local storage

The snapshots are arranged in repositories of snapshots for a finer grained control. The user can define as many repo as possible in both local and remote sites.

Note: The default repository is *router-backup*.

Note: For keeping integrity and privacy, Each repository is encrypted with the user-provided password and should not be tampered with.

Warning: Backing up private keys to a remote host is ill-advised and should be avoided but if it's needed, consider further safety measures for remote snapshots and their accessibilities.

Commands

copy <system:startup-config|system:running-config> <sftp:|backup:> [no-pki]

Create a snapshot from the current startup-config and save it to a remote host. the *sftp:* URI could contain the username, password and address of the remote computer with the path in remote, repository name and snapshot tag. The *system:* URI contains the repo's name and snapshot tag. If URI is provided, all fields are shown to the user for confirmation; Otherwise, the user is asked for the required information. The *no-pki* option disables backing-up private keys and imported CAs/Certificates.

Note: *sftp* URI is: *sftp:[user]:[password]@[host]:[path]/[repo]:[repo-password]/[tag]*.

Note: *backup* URI is: *backup:[tag]*.

Examples:

```
soodar# ! copy startup-config with full URI
soodar# copy system:startup-config sftp:john:1234@test:/data/backups/router-1:9876/
↪backup1
Address or name of remote host [test]?
Remote host user [john]?
Remote host password [*****]?
Repository path [/data/backups]?
Repository name [router-1]?
Repository password [*****]?
Destination tag [backup1]?
soodar# ! copy startup-config without providing password in URI
soodar# copy system:startup-config sftp:john@192.168.1.2:backup2
Address or name of remote host [192.168.1.2]?
Remote host user [john]?
Remote host password [admin]?
Repository path [/home/john]?
Repository name [router-backup]?
Repository password []?
Destination tag [backup2]?
soodar# ! copy startup-config with providing only repo and tag name
soodar# copy system:startup-config sftp:backups/backup3
Address or name of remote host [192.168.1.1]?
```

(continues on next page)

(continued from previous page)

```
Remote host user [admin]?
Remote host password [admin]?
Repository path [/home/admin]?
Repository name [backups]?
Repository password []?
Destination tag [backup3]?
soodar# ! copy startup-config without providing anything
soodar# copy system:startup-config sftp:
Address or name of remote host [192.168.1.1]?
Remote host user [admin]?
Remote host password [admin]?
Repository path [/home/admin]?
Repository name [router-backup]?
Repository password []?
Destination tag [router-config]?
soodar# ! copy to local backup storage
soodar# copy system:running-config backup:
Destination tag [router-config]? backup4
```

copy <sftp:|backup:> <system:startup-config> [no-license]

Restore a snapshot from the provided source. The *no-license* option disables restoring licensing materials.

Note: restored snapshot takes effect after rebooting the device.

```
soodar# copy backup: system:startup-config
Tag to restore [router-config]? backup4
```

show archive snapshots [sftp:|backup:]

List available snapshots in the source.

```
soodar# show archive snapshots backup:
Repository name [router-backup]?
Repository password []?


   Tag         Host              Time
================================================
r1             soodar   Wed Jun 15 14:07:45 2022
------------------------------------------------
keybackup1     soodar   Fri Jun 24 14:13:22 2022
------------------------------------------------
backup4        soodar   Sun Jul  3 14:50:37 2022
```

show archive config <sftp:|backup:>

Show snapshot contents. only config snapshots can be shown.

```
soodar# show archive config backup:admin-1:1234
Repository name [admin-1]?
Repository password [*****]?
Destination tag [router-config]? r1
```

(continues on next page)

(continued from previous page)

```
r1
==
hostname soodar
no ipv6 forwarding
no zebra nexthop kernel enable
security passwords min-length 8
log syslog errors
log monitor
no banner motd
!
no ntp
!
interface ge1
 no ip address
!
interface ge2
 no ip address
!
interface ge3
 no ip address
!
interface lo
 no ip address
!
interface ge0
 no shutdown
 ip address 192.168.1.55/24
exit
!
end
```

show archive config differences <system:startup-config| system:running-config |sftp:|backup:> \ <system:startup-config|system:running-config | sftp:|backup:>

Compare snapshots and print the differences.

```
soodar# show archive config differences system:running-config backup:r1
Repository name [router-backup]?
Repository password []?
Destination tag [r1]?

running-config                                              r1
==============                                              ==
hostname soodar                                                 hostname soodar
no ipv6 forwarding                                        no ipv6 forwarding
no zebra nexthop kernel enable                                  no zebra␣
 ↪nexthop kernel enable
security passwords min-length 8                                 security␣
 ↪passwords min-length 8
log syslog errors                                         log syslog errors
log monitor                                               log monitor
no banner motd                                                  no banner motd
```

(continues on next page)

(continued from previous page)

```
!                                                    !
no ntp                                                 no ntp
!                                                    !
interface ge1                                          interface ge1
no ip address                                           no ip address
!                                                    !
                              >   interface ge2
                              >    no ip address
                              >   !
interface ge3                                          interface ge3
no ip address                                           no ip address
!                                                    !
interface lo                                           interface lo
no ip address                                           no ip address
!                                                    !
interface ge0                                          interface ge0
no shutdown                                           no shutdown
ip address 192.168.1.55/24                            ip address 192.168.1.
↪55/24
exit                                                        <
!                                                  <
interface ge2                                               <
no shutdown                                        <
ip address 1.1.1.1/24                                       <
exit                                                          exit
!                                                    !
end                                                  end
```

delete <backup:|sftp:>

Delete snapshot from provided source

```
soodar# delete sftp:
Address or name of remote host [192.168.1.1]? 192.168.1.2
Remote host user [admin]? john
Remote host password [admin]?
Repository name [router-backup]?
Repository password []?
Destination tag [router-config]? backup3
```

Prometheus Monitoring

SoodarOS supports both SNMP and Prometheus for monitoring purposes. Users can enable Prometheus monitoring by running *soomon* service on the router. After running and enabling *soomon* service, the router can provide metrics on port 9200.

system service enable soomon

Start *soomon* service to provide Prometheus monitoring.

Note: Currently, *soomon* only works on port 9200. This behavior could change in the future.

System Services

Services are running in the background for accomplishing tasks. These services include:

- NTP: Network Time Protocol service.
- Mender: System update service.
- Soolog: Remote and local syslog service.
- SNMPD: SNMP Services
- VPP: Router service. Restarting this service is like restarting the router.
- soomon: Soodar Prometheus monitoring service.

Commands

`show system service status SERVICE`
Show service status based on the output of the systemd.

`system service restart SERVICE`
Restart a service. If the service is not running, starts the service.

Note: An explicitly disabled service can not be restarted(for example, when a user has set `no ntp` command, one can not restart the NTP service).

System Security

The admin can set the maximum TCP SYN limit to protect the system from SYN flood attacks.

`tcp syn-flood limit (1-4294967295)`
The command is used in devices to configure the maximum number of allowed half-open TCP connections. Half-open connections occur during a SYN flood attack, which is a type of denial of service (DoS) attack. During a SYN flood attack, an attacker sends a large number of TCP SYN packets to the target device, but never completes the connection by sending the ACK packet.

- `(1-4294967295)`: specifies the maximum number of allowed half-open TCP connections.

Note: The default limit is `256`.

1.2.7 Crash Handling System

The Crash Handling System is a crucial component of the system diagnostic and troubleshooting process. It allows for the collection and analysis of critical information when a system experiences a crash or unexpected failure. Two key artifacts used in this process are coredumps and SOS-Reports.

Coredump

A coredump is a file that contains the memory image of a process when it encounters a fatal error or crashes. It captures the state of the process, including its memory contents, registers, and stack trace. The coredump serves as a valuable resource for post-mortem analysis to identify the root cause of the crash. It provides insights into the specific memory locations and code execution paths that led to the failure.

SOS-Report

An SOS-Report, is a comprehensive snapshot of system information and configuration settings at the time of a crash. It includes various logs, kernel parameters, network configuration, hardware details, and other relevant diagnostic data. The SOS-Report provides a holistic view of the system's state, aiding in troubleshooting and identifying potential issues that may have contributed to the crash.

It is important to exercise caution when handling SOS-Reports, as they may contain sensitive information. Before sharing an SOS-Report with the support team or any external party, it is essential to review its contents and ensure that no sensitive or confidential data is included. Sensitive data may include passwords, IP addresses, customer-specific information, or any other data that should not be disclosed. Removing or obfuscating sensitive data helps protect the privacy and security of the system and its users.

To review an SOS-Report for sensitive data, carefully examine the report's content, such as log files and system configuration files. Ensure that any sensitive information is appropriately redacted or removed. It is advisable to follow established data protection guidelines and internal security policies while handling and sharing SOS-Reports.

CLI

`show crashinfo`

The command is used to display information about system crashes or failures that have occurred on a device. It provides details about the crash, including the time of the crash and the program that has crashed.

Example:

`copy crashinfo: sftp:`

The command is used to copy crash information files from a local device to a remote server using the Secure File Transfer Protocol (SFTP).

Note: *sftp* URI is: *sftp:[user]:[password]@[host]:[path]*.

Note: *crashinfo* URI is: *crashinfo:[index]*.

Examples:

```
soodar# ! copy second crashinfo with full URI
soodar# copy crashinfo:2 sftp:john:1234@test:/data/crashinfo
Address or name of remote host [test]?
Remote host user [john]?
Remote host password [*****]?
Remote path [/data/crashinfo]?
Index to export [2]?
soodar# ! copy crashinfo without providing anything
soodar# copy crashinfo: sftp:
```

(continues on next page)

(continued from previous page)

```
Address or name of remote host [192.168.1.1]?
Remote host user [admin]?
Remote host password [admin]?
Remote path [/home/admin]?
Index to export [-1]?
```

delete crashinfo:

The command is used to delete crash-related files stored in the crashinfo directory on a device.

Note: *crashinfo* URI is: *crashinfo:[index]*.

Note: index **0** means every crashinfo, so using this index with delete(or copy) command will remove(or export) all crashes.

1.2.8 SNMP

SNMP (Simple Network Management Protocol) is an application-layer protocol used to manage and monitor network devices. It allows administrators to collect, store, and analyze data from network devices, such as routers, switches, and servers. SNMP works by defining a standardized set of messages, known as protocol data units (PDUs), that can be used to communicate with network devices.

SNMP is based on a client-server architecture, where the client (usually an SNMP manager) sends requests to the server (usually an SNMP agent) to retrieve information about the device's configuration, performance, and status. The SNMP agent on the device receives these requests, processes them, and returns the requested information to the SNMP manager.

AgentX is an extension to SNMP that allows multiple SNMP agents to be controlled by a single SNMP manager. With AgentX, an SNMP manager can communicate with an agent that resides on a different device, enabling centralized management of a large number of network devices. AgentX uses a client-server model, where the SNMP manager acts as the client and the SNMP agent acts as the server.

agentx

Start SNMP Daemon and AgentX on the system

no agentx

Stop SNMP Daemon and AgentX on the system

SNMP Users

To access the SNMP MIBs, one or more users should be available. Currently, only SNMPv3 is supported.

snmp-server user USER auth <md5|sha> PASSWORD [priv des56 PRIV]

The command is used to configure SNMPv3 user authentication and authorization parameters on a device.

- `USER`: is the username of the SNMPv3 user being configured.
- `auth`: specifies the authentication type, either `md5` or `sha`.
- `PASSWORD`: is the authentication passphrase used to authenticate the user.
- `priv`: (optional) specifies the encryption type, which can be `des`.

- PRIV: (optional) is the encryption passphrase used to encrypt SNMPv3 packets.

This command creates a new SNMPv3 user on the device and sets its authentication and encryption parameters. The md5 and sha parameters specify the authentication algorithm used to protect SNMPv3 messages, while the des56 parameter specifies the encryption algorithm used to encrypt the SNMPv3 messages.

If the priv parameter is specified, the user is granted access to SNMPv3 encrypted data. The PRIV parameter specifies the encryption passphrase that is used to encrypt the SNMPv3 packets.

Note: Password length can't be lesser than 8 characters.

Example:

```
soodar(config)# snmp-server user normal-user auth sha 12345678
soodar(config)# snmp-server user priv-user auth sha 12345678 priv des56 87654321
```

The first command creates an SNMP user called "normal-user" with SHA authentication and a password of "12345678".

The second command creates an SNMP user called "priv-user" with SHA authentication, a password of "12345678", and DES56 privacy encryption with a password of "87654321". This means that any SNMP traffic sent from this user will be both authenticated and encrypted

1.2.9 NTP

NTP (Network Time Protocol) is a protocol used to synchronize the clocks of computers and other network devices to a reference time source. The purpose of NTP is to ensure that all devices on a network have a consistent time and that the time is accurate.

NTP operates using a hierarchical system of time sources. At the top of the hierarchy are stratum 1 time servers, which are directly connected to a source of accurate time, such as an atomic clock or GPS satellite. Stratum 2 servers are those that synchronize with stratum 1 servers, and so on down the hierarchy.

To synchronize their clocks using NTP, devices send requests to a time server and receive a response containing the current time. The device then adjusts its own clock to match the time received from the server.

NTP can be configured to use multiple time sources, which provides redundancy in case one source becomes unavailable. NTP can also be configured to use authentication to ensure that time sources are legitimate and not subject to spoofing or tampering.

NTP is widely used in computer networks to ensure accurate timekeeping for various applications, such as log file timestamps, transaction records, and network security protocols that rely on time synchronization.

Using *chrony*, SoodarOS can be an NTP client supporting Version 3 and Version 4 of the NTP protocol

Setting up NTP

Setting up an NTP client is just as simple as providing one(or more) NTP servers and giving needed options.

`ntp server SERVER [OPTIONS]`

The command is used to configure an NTP (Network Time Protocol) server on a device.

- SERVER: is the hostname or IP address of the NTP server

The following are the available options:

- `burst`: With this option, the client will shorten the interval between up to four requests to 2 seconds or less when it cannot get a good measurement from the server.
- `iburst`: With this option, the interval between the first four requests sent to the server will be 2 seconds or less instead of the interval specified by the `minpoll` option.
- `key (1-65535)`: The key option specifies which key (with an ID in the range 1 through 65535) should client use to authenticate requests sent to the server and verify its responses. The server must have the same key for this number configured, otherwise no relationship between the computers will be possible.
- `maxpoll (-6-24)`: This option specifies the maximum interval between requests sent to the server as a power of 2 in seconds. For example, maxpoll 9 indicates that the polling interval should stay at or below 9 (512 seconds). The default is 10 (1024 seconds), the minimum is -6 (1/64th of a second), and the maximum is 24 (6 months).
- `minpoll (-6-24)`: This option specifies the minimum interval between requests sent to the server as a power of 2 in seconds. For example, minpoll 5 would mean that the polling interval should not drop below 32 seconds. The default is 6 (64 seconds), the minimum is -6 (1/64th of a second), and the maximum is 24 (6 months). Note that intervals shorter than 6 (64 seconds) should generally not be used with public servers on the Internet, because it might be considered abuse. A sub-second interval will be enabled only when the server is reachable and the round-trip delay is shorter than 10 milliseconds, i.e. the server should be in a local network.
- `prefer`: Prefer this source over sources without the prefer option.
- `version (3-4)`: This option sets the NTP version of packets sent to the server. The default version is 4.

Example:

```
soodar(config)# ntp server ir.pool.ntp.org burst iburst version 3
```

This command configures the network time protocol (NTP) server to synchronize the local device's clock with the NTP server located at "ir.pool.ntp.org" using the "burst" and "iburst" options to quickly obtain accurate time information. The command also specifies NTP version 3 to be used for this synchronization.

Note: If you have a firewall between the NTP server and the router, you may need to allow NTP traffic (UDP port 123) through the firewall.

Note: Default value for minpoll is 6(64 seconds) and for maxpoll is 10(1024 seconds).

Setting up NTP Authentication

NTP (Network Time Protocol) authentication is a security mechanism that verifies the authenticity of NTP packets received from a trusted NTP server. This mechanism helps to prevent NTP spoofing attacks where attackers send false NTP packets to clients and alter the system time of a device.

Authentication is achieved by using a shared secret key, which is a cryptographic key that is known only to the NTP server and clients. The server includes the key in the NTP packets it sends, and the clients use this key to authenticate the packets and ensure that they are not altered in transit.

Note: NTP authentication is optional and not all NTP servers support it.

Add a New Key

ntp authentication-key (1-65535) sha1 KEYVALUE

The command is used to create a new NTP authentication key and add it to the NTP authentication key database on the device.

- `(1-65535)`: A numeric value between 1 and 65535 that represents the identification number of the key. **This number must match the key ID configured on the NTP server**.
- `sha1`: specifies that the SHA1 algorithm is used for message authentication.
- `KEYVALUE`: The authentication key value that is used to authenticate NTP packets between the client and server. The key value must be a string of up to 20 characters.

Remove a key

no ntp authentication-key (1-65535)

The command is used to remove a previously configured NTP authentication key.

- `(1-65535)`: specifies the key ID number that is being removed.

Enabling and Disabling NTP Authentication

The NTP authentication mechanism only takes effect after it's been explicitly enabled. Without it, all connections to servers that are configured to use authentication would switch to simple unauthenticated mode. Vice versa, one can disable all NTP authentications by simply disabling them.

ntp authentication

The command is used to enable NTP authentication. When this command is configured, NTP packets exchanged between peers will be authenticated using a message digest algorithm to ensure their integrity.

Showing NTP status

You can see information about current time sources that the client is accessing by issuing *show ntp sources* command

show ntp sources [json]

Print current server information.

Example:

Also a json output is available:

```
soodar(config)# do show ntp sources json
{
  "servers":[
    {
      "mode":"^",
```

(continues on next page)

```
soodar# show ntp sources

  .-- Source mode  '^' = server, '=' = peer, '#' = local clock.
 / .- Source state '*' = current best, '+' = combined, '-' = not combined,
| /             'x' = may be in error, '~' = too variable, '?' = unusable.
||                                                 .- xxxx [ yyyy ] +/- zzzz
||      Reachability register (octal) -.           |  xxxx = adjusted offset,
||      Log2(Polling interval) --.      |          |  yyyy = measured offset,
||                                \     |          |  zzzz = estimated error.
||                                 |    |           \
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^* 77.104.70.70                  3   8   347   249  +1050us[+1527us] +/-  103ms
```

(continued from previous page)


```
      "state":"*",
      "address":"194.225.50.25",
      "stratum":"3",
      "polling_interval":"6",
      "reachability_register":"17",
      "last_rx":"6",
      "adjusted_offset":"0.000333353",
      "measured_offset":"-0.001453500",
      "estimated_error":"0.094937660"
    }
  ]
}
```


Also, you can see information about the drift rate and offset estimation process for each of the sources currently being examined by the client.

show ntp sources stats

Print offset estimation for sources.

Example:

```
soodar# show ntp sources stats
                             .- Number of sample points in measurement set.
                            /    .- Number of residual runs with same sign.
                           |    /    .- Length of measurement set (time).
                           |   |    /      .- Est. clock freq error (ppm).
                           |   |   |      /           .- Est. error in freq.
                           |   |   |     |           /         .- Est. offset.
                           |   |   |     |          |          |   On the -.
                           |   |   |     |          |          |   samples. \
                           |   |   |     |          |          |             |
Name/IP Address            NP  NR  Span  Frequency  Freq Skew  Offset  Std Dev
==============================================================================
77.104.70.70               24  17   39m     +0.038      3.994  +9768ns  3015us
```

Manual clock

If you have no or restricted internet connection, you can disable NTP and set the date manually.

`no ntp`

Disable NTP service and remove all its configurations(servers)

Note: NTP service is enabled by default. You should explicitly disable it. To reenable it, just set up NTP and add a server

---

`clock set TIME (1-12) (1-31) (2000-4192)`

Set clock. TIME is current time in hh:mm:ss format.

1.2.10 IPv6 Support

SoodarOS fully supports IPv6 routing. As described so far, SoodarOS supports RIPng, OSPFv3, and BGP-4+. You can give IPv6 addresses to an interface and configure static IPv6 routing information. SoodarOS IPv6 also provides automatic address configuration via a feature called `address auto configuration`. To do it, the router must send router advertisement messages to the all nodes that exist on the network.

Previous versions of SoodarOS could be built without IPv6 support. This is no longer possible.

Enable IPv6

To use IPv6 features, first it's needed to be enabled on interface. There are 2 ways to enable IPv6 on an interface: #. Issue `ipv6 enable` command #. Add an IPv6 address

`ipv6 enable`

The command is used to enable IPv6 processing on an interface. When this command is executed, IPv6 is enabled on the interface and it starts processing IPv6 packets.

> **Warning:** Note that IPv6 can't be enabled on virtual interfaces(like *tunnels* and *loopbacks*).

Note: Although tunnels can't have IPv6 addresses, but they can be passed through IPv6 network(source and destination can be IPv6).

Note: this command only enables IPv6 on an interface. You still need to configure an IPv6 address on the interface to be able to use it for IPv6 communication.

Router Advertisement

Router Advertisement (RA) is a message sent periodically by routers on a network to announce their presence and provide network configuration information to neighboring nodes. The main purpose of RAs is to enable automatic address configuration of nodes and provide other network parameters, such as the default router and prefix information.

RAs are sent by routers on the link-local multicast address FF02::1 (all-nodes multicast address) and are received by all nodes on the link. The frequency of RA transmissions can be configured on the router, typically ranging from a few seconds to several minutes.

When a node receives an RA, it can automatically configure its own IPv6 address using Stateless Address Autoconfiguration (SLAAC), which involves generating an interface identifier based on the network prefix in the RA and the MAC address of the interface. The node can also obtain other network information, such as the default router and DNS server addresses, from the RA.

In addition to SLAAC, routers can also provide additional information in RAs, such as the prefix length, prefix options, hop limit, and MTU, which can be used by the nodes to configure themselves accordingly.

`show ipv6 nd ra-interfaces`

The command is used to display the Router Advertisement (RA) configuration on interfaces that are sending Router Advertisement messages.

`ipv6 nd suppress-ra`

The command is used to suppress the generation of router advertisements (RAs) on an interface. By suppressing the generation of RAs on an interface, the router will not inform other nodes on the network segment of its presence or configuration information.

This command can be useful in scenarios where a router is not the default gateway for hosts on the network segment, or where another router is already sending RAs on the same segment. It can also be used to conserve network bandwidth by reducing the amount of network traffic generated by the router.

The `no` form of this command enables sending RA messages.

`ipv6 nd prefix ipv6prefix [valid-lifetime] [preferred-lifetime] \`
`[off-link] [no-autoconfig] [router-address]`

Configuring the IPv6 prefix to include in router advertisements. Several prefix specific optional parameters and flags may follow:

- `valid-lifetime`: the length of time in seconds during what the prefix is

valid for the purpose of on-link determination. Value `infinite` represents infinity (i.e. a value of all one bits (`0xffffffff`)). Range: `(0-4294967295)` Default: `2592000`

- `preferred-lifetime`: the length of time in seconds during what addresses

generated from the prefix remain preferred. Value `infinite` represents infinity. Range: `(0-4294967295)` Default: `604800`

- `off-link`: indicates that advertisement makes no statement about on-link or

off-link properties of the prefix. Default: not set, i.e. this prefix can be used for on-link determination.

- `no-autoconfig`: indicates to hosts on the local link that the specified prefix

cannot be used for IPv6 autoconfiguration.

Default: not set, i.e. prefix can be used for autoconfiguration.

- `router-address`: indicates to hosts on the local link that the specified

prefix contains a complete IP address by setting R flag.

Default: not set, i.e. hosts do not assume a complete IP address is placed.

ipv6 nd ra-interval [(1-1800)]

The maximum time allowed between sending unsolicited multicast router advertisements from the interface, in seconds. Default: `600`

ipv6 nd ra-interval [msec (70-1800000)]

The maximum time allowed between sending unsolicited multicast router advertisements from the interface, in milliseconds. Default: `600000`

ipv6 nd ra-fast-retrans

RFC4861 states that consecutive RA packets should be sent no more frequently than three seconds apart. FRR by default allows faster transmissions of RA packets in order to speed convergence and neighbor establishment, particularly for unnumbered peering. By turning off ipv6 nd ra-fast-retrans, the implementation is compliant with the RFC at the cost of slower convergence and neighbor establishment. Default: enabled

ipv6 nd ra-retrans-interval [(0-4294967295)]

The value to be placed in the retrans timer field of router advertisements sent from the interface, in msec. Indicates the interval between router advertisement retransmissions. Setting the value to zero indicates that the value is unspecified by this router. Must be between zero or 4294967295 msec. Default: `0`

ipv6 nd ra-hop-limit [(0-255)]

The value to be placed in the hop count field of router advertisements sent from the interface, in hops. Indicates the maximum diameter of the network. Setting the value to zero indicates that the value is unspecified by this router. Must be between zero or 255 hops. Default: `64`

ipv6 nd ra-lifetime [(0-9000)]

The value to be placed in the Router Lifetime field of router advertisements sent from the interface, in seconds. Indicates the usefulness of the router as a default router on this interface. Setting the value to zero indicates that the router should not be considered a default router on this interface. Must be either zero or between value specified with `ipv6 nd ra-interval` (or default) and 9000 seconds. Default: `1800`

ipv6 nd reachable-time [(1-3600000)]

The value to be placed in the Reachable Time field in the Router Advertisement messages sent by the router, in milliseconds. The configured time enables the router to detect unavailable neighbors. The value zero means unspecified (by this router). Default: `0`

ipv6 nd managed-config-flag

Set/unset flag in IPv6 router advertisements which indicates to hosts that they should use managed (stateful) protocol for addresses autoconfiguration in addition to any addresses autoconfigured using stateless address autoconfiguration. Default: not set

ipv6 nd other-config-flag

Set/unset flag in IPv6 router advertisements which indicates to hosts that they should use administered (stateful) protocol to obtain autoconfiguration information other than addresses. Default: not set

ipv6 nd home-agent-config-flag

Set/unset flag in IPv6 router advertisements which indicates to hosts that the router acts as a Home Agent and includes a Home Agent Option. Default: not set

ipv6 nd home-agent-preference [(0-65535)]

The value to be placed in Home Agent Option, when Home Agent config flag is set, which indicates to hosts Home Agent preference. The default value of 0 stands for the lowest preference possible. Default: `0`

ipv6 nd home-agent-lifetime [(0-65520)]

The value to be placed in Home Agent Option, when Home Agent config flag is set, which indicates to hosts Home Agent Lifetime. The default value of 0 means to place the current Router Lifetime value.

Default: `0`

ipv6 nd adv-interval-option

Include an Advertisement Interval option which indicates to hosts the maximum time, in milliseconds, between successive unsolicited Router Advertisements. Default: not set

ipv6 nd router-preference [(high|medium|low)]

Set default router preference in IPv6 router advertisements per RFC4191. Default: medium

ipv6 nd mtu [(1-65535)]

Include an MTU (type 5) option in each RA packet to assist the attached hosts in proper interface configuration. The announced value is not verified to be consistent with router interface MTU.

Default: don't advertise any MTU option.

ipv6 nd rdnss ipv6address [lifetime]

Recursive DNS server address to advertise using the RDNSS (type 25) option described in RFC8106. Can be specified more than once to advertise multiple addresses. Note that hosts may choose to limit the number of RDNSS addresses to track.

Optional parameter:

- `lifetime`: the maximum time in seconds over which the specified address may be used for domain name resolution. Value `infinite` represents infinity (i.e. a value of all one bits (`0xffffffff`)). A value of 0 indicates that the address must no longer be used. Range: `(0-4294967295)` Default: `3 * ra-interval`

Default: do not emit RDNSS option

ipv6 nd dnssl domain-name-suffix [lifetime]

Advertise DNS search list using the DNSSL (type 31) option described in RFC8106. Specify more than once to advertise multiple domain name suffixes. Host implementations may limit the number of honored search list entries.

Optional parameter:

- `lifetime`: the maximum time in seconds over which the specified domain suffix may be used in the course of name resolution. Value `infinite` represents infinity (i.e. a value of all one bits (`0xffffffff`)). A value of 0 indicates that the name suffix must no longer be used. Range: `(0-4294967295)` Default: `3 * ra-interval`

Default: do not emit DNSSL option

Router Advertisement Configuration Example

A small example:

```
interface ge0
 ipv6 enable
 no ipv6 nd suppress-ra
 ipv6 nd prefix 2001:1::/64
```

See also:

- **RFC 2462** (IPv6 Stateless Address Autoconfiguration)
- **RFC 4861** (Neighbor Discovery for IP Version 6 (IPv6))
- **RFC 6275** (Mobility Support in IPv6)
- **RFC 4191** (Default Router Preferences and More-Specific Routes)
- **RFC 8106** (IPv6 Router Advertisement Options for DNS Configuration)

1.2.11 IPFIX

Internet Protocol Flow Information Export (IPFIX) is an IETF protocol and the name of the IETF working group defining the protocol. It was created based on the need for a common, universal standard of export for Internet Protocol flow information from routers, probes and other devices that are used by mediation systems, accounting/billing systems and network management systems to facilitate services such as measurement, accounting, and billing. The IPFIX standard defines how IP flow information is formatted and transferred from an exporter to a collector.

IPFIX flow exporter

An IPFIX flow exporter is a component that collects and forwards network flow information from the device to a collector or analyzer for further analysis, visualization, and reporting.

Flow information includes information about network traffic such as source and destination IP addresses, ports, protocol types, packet and byte counts, timestamps, and other relevant metadata. IPFIX flow exporters can be configured with various filtering and sampling options to reduce the amount of data that is exported, thus improving scalability and efficiency.

Commands

`flow exporter`

The flow exporter command is used to configure and define the parameters of an IP Flow Information Export (IPFIX) exporter.

The flow exporter command enters the flow-exporter configuration mode. In this mode, several commands can be used to configure the exporter.

`destination A.B.C.D`

The destination command is used in the configuration of a flow exporter in IPFIX (Internet Protocol Flow Information Export) to specify the IP address of the system where the flow records will be sent.

- `A.B.C.D`: The IP address of the system where the flow records will be sent.

`source A.B.C.D`

Set IPFIX flow packets source. This address should be valid on the router.

- `A.B.C.D`: The IP address that is used as IPFIX flow packets source in header.

`transport udp (1-65535)`

The command is used to specify the transport protocol to be used for exporting IPFIX data.

- `(1-65535)`: The port number to be used for the UDP transport protocol. This can be any value between 1 and 65535.

Note: Default value is 4739.

IPFIX flow monitor

IPFIX flow definitions

Commands

`flow monitor`

Enter flow monitor configuration mode

`cache timeout active (1-604800)`

It sets the maximum number of seconds that a flow record can remain in the active cache before being aged out. When a cache entry is aged out, it is exported.

- `(1-604800)`: is the number of seconds (in the range of 1 to 604800) that a flow record can remain in the active cache.

Note: Default value is 120

`cache timeout inactive (1-604800)`

The command is used to set the maximum amount of time an inactive flow can remain in the IPFIX cache. If the inactive timeout period expires, the flow is removed from the cache.

- `(1-604800)`: specifies the timeout value in seconds. The minimum value is 1 second and the maximum value is 604800 seconds (7 days).

Note: Default value is 20

`record netflow <ipv4|ipv6> prefix-port`

Start recording flows information containing 5-tuple of `source address`, `destination address`, `protocol`, `source port` ,and `destination port`.

`no record netflow <ipv4|ipv6> prefix-port`

stop recording flows information

`ip flow monitor {output|input}`

Apply flow monitor on an interface.

- `input`: Collect the flows of the ingress traffic.
- `output`: Collect the flows of the egress traffic.

`no ip flow monitor {output|input}`

Remove flow monitor from an interface.

Logging

Debugging logs can be set in case of need.

[no] debug ipfix event

log data plane installation processes and results

Setup IPFIX

To setup IPFIX, one needs to do three things:

1. Define flow exporter
2. Define flow monitor
3. Apply monitor on 1 or more interfaces to collect data

Example configuration

```
soodar(config)# interface ge3
soodar(config-if)# ip address 192.168.1.10/24
soodar(config-if)# flow exporter
soodar(config-flow-exporter)# destination 192.168.1.20
soodar(config-flow-exporter)# source 192.168.1.10
soodar(config-flow-exporter)# transport udp 15200
soodar(config-flow-exporter)# flow monitor
soodar(config-flow-monitor)# cache timeout active 1800
soodar(config-flow-monitor)# cache timeout inactive 15
soodar(config-flow-monitor)# record netflow ipv4 prefix-port
soodar(config-flow-monitor)# interface ge0
soodar(config-if)# ip flow monitor output
```

1.2.12 DHCP

Dynamic Host Configuration Protocol (DHCP) is a network protocol used to automatically assign IP addresses and other network configuration settings, such as subnet mask and default gateway, to devices on a network. DHCP enables network administrators to manage and automate the process of IP address assignment, making it more efficient and less error-prone. It allows devices to connect to a network without requiring manual configuration of network settings, simplifying network setup and maintenance. DHCP is widely used in local area networks (LANs) and is a key component of many enterprise networks.

DHCP Client

A DHCP client is a device that requests and obtains an IP address, subnet mask, default gateway, and other network configuration information from a DHCP server. When a DHCP client is connected to a network, it sends a broadcast request message (DHCPDISCOVER) requesting IP configuration information. The DHCP server responds with an offer (DHCPOFFER) containing the requested information. The client then sends a request (DHCPREQUEST) to the server indicating its acceptance of the offered IP configuration. Finally, the server acknowledges (DHCPACK) the request, and the client can begin to use the assigned IP address and network configuration information to communicate on the network.

SoodarOS uses `DHCPCD` as its DHCP client program to automatically get IP address for interfaces.

DHCP Client Configuration

To configure a DHCP client, you need to enable DHCP on the interface that will receive an IP address from the DHCP server.

`ip dhcp client hostname HOSTNAME`

The command is used to configure a hostname for a DHCP client. When the router or switch sends a DHCP request, it includes the configured hostname in the request packet. This can be useful for identifying the device on the DHCP server, particularly in large networks with many DHCP clients.

- `HOSTNAME`: Specifies the name to be used as the DHCP client hostname.

Note: The default value is the router's hostname.

`ip dhcp client request router`

The command is used to configure the device running as a DHCP client to request the IP address of the default gateway (router) from the DHCP server.

By default, the DHCP client requests the IP address of the default gateway from the DHCP server, so the `no ip dhcp client request router` command can be used to disable this behavior.

Note: When the `no` form of the command is used, it will **deny** the provided gateway address.

`ip dhcp client request dns-nameserver`

Request(deny in case of negating) DNS option from server.

DHCP Server

A DHCP server is a network server that automatically assigns IP addresses and other network configuration parameters to devices on a network, such as subnet mask, default gateway, and DNS servers. It helps simplify network administration by providing a centralized way to manage IP addresses and reduce the likelihood of address conflicts. The DHCP server listens for DHCP requests from clients and responds with the necessary configuration information to enable the client to communicate on the network. DHCP servers are commonly used in home and business networks.

SoodarOS uses Kea DHCP as a backend to provide DHCP server functionalities. These functionalities include:

- Add IP pools and define subnet and IP ranges to allocate to hosts
- Provide clients DNS server addresses
- Provide clients gateway address
- Provide clients NTP server addresses
- Set leasing time for each pool
- Show/Clear bindings

Configuring Pool

ip dhcp pool DHCP4POOL

The command is used to configure a DHCP (Dynamic Host Configuration Protocol) server pool for IPv4 addresses. When configuring a DHCP pool, the network administrator defines a pool of IP addresses that can be assigned dynamically to clients requesting an IP address from the DHCP server. This command is used to create the pool and configure various parameters for the pool such as subnet mask, default gateway, DNS server, lease duration, and more.

- `DHCP4POOL`: Specifies the pool name.

```
soodar(config)# ip dhcp pool p1
soodar(dhcp-config)#
```

network A.B.C.D/M

In DHCP pool configuration, the `network` command is used to specify the network address and the prefix length of the IP address pool to be assigned to DHCP clients. The syntax of this command is as follows:

- `A.B.C.D/M`: Specifies the pool prefix address.

For example, to configure a DHCP pool with the network address of 192.168.1.0/24, the command would be:

```
soodar(config)# ip dhcp pool p1
soodar(dhcp-config)# network 192.168.1.0/24
```

This command tells the DHCP server that it should assign IP addresses within the 192.168.1.0/24 network to DHCP clients.

included-address A.B.C.D A.B.C.D

The command is used to specify a range of addresses that could be used for allocating to clients from the subnet. By default, every IP in the pool subnet is available to offer by DHCP Server and, to change this behavior one should set the included address ranges.

Each pool can have multiple `included-address` commands.

```
soodar(config)# ip dhcp pool p1
soodar(dhcp-config)# network 192.168.1.0/24
soodar(dhcp-config)# included-address 192.168.1.100 192.168.1.120
soodar(dhcp-config)# included-address 192.168.1.140 192.168.1.165
```

dns-server A.B.C.D ...

The `dns-server` command is used to specify the IP address of a Domain Name System (DNS) server for a DHCP pool. When a DHCP client receives an IP address from the pool, the DNS server information is also provided so that the client can perform name resolution using DNS.

- `A.B.C.D ...`: represents the IP address of a DNS server. Multiple IP addresses can be specified in a single command or by adding additional `dns-server` commands.

Note: Up to 8 DNS servers can be set. It is not allowed to set up more than 8 servers.

Example:

Following lines set `1.1.1.1`, `4.2.2.4` and `8.8.8.8` as DNS servers:

```
soodar(config)# ip dhcp pool p1
soodar(dhcp-config)# dns-server 4.2.2.4 1.1.1.1
soodar(dhcp-config)# dns-server 8.8.8.8
```

default-router A.B.C.D ...

The command is used in DHCP pool configuration mode to specify the default gateway IP address to be assigned to DHCP clients.

- `A.B.C.D ...`: Specifies the IP address of the default gateway to be used by DHCP clients. Multiple `default-router` commands can be used in the same DHCP pool configuration to specify multiple default gateway IP addresses.

Note: Up to 8 gateway can be set. It is not allowed to set more than 8 addresses.

For example, to configure a DHCP pool with a default gateway of 192.168.1.1, the following command would be used:

```
soodar(config)# ip dhcp pool p1
soodar(dhcp-config)# default-router 192.168.1.1
```

ntp-server NTP ...

The `ntp-server` command is used to specify the IP address of the Network Time Protocol (NTP) server in a DHCP pool configuration. The NTP server provides accurate time information to clients on the network.

- `NTP`: Specifies the IP address or name of the NTP server.

Note: Up to 8 NTP servers can be set. It is not allowed to set more than 8 servers.

```
soodar(config)# ip dhcp pool p1
soodar(dhcp-config)# ntp-server ir.pool.ntp.org
```

domain-name NAME

Specifies the domain name for clients

```
soodar(config)# ip dhcp pool p1
soodar(dhcp-config)# domain-name example.org
```

lease <(0-365)$days (0-23)$hours (0-59)$minutes|infinite>

The lease command is used in a DHCP pool configuration to specify the lease duration for DHCP clients. The lease duration specifies the amount of time that a client can use an assigned IP address before the lease expires and the client must request a new IP address assignment.

- `days`: Specifies the number of days in the lease duration (0-365).
- `hours`: Specifies the number of hours in the lease duration (0-23).
- `minutes`: Specifies the number of minutes in the lease duration (0-59).
- `infinite`: Specifies an infinite lease duration, which means that the IP address assignment will never expire unless manually revoked.

For example, to configure a lease duration of 7 days and 12 hours, you would use the following command:

```
soodar(config)# ip dhcp pool p1
soodar(dhcp-config)# lease 7 12 0
```

Note: Default lease time is **1 day**.

Configure interfaces

ip dhcp server

The `ip dhcp server` command is used to enable the device to act as a DHCP server.

When this command is executed, the device starts to listen on the interface for incoming DHCP requests from clients. If a client request is received, the DHCP server will assign an IP address, subnet mask, default gateway, and other options (such as DNS servers and NTP servers) to the requesting client.

It should be noted that additional configuration is required to configure the DHCP server, such as defining DHCP pools, specifying options to be assigned to clients, and configuring DNS and NTP servers.

```
soodar(config-if)# ip dhcp server
```

ip address dhcp

The `ip address dhcp` command is used to obtain an IP address dynamically through DHCP (Dynamic Host Configuration Protocol) from a DHCP server.

When this command is used on an interface, the router will act as a DHCP client and request an IP address from a DHCP server. The DHCP server will assign an IP address along with other network settings, such as subnet mask, default gateway, and DNS server address, to the device.

Pool status and management

show ip dhcp pool

Show general information about a pool(s)

```
soodar# show ip dhcp pool p1

Pool p1 :
 Total addresses: 47
 Leased addresses:        1
 Declined addresses:      0
 2 ranges are currently in the pool :
 Range's low       Range's high
 192.168.1.100     192.168.1.120
 192.168.1.140     192.168.1.165
```

show ip dhcp binding [<DHCP4POOL|A.B.C.D>]

The show command is used to display information about DHCP bindings on the device. It can be used with or without a specific pool name or client IP address.

- `DHCP4POOL`: (Optional) Specifies the DHCP pool name for which to display the bindings.
- `A.B.C.D`: (Optional) Specifies the IP address of the DHCP client for which to display the binding.

If a pool name or client IP address is specified, the command displays information about the bindings for that specific entity. If no parameters are given, the command displays information about all DHCP bindings on the device.

```
soodar# show ip dhcp binding
  IP Address   Client hostname   Client HW address   Lease expiration          ␣
↪Pool   Type        State
  1.1.1.100    n4               00:00:00:aa:00:01   Wed Nov  3 11:20:06 2021   p2 ␣
↪   Automatic   Leased
```

In the above example, the command output displays the IP address, client ID or hardware address, lease expiration time, and the type of the DHCP binding.

clear ip dhcp binding <*|A.B.C.D>

The command is used to remove a DHCP address binding from the DHCP server database.

- *: Specifies to clear all DHCP address bindings
- A.B.C.D: Specifies the IP address of the client whose DHCP address binding is to be cleared

Note that clearing a DHCP address binding will release the IP address back to the address pool, allowing it to be assigned to another client.

```
soodar# clear ip dhcp binding *
```

1.2.13 License

SoodarOS uses a license manager to allow users to choose their plans and use the trial version flexibly. To change the licensing, one needs to create a license request and send it to the corporation for signing. Once the signed certificate is imported, its effect is immediate.

Default license

When SoodarOS lacks a license file, it continues to work. But restrictions are applied. These restrictions are:

- Drop supporting ethernets faster than 10Gigabit ethernet.
- Support a maximum of 4 hardware interfaces wit accumulated capacity of 10G.
- Limit VPLS interfaces count to 2.
- Limit VXLAN interfaces count to 2.
- Limit VLAN interfaces count to 2.
- Limit IPSec profiles to 2 profiles.
- Support up to 2 Access-list.
- Limit access-list entries to 10 per ACL.
- Support up to 2 Policy map.
- NAT44 IP pool is limited to 1 pool of 1 IP address.
- NAT44 static entries are limited to 2 entries.
- Limit VRFs to 2 VRFs(not counting default VRF).
- Limit SLA definitions to 2.

- Limit Tracks to 2 each could be tracked by one item.
- Wireguard interfaces is restricted to 1 instances with a maximum of 2 peers.

License request

To import a license, an enrollment is needed. SoodarOS makes a license request on user demand and displays it on the screen to achieve this. The displayed request should be sent for signing.

show license license-request
: Display license request on screen.

Import license

Importing a signed license is by copy-pasting the license on the screen.

license import license terminal
: Import a signed license.

Show license

Checking current limits(and used quotas) is done via `show license` command.

show license
: Show current license limits.

> **Note:** A negative value for a resource limit means that the resource is unlimited.

Example :

```
n1# show license
  License found: Yes
                           Name   Limit   Used
   ------------------------------------------------
            Hardware Interfaces      16      0
       Hardware Interfaces Type       8      -
   Hardware Interfaces Capacity     8Gb    0Gb
                VPLS Interfaces       2      0
               VXLAN Interfaces       3      0
                VLAN Interfaces       2      0
                 IPSec Profiles       4      0
                     QoS Policy       2      0
                    NAT44 Pools       2      0
                 NAT44 Pool IPs      10      -
           NAT44 Static Entries      10      0
              Wireguard Tunnels       2      0
    Peers per Wireguard Tunnels       2      -
                            SLA       3      0
                          Track       3      0
                Tracked objects       3      -
                            VRF       5      0
                 Routes per VRF      10      -
```

(continues on next page)

(continued from previous page)

```
                          ACL        2       0
                  ACE per ACL       10       -

  BGP Support: Available
  MPLS Support: Available
  MP-GRE Support: Available
  EIGRP Support: Available
  OSPF Support: Not available
  OSPFv3 Support: Not available
  RIP Support: Not available
  RIPng Support: Not available
  Multicast Support: Not available
```

1.3 Sooshell

1.3.1 Sooshell

The Sooshell shell leverages ZSH capabilities to provide new features to users. In Sooshell, you have access to both Soodar configuration commands and POSIX shell syntax structures simultaneously. Sooshell offers new capabilities to Soodar users and network administrators in both interactive usage and scripting, allowing you to increase your efficiency.

Interactive Usage

Sooshell presents unique features for interactive user to use, making working with the command-line interface easier, more efficient, and more enjoyable.

Prompt

The Sooshell prompt accurately indicates the status of the user session in terms of the configuration path it is in:

```
router1/c/interface/ge0#
```

For example, the above prompt shows that we have entered the path enable -> config -> interface : *ge0*

```
router1/c/i/g/link-params#
```

Or here we have entered enable -> config -> interface : *ge0* -> link-params.

Auto-completion

Using the auto-completion feature, you can avoid writing full commands and save time. To complete a word, simply press the *TAB* key, and Sooshell will suggest possible words to complete the commands:

```
router1/config# p<TAB>

password    pbr     pbr-map     pseudowire
```

You can select one of the options by pressing the arrow keys or the *TAB* key. Also, if Sooshell finds only one possible way to complete the command, it will automatically complete it without the need to press another key.

```
router1/config# ps<TAB>
router1/config# pseudowire
```

History

Sooshell stores a list of commands you've previously entered. The command history doesn't disappear when you close the session and will be available in the next Sooshell execution. By pressing the up and down arrow keys, you can view the history and re-execute previously entered commands.

Highlighting

Sooshell indicates whether the entered command is correct or not by coloring the initial word of the commands:

- Red color: The command does not exist.
- Green color: The command exists.
- Yellow color: The command is ambiguous as typed so far and needs more of it to be typed.
- Blue color: The command exists but was in the previous configuration path, and executing it will cause us to exit the current path.

Auto-suggestion

Sooshell simultaneously suggests a command in faded text in front of the user's input based on the history of commands the user has entered and the input typed so far. The user can select that suggestion by pressing the right arrow key or continue typing and ignore it.

Scripting

Sooshell's scripting capability allows us to automate many tedious manual tasks. The syntax of Sooshell scripts is the same as POSIX shells in Unix-like operating systems such as Bash and Zsh. The programmability of Sooshell gives network administrators the ability to write and execute scripts for configuration automation and any desired routine. Scripts can also be executed interactively. Just paste the script code you've written previously into the terminal to run it. However, a more efficient method of running scripts is through saving and calling them by their names, which we'll discuss next.

Syntax

In this part, we will show with numerous examples how to make the most of Sooshell's scripting capabilities.

Variable Definition

Defining variables allows us to run our commands for different values by simply changing the value of that variable, without needing to change the command itself.

```
router1/config# # set variable IFNAME to ge0
router1/config# IFNAME=ge0
router1/config# # following command evaluates to 'interface ge0' and
router1/config# interface $IFNAME
router1/c/interface/ge0#
```

Deleting a variable:

```
router1/config# unset IFNAME
```

Command Execution Syntax

The above commands can be written in one line:

```
router1/config# IFNAME=ge0; interface $IFNAME && { no shut; quit; }
```

- Commands are separated by the ; character.
- The && series also acts like ;, with the difference that the second command is only executed if the first command was executed successfully.
- The {} characters place one or more commands in a block.

In the above example, if the value we put in the IFNAME variable is not the name of an existing interface, the *interface $IFNAME* command fails and the commands placed in the block are not executed.

Output Processing

Consider the following command:

```
router1# show interface brief
Interface       Status  VRF             Addresses
---------       ------  ---             ---------
lo              up      default
ge0             up      default         192.168.1.1/24
ge1             up      default         192.168.2.1/24
wg0             down    default         10.10.49.1/16
```

We want to get only the information related to *ge1*. Sooshell allows us to use the output of one command as input for another command. The *grep* command can search for a pattern in its input. We just need to give the name of our desired interface along with the input of the previous command to *grep*:

```
router1# show interface brief | grep ge1
ge1             up      default         192.168.2.1/24
```

Another command we can use is the *awk* command. This command provides the user with more advanced search capabilities. For example, we want to find the IP address of the *ge1* interface:

```
router1# show interface brief | awk 'NR > 1 && $1 == "ge1" { print $4 }'
192.168.2.1/24
```

It's also possible to store the output of a command in a variable to use later, which we do using the *VAR=$(COMMAND)* syntax:

```
router1# ipv4=$(show interface brief | awk '$1 == "ge1" { print $4 }')
```

Conditional Operations

Sometimes we need certain commands to be executed only under specific conditions. In Sooshell, each condition is actually a command where the successful execution of the command means the condition is met. Let's say we want to show all interfaces if there are less than 5 interfaces:

```
router# # get all rows except the first 2 heading rows
router# interfaces=$(show interface brief | tail +3)
router# # count the number of rows
router# count=$(wc -l <<< $interfaces)
router# # check if we have less than 5 rows
router# if [[ $count -lt 5 ]]
if> then
then>     # print all rows
then>     echo $interfaces
then> fi
ge0             up      default         192.168.30.251/24
ge1             up      default         200.1.2.1/24
ge2             down    default
ge3             up      default         1.1.1.1/24
```

- In the first line, we use the tail command to store only from the third line onwards.
- We get the number of rows using the *wc -l* command.
- In the next line, we use the *if* structure to check the mentioned condition
- If the condition is met, we print all rows

Note: The *command arg1 arg2 <<< $variable* syntax gives the value of a variable as an argument to a command.

The general syntax of the conditional control structure is as follows:

```
if command1
then
    command2
    command3
fi
```

If the first command is successful, both subsequent commands are executed. In the example we mentioned, the first command is *[[$count -lt 5]]*. The string *[[* ` is actually a command that is called with the arguments *$count*, *-lt*, and *5*. Its second argument, *-lt*, means less than, and the command is successful and the condition is met if the first argument is less than the third.

Now suppose we want to print only the interface names if the number of interfaces is between 5 and 10, and if it's more, print only their number:

```
router1# # get all rows except the first 2 heading rows
router1# interfaces=$(show interface brief | tail +3)
router1# # count the number of rows
router1# count=$(wc -l <<< $interfaces);
router1# # check if we have less than 5 rows
router1# if [[ $count -lt 5 ]]
if> then
then>     # print all rows
then>     echo $interfaces
then> elif [[ $count -lt 10 ]]
elif> then
elif-then>     # print first column of all rows
elif-then>     echo $interfaces | awk '{print $1}'
elif-then> else
else>     echo "number of interfaces: $count"
else> fi
number of interfaces: 16
```

The complete syntax of the conditional control structure is as follows:

```
if cond1
then
    command1-1
    command1-2
    ...
elif cond2
then
    command2-1
    command2-2
    ...
elif...
fi
```

Other conditional operators:

- Is the length of string a greater than zero? *[[-n a]]*
- Is the length of string a equal to zero? *[[-z a]]*
- Are strings a and b equal? *[[a == b]]*
- Are strings a and b not equal? *[[a != b]]*
- Does string a match regex b? *[[a =~ b]]*
- Is number a less than number b? *[[a -lt b]]*
- Is number a greater than number b? *[[a -gt b]]*
- Is number a less than or equal to number b? *[[a -le b]]*

- Is number a greater than or equal to number b? *[[a -ge b]]*
- Is number a equal to number b? *[[a -eq b]]*
- Is number a not equal to number b? *[[a -ne b]]*

Helper Commands

When you are interacting with Sooshell or running a script, you are always in a configuration path.

```
router1/c/interface/ge0# node
interface
```

With this command, you can get the name of the node you are in. If that node has additional information, you can also obtain it:

```
router1/c/interface/ge0# node
interface
router1/c/interface/ge0# node -x
ge0
```

Also, we can use the following conditional command to check if we are in a specific node or not:

```
router1/c/interface/ge0# if [[ -in interface ]]; then echo "yes"; else echo "no"; fi
yes
```

For Loop

In Sooshell script, you can use a loop to perform repetitive tasks. The for loop in Sooshell has a simple syntax and can be used to perform a task for a range of numbers. For example, here we want to repeat the command for adding a static NAT for ports [1200,1400]:

```
router1/config# for port in {1200..1400}; do
for>     ip nat inside source static tcp "1.1.1.10" $port "200.1.2.2" $port
for> done
```

In this loop, the variable *port* is the loop index that can be used in the commands within the loop. Sometimes we want to execute commands for each element in an array. In that case, we act like the following example:

```
router1/config# # Define variables for VLAN configuration
router1/config# vlan_ids=(10 20 30)
router1/config# vlan_ifname="ge0"
router1/config# # Loop through VLAN configuration
router1/config# for idx in $vlan_ids; do
for>     vlan_name="$vlan_ifname.$idx"
for>     echo "Configuring VLAN $vlan_name ..."
for>     interface "$vlan_name"
for>     encapsulation dot1q $idx
for>     bridge-group 1 split-horizon group 1
for>     no shutdown
for>     exit
for> done
Configuring VLAN ge0.10 ...
```

(continues on next page)

(continued from previous page)

```
Configuring VLAN ge0.20 ...
Configuring VLAN ge0.30 ...
n1/config# echo "VLAN configuration completed."
VLAN configuration completed.
n1/config#
```

In this example, the commands inside the loop are executed 3 times, where idx is equal to 10, 20, and 30.

While Loop

Another type of loop is while loops, which execute the commands inside them as long as their condition is met. In the following example, a script is written that reads the name of a protocol from the input and as long as the input is not empty, it denies that protocol and gets the next protocol from the input:

```
# read protocol names (one per line) and deny them in current ACL
[[ -in "ip-access-list" ]] || {
    echo Error: Must be in ACL node 1>&2
    return 1
}
while read PROTOCOL && [[ -n $PROTOCOL ]]
do
    deny $PROTOCOL any any
    echo denied protocol $PROTOCOL in ACL $(node -x)
done
permit any
```

The read command reads a line from the input and puts it in the variable whose name it received as an argument (here *Protocol*`). If it receives an EOF(Ctrl+D), the command fails and the loop condition is not met.

Functions

Sooshell provides the ability to define functions and call them for code reuse. Functions can receive arguments when called and even return a status code. The function's status code acts like the status code of other commands, with code zero meaning success and other codes meaning failure. Functions can also act like output processing commands, meaning they can be called after the | character to take the output of the previous command as input. Here we define a simple function that fails if all its arguments are not equal:

```
check-equality()
{
    if [[ $# -eq 0 ]]; then
        return 2
    fi
    for arg in $@; do
        if [[ $arg -neq $1 ]]; then
            return 1
        fi
    done
    return 0
}
```

To define a function, we must use the *name(){ ... }* syntax. The function definition is placed between two curly braces. At the beginning of the function, we check if any arguments have been given to the function. The term *$#* represents the number of arguments. We want the function to fail and return value 2 if no argument is given. Then we put a loop that traverses all arguments using the *$@* expression and compares them with the first argument, i.e., *$1* (in general, *$n* means the nth argument.) and if they are not equal, it returns 1 meaning error. Now we want to call the function we've written. The function call is in the form *check-equality arg1 arg2 arg3 ...*. In the example below, in the first line, we use the && operator to check if the execution was successful or not, because if it failed, the next command should not be executed. In the next line, we also ensure the function fails, because the || operator with the meaning "or" causes Sooshell to ignore executing the next command if the first command was successful.

```
check-equality a a a && echo '1st' case passed
check-equality a a b || echo '2nd' case failed
```

Execution output:

```
1st case passed
2nd case failed
```

Examples:

A function to get the names of all interfaces:

```
interfaces()
{
    do sh int brief json | jq -r 'keys[]'
}
```

A function to bring up all VLAN interfaces:

```
vlanup()
{
    if [[ ! -in config ]]; then
        echo this command must be run in config node, not $(node) node 1>&2
        return 1
    fi

    for ifname in $(interfaces); do
        if [[ $ifname =~ .*\\..* ]]; then
            interface $ifname
            no shut
            quit
        fi
    done
}
# usage
valnup
```

A function that receives interface/IP pairs and sets the IPs on the interfaces:

```
addips()
{
    local retcode=0

    if [[ ! -in config ]]; then
        echo this command must be run in config node, not $(node) node 1>&2
```

(continues on next page)

(continued from previous page)

```
        return 1
    fi

    for pair in "$@"
    do
        ifname=($(cut -d '=' -f '1' <<< $pair))
        ipaddr=($(cut -d '=' -f '2' <<< $pair))
        if do sh int json | jq -e -r ".$ifname" &> /dev/null
        then
            echo "On $ifname set IP $ipaddr"
            interface $ifname
            ip address $ipaddr
            no shut
            quit
        else
            echo "interface $ifname does not exist" 1>&2
            retcode=1
        fi
    done
    return $retcode
}
# usage
addips wg0="192.168.1.1/24" wg1="192.168.2.1/24" wg2="192.168.3.1/24"
```

Manual pages

Sooshell provides a manual page for its functionality and other system binaries. To access the manual page of a wanted section, use the *man* command followed by the section name. As an example, the following manual pages are available:

- *man sooshell* : Sooshell manual. It provides a more detailed overview of Sooshell and its features.
- *man awk* : AWK is a powerful text processing tool. The manual of AWK provides an overview of the AWK command and its parameters.
- *man jq* : jq is a lightweight and flexible command-line JSON processor. The manual of jq provides an overview of the jq command and its parameters.

Use *man* commands completion feature to see all available manual pages.

Script management

Script management is a feature that allows users to create, store, and execute custom scripts on the Soodar router. This functionality leverages the router's file management engine.

Scripts are stored in a dedicated filesystem within the router, specifically the "script:" filesystem. This organization ensures that scripts are kept separate from other system files and can be easily managed, accessed, and executed as needed.

CLI

The Sooshell CLI provides a set of commands to manage scripts. The following commands are available:

copy <system:startup-config|system:running-config> script: [force]

This command copies the startup configuration or the running configuration of the router to the script filesystem. It allows you to save the current configuration as a script, which can be useful for simple backup purposes or for creating templates for future configurations.

- `system:startup-config`: Specifies the source to be the saved startup configuration.
- `system:running-config`: Specifies the source to be the currently active running configuration.
- `script:`: Specifies that the source is copied to script filesystem. This URI accepts an optional name parameter to specify the name of the script.
- `force`: Optional parameter to overwrite an existing file. If not provided, the command will fail if the file already exists.

Note: *script:* URI is: *script:[name]*.

Note: If a filename is not specified, the script will be saved as *startup* for the startup config and *running* for the running config.

Examples:

Copy the startup configuration to the script filesystem with default name:

```
router1# copy system:startup-config script:
```

Copy the running configuration to the script filesystem, overwriting any existing file named *running*:

```
router1# copy system:running-config script:running force
```

copy <sftp:|ftp:|http:|https:> script: [force]

This command copies a file from a remote server using various file transfer protocols (SFTP, FTP, HTTP, or HTTPS) to the router's script filesystem. It allows you to download scripts or configuration files from external sources and store them in the router's script directory.

- `sftp:`: Specifies the source to be an SFTP server. the *sftp:* URI could contain the username, password and address of the remote computer with the path in remote. If URI is provided, all fields are shown to the user for confirmation; Otherwise, the user is asked for the required information
- `ftp:`: specifies the source to be an FTP server. the *ftp:* URI could contain the username, password and address of the remote computer with the path in remote. If URI is not provided with username and password, the operation will fail.
- `http[s]:`: specifies the source to be an HTTP[S] server.
- `script:`: Specifies that the source is copied to the script filesystem. This URI accepts an optional name parameter to specify the name of the script.
- `force`: Optional parameter to overwrite an existing file. If not provided, the command will fail if the file already exists.

Note: *script:* URI is: *script:[name]*.

Note: *sftp* URI is: *sftp://[user]:[password]@[host]:[path]*.

Note: *ftp* URI is: *ftp://[user[:password]@]host[:port]/[url-path]*.

Note: If a filename is not specified in the destination, the system will use the original filename from the source.

Examples:

Copy a file from an SFTP server:

```
router1# sftp://user:password@192.168.1.100/scripts/my_script script:
```

Download a script from an HTTP server

```
router1# copy http://example.com/router_scripts/config_template script:
```

Retrieve a file from an FTP server and force overwrite any existing file

```
router1# ftp://anonymous@ftp.example.com/public/network_script script: force
```

Download a file from an HTTPS server and save it with a specific name:

```
router1# copy https://example.com/scripts/script script:new_script
```

`copy script: <sftp:> [force]`

This command copies a file from the router's script filesystem to a remote server using the SFTP (Secure File Transfer Protocol) protocol. It allows you to upload scripts or configuration files from the router to an external SFTP server.

- `script:`: Specifies the source to be the script filesystem.
- `sftp:`: Specifies the destination to be an SFTP server. the *sftp:* URI could contain the username, password and address of the remote computer with the path in remote. If URI is provided, all fields are shown to the user for confirmation; Otherwise, the user is asked for the required information
- `force`: Optional parameter to overwrite an existing file. If not provided, the command will fail if the file already exists.

Note: *script:* URI is: *script:name*.

Note: *sftp* URI is: *sftp://[user]:[password]@[host]:[path]*.

Note: If a filename is not specified in the destination, the system will use the original filename from the source.

Examples:

Copy a script file to an SFTP server:

```
router1# copy script:my_script sftp://user:password@192.168.1.100/backups/
```

Copy a script to a specific directory on the SFTP server with a new name:

```
router1# copy script:router_config sftp://user:password@10.0.0.5/configs/router1_
↪backup
```

copy script: script: [force]

This command copies a file from the router's script filesystem to another file in the script filesystem. It allows you to create a copy of a script or configuration file within the router's script directory.

- `script:`: Specifies the source to be the script filesystem.
- `script:`: Specifies the destination to be the script filesystem.
- `force`: Optional parameter to overwrite an existing file. If not provided, the command will fail if the file already exists.

Note: *script:* URI is: *script:name*.

Note: Both source and destination file names should be provided.

---

delete script:

This command is used to delete files from the router's script filesystem.

- `script:`: Specifies the file to be deleted from the script filesystem.

Note: *script:* URI is: *script:name*.

---

dir script:

This command lists the files in the router's script filesystem.

import script:

This command is used to execute scripts or apply configuration files from the router's script filesystem. It functions similarly to the source command in POSIX shells, reading and executing the contents of the specified file in the current context of the router's CLI.

- `script:`: Specifies the script to be imported from the script filesystem.

Note: *script:* URI is: *script:name*.

Note: When importing a script, the router executes the commands in the script line by line, as if they were entered manually in the CLI.

Note: The execution context is the same as the current CLI session, meaning variables and environment settings are preserved and can be used or modified by the script.

Examples:

content of my_script:

Import the file:

```
router1# import script:my_script
hello world!
router1# echo $var1
hello world!
router1#
```

run script:

This command runs a script from the router's script filesystem. It allows you to execute a script stored in the router's script directory.

- `script:`: Specifies the script to be executed from the script filesystem.

Note: *script:* URI is: *script:name*.

Note: Unlike the 'import' command, which executes commands in the current CLI context, 'run' typically executes the script in a separate environment and the current shell context is not affected.

Examples:

content of my_script:

Run the file:

```
router1# run script:my_script
hello world!
router1# echo $var1

router1#
```

check-syntax script:

This command checks the syntax of a script without executing it. It allows you to verify the correctness of a script before running it.

- `script:`: Specifies the script to be checked from the script filesystem.

Note: *script:* URI is: *script:name*.

Note: The script is checked for syntax errors, but it is not checked for errors in commands spelling or their presence.

---

edit script:

This command opens a script in the router's script filesystem for editing. It allows you to create or modify the contents of a script stored in the router's script directory.

- `script:`: Specifies the script to be edited from the script filesystem.

Note: *script:* URI is: *script:name*.

Note: The script is opened in the **nano** text editor.

`more script:`

This command displays the contents of a script stored in the router's script filesystem. It allows you to view the contents of a script without executing it.

- `script:`: Specifies the script to be displayed from the script filesystem.

Note: *script:* URI is: *script:name*.

`rename script: script:`

This command renames a file in the router's script filesystem. It allows you to change the name of a script stored in the router's script directory.

- `script:`: Specifies the source file to be renamed in the script filesystem.
- `script:`: Specifies the destination file name in the script filesystem.

Note: *script:* URI is: *script:name*.

Note: Both source and destination file names should be provided.

1.4 Tune

1.4.1 System Tune

SoodarOS consist of services working together. These services can be categorized into three classes:

- Data-plane Services
- Control-plane Services
- Security-plane services
- Management Services
- System services

Data-plane Services

Data-plane services are responsible for forwarding, routing, and filtering incoming packets.

Members

- `VPP`: VPP (Vector Packet Processing) is a fast, scalable layer 2-4 multi-platform network stack.

Control-plane Services

All services related to signaling and routing protocols are in this class.

Members

- `FRRouting`: FRRouting (FRR) is a free and open-source Internet routing protocol suite.

Security-plane Services

All services related to IPSec secure tunnels are in this class.

Members

- `Strongswan`: Strongswan is an open-source IPsec-based VPN solution for secure communication over networks.

Management Services

Services responsible for connecting from outside to SoodarOS or managing internal services.

Members

- `SooSLA`: SooSLA is the underlying service that does the IP SLA scheduling and running
- `Sooweb`: The web server for Soodar(used in Amnet product only).

System services

All other services that are used to make the system running are under this class.

Tweaks

Depending on each use case's needs, admins can tweak services and their resources. These tweaks are defined as `Tuning profiles`.

Common tweaks

For all above three categories of services, the following parameters can be set:

- CPU cores: Admin can specify CPU cores that the service is allowed to utilize. This core pinning is done in two ways:
  - Exclusive: The core is only assigned to this service and is not used in OS Scheduler.
  - Shared: The core lists in OS Scheduler and other processes can use them.

Note: For data-plane services, CPU core assigning is done differently. refer to *Data-plane tweaks*

- CPU weight: In case of shared CPUs, The available CPU time is split up among all services relative to their CPU time weight. A higher weight means more CPU time, a lower weight means less. The allowed range is 1 to 10000. Defaults to 100 for all processes.
- Maximum memory: Specify the absolute limit on memory usage of the executed processes in this class.

Hugepages

The Hugepages subsystem is responsible for managing and configuring the usage of hugepages in the system. Hugepages are larger memory pages compared to regular pages, typically ranging from 2MB to 1GB in size. These larger pages can improve system performance by reducing the overhead associated with managing a large number of smaller pages.

Note: The hugepages configured in this subsystem are currently utilized by the data-plane component exclusively.

Parameters

The Hugepages subsystem provides the following configuration parameters:

- Hugepages Size: The hugepages size parameter specifies the size of each hugepage. It determines the amount of memory allocated for each hugepage. Supported sizes may vary depending on the system architecture, but typical sizes include 2MB and 1GB.
- Hugepages Number: The hugepages number parameter specifies the total number of hugepages to be allocated in the system. It determines the overall amount of memory reserved for hugepages. The number of hugepages allocated should be carefully determined based on the system's memory requirements and workload characteristics.

Data-plane tweaks

VPP is a sophisticated and high-performance network that has lots of customizable options.

Data-plane CPU configuration

VPP needs at least one main thread and more optional worker threads. Each of These threads is pinned to a CPU core. The default configuration uses core 0 as the main thread(in shared mode) and no workers. Admin can set main thread/ worker threads, CPU exclusiveness, and service CPU weight.

Data-plane Memory configuration

VPP uses its memory management system, so in the beginning, it `mmap` needed memories from OS heap.

- Main heap: VPP's main memory. Used to store ACL rules, IP Routes,... . Defaults to 1GB.
- Stat heap: Memory assigned to storing stats(like packets count). Defaults to 32MB.

Admin can change these values when needed.

Data-plane buffer configuration

VPP uses per NUMA pre-allocated buffers to process incoming packets. It's advised not to change default values, but admins can change the number of buffers and buffer size.

Data-plane poll sleep

VPP's main loop poll always fetches new packets and uses 100% CPU. An admin can add a fixed sleep between the main loop poll to lower the power usage, CPU usage, and heat production in small devices.

> **Warning:** Enabling poll sleep can cause performance degradation.

Interface order configuration

Network interfaces are sorted by their PCI address. For example, a router with 4 network interfaces and their respective addresses is mapped like:

Table 1: Interfaces' list(original)

PCI Address	Order	Interface Name	MAC Address
00:04.0	0	ge0	0c:61:0c:83:00:00
00:05.0	1	ge1	0c:61:0c:83:00:01
00:06.0	2	ge2	0c:61:0c:83:00:02
00:07.0	3	ge3	0c:61:0c:83:00:03

Under some circumstances, admins may be willing to change this order. SoodarOS Tuning Profiles supports static interface order based on MAC address. Interfaces without a static order use the explained order. For instance, in the above example, admin sets the order of MAC address `0c:61:0c:83:00:00` as `2`. The resulting interface list is:

Table 2: Interfaces' list(reordered)

PCI Address	Order	Interface Name	MAC Address
00:04.0	2	ge2	0c:61:0c:83:00:00
00:05.0	0	ge0	0c:61:0c:83:00:01
00:06.0	1	ge1	0c:61:0c:83:00:02
00:07.0	3	ge3	0c:61:0c:83:00:03

Auto-generated tuning profiles

Auto-generated tuning profiles are pre-configured profiles designed to optimize system performance and resource allocation based on the system's hardware configuration. These profiles provide a balanced and power-saving configuration by assigning CPU cores to the data-plane and setting memory limits for the system and adjusting data-plane memory usage.

CLI Commands

Adding/ Removing tuning profile

system tune profile TPROF

Create and Enter Tuning Profile configuration

- TPROF: specifies the tuning profile name.

Note: Removing current profile does **not** apply default configuration to the system.

```
soodar(config)# system tune profile max-perf
soodar(tune-profile)#
```

Configuring Hugepages

hugepages (2-100000)

The command is used to configure the number of hugepages allocated in the system.

- (2-100000): Specifies the number of hugepages to be allocated. The valid range is from 2 to 100,000.

hugepages size <2|1024>

The command is used to configure the size of hugepages in the system.

- <2|1024>: Specifies the size of the hugepages. You can choose either 2 or 1024, which represents the page size in megabytes.

Configuring Data-plane

data-plane

Enter the Data-plane configuration node.

Configuring Data-plane CPU

cpu main [exclusive] (1-256)

Assign entered CPU core as VPP main thread core.

- `exclusive`: Specifies if the CPU core should be isolated from the OS scheduler.
- `(1-256)`: CPU core number.

Note: CPU Core 0 is reserved for OS kernel.

---

no cpu main [[exclusive] (1-256)]

Change current assigned CPU to the main thread to default(Shared CPU 0).

cpu worker [exclusive] (1-256)...

Add a list of CPU cores to be used as VPP worker threads.

- `exclusive`: Specifies if the CPU core should be isolated from the OS scheduler.
- `(1-256)`: CPU core number.

no cpu worker [[exclusive] (1-256)]

Remove a list of CPU cores from VPP worker threads. If the command is used without any input, remove all VPP worker threads.

```
soodar(config)# system tune profile max-perf
soodar(tune-profile)# data-plane
soodar(tune-dp-cfg)# cpu main exclusive 1
soodar(tune-dp-cfg)# cpu worker exclusive 2 3 4 5
soodar(tune-dp-cfg)# cpu worker 6 7
```

Configuring Data-plane Memory

memory heap main SIZE

Set the VPP main heap size.

- `SIZE`: specifies the VPP main heap size. can be in bytes or human-readable format.

Note: Default value for main heap size is 1GB.

---

no memory heap main [SIZE]

Reset the VPP main heap size to the default value of 1GB.

memory heap stats SIZE

Set the VPP stats heap size to `SIZE` value.

- `SIZE`: specifies the VPP stats heap size. can be in bytes or human-readable format.

Note: Default value for stats heap size is 32MB.

no memory heap stats [SIZE]

Reset the VPP stats heap size to the default value of 32MB.

```
soodar(config)# system tune profile max-perf
soodar(tune-profile)# data-plane
soodar(tune-dp-cfg)# memory heap main 2G
soodar(tune-dp-cfg)# memory heap stats 64M
soodar(tune-dp-cfg)# memory max 4G
```

Configuring Data-plane Buffers

memory packet-buffer size (2048-65536)

Set the packet buffer's data segment size.

- `(2048-65536)`: specifies the packet buffer's data segment size.

Note: Default value for packet buffer's data segment size is 2048.

Warning: Packet buffer's size is carefully chosen and changing it is not advised.

no memory packet-buffer size [(2048-65536)]

Reset the packet buffer's data segment size to the default value of 2048.

memory packet-buffer count (16384-1049776)

Set the data-plane packet buffers' count.

- `(16384-1049776)`: specifies the number of buffers to be used for packets.

Note: Default packet buffers' count is 16384.

Note: Each network interface consumes about 2000 packet buffer. As a rule of thumb, to calculate the needed packet buffers one can use `2000 * Number of interfaces + 1000`.

no memory packet-buffer count [(16384-1049776)]

Reset the data-plane packet buffers' count to the default value of 16384

Configuring Data-plane Poll Sleep

poll sleep (0-10000)

Set fixed sleep between VPP's main loop polls. The sleep is in microseconds.

no poll sleep [(0-10000)]

Disable fixed sleep between VPP's main loop polls.

```
soodar(config)# system tune profile max-perf
soodar(tune-profile)# data-plane
soodar(tune-dp-cfg)# poll sleep 100
```

Configuring Control-plane

control-plane

Enter Control-plane configuration node.

Configuring Security Plane

security-plane

Enter security-plane configuration node.

Configuring Management Plane

management-plane

Enter Management-plane configuration node.

Configuring System services

system

Enter system services configuration node.

Common CPU Configuration

cpu [exclusive] (1-256)...

Add a list of CPU cores for threads of this class's services.

- `exclusive`: Specifies if the CPU core should be isolated from the OS scheduler.
- `(1-256)`: CPU core number.

no cpu [[exclusive] (1-256)]

Remove a list of CPU cores from services of this class. If the command is used without any input, remove all CPUs.

cpu weight (1-10000)

The cpu weight command is used to adjust the CPU usage priority for a process or group of processes. Useful when CPU cores are shared between the OS scheduler and the service.

- `(1-10000)`: specifies the weight of the process or group of processes, and it must be a number between 1 and 10000. A higher weight means higher priority, and a weight of 1 means the lowest priority.

When using this command, it is important to note that it does not guarantee a certain amount of CPU time to the process or group of processes. Instead, it adjusts the relative priority among them. Therefore, if the system is heavily congested, the processes with higher weights may still be starved of CPU time.

Note: Default weight value is 100

---

no cpu weight [(1-10000)]

Change the service CPU time weight to the default value of 100.

```
soodar(config)# system tune profile max-perf
soodar(tune-profile)# control-plane
soodar(tune-cp-cfg)# cpu exclusive 8
soodar(tune-cp-cfg)# cpu 9 10
soodar(tune-cp-cfg)# cpu weight 1000
```

Common Memory Configuration

memory max SIZE

Limit the service's maximum available memory.

- `SIZE`: specifies the service's maximum available memory. `SIZE` can be in bytes, human-readable format, or a percentage of the system's total memory.

```
soodar(config)# system tune profile max-perf
soodar(tune-profile)# control-plane
soodar(tune-cp-cfg)# memory max 16G
```

Applying tuning profile

After creating a tuning profile, one can apply this profile to SoodarOS by using the following commands.

system tune apply PROFILE

The command is used to apply a tuning profile to SoodarOS. To changes take effect, a machine restart is needed.

- `PROFILE`: is the name of the tuning profile to be applied.

system tune apply default

The command is used to apply the default profile to SoodarOS. To changes take effect, a machine restart is needed.

Note: On the first boot of the device, 2 `balanced` and `powersaving` profiles are generated and are ready to be used(But none of them is applied).

Tips

- CPU 0 is always used by the SoodarOS kernel. Try to assign services to other cores.
- It is best to use exclusive CPU cores for the data plane's threads(main or workers).
- One or 2 cores should be enough for control-plane and management-plane services, they can even be shared with the system kernel.
- Memory usage usually is not a concern, and limiting memory usage could be skipped.
- Put all cores of your router to use!

1.5 Protocols

1.5.1 Bidirectional Forwarding Detection

BFD (Bidirectional Forwarding Detection) stands for Bidirectional Forwarding Detection and it is described and extended by the following RFCs:

- **RFC 5880**
- **RFC 5881**
- **RFC 5882**
- **RFC 5883**

BFDd Commands

bfd

Opens the BFD daemon configuration node.

peer <A.B.C.D|X:X::X:X> [{multihop|local-address <A.B.C.D|X:X::X:X>|interface IFNAME|vrf NAME}]

Creates and configures a new BFD peer to listen and talk to.

multihop tells the BFD daemon that we should expect packets with TTL less than 254 (because it will take more than one hop) and to listen on the multihop port (4784). When using multi-hop mode *echo-mode* will not work (see **RFC 5883** section 3).

local-address provides a local address that we should bind our peer listener to and the address we should use to send the packets. This option is mandatory for IPv6.

interface selects which interface we should use.

vrf selects which domain we want to use.

profile WORD

Creates a peer profile that can be configured in multiple peers.

Deleting the profile will cause all peers using it to reset to the default values.

show bfd [vrf NAME] peers [json]

Show all configured BFD peers information and current status.

show bfd [vrf NAME] peer <WORD|<A.B.C.D|X:X::X:X> [{multihop|local-address <A.B.C.D|X:X::X:X>|interface IFNAME}]> [json]

Show status for a specific BFD peer.

show bfd [vrf NAME] peers brief [json]

Show all configured BFD peers information and current status in brief.

show bfd distributed

Show the BFD data plane (distributed BFD) statistics.

Peer / Profile Configuration

BFD peers and profiles share the same BFD session configuration commands.

detect-multiplier (2-255)

Configures the detection multiplier to determine packet loss. The remote transmission interval will be multiplied by this value to determine the connection loss detection timer. The default value is 3.

Example: when the local system has *detect-multiplier 3* and the remote system has *transmission interval 300*, the local system will detect failures only after 900 milliseconds without receiving packets.

receive-interval (10-60000)

Configures the minimum interval that this system is capable of receiving control packets. The default value is 300 milliseconds.

transmit-interval (10-60000)

The minimum transmission interval (less jitter) that this system wants to use to send BFD control packets. Defaults to 300ms.

echo receive-interval <disabled|(10-60000)>

Configures the minimum interval that this system is capable of receiving echo packets. Disabled means that this system doesn't want to receive echo packets. The default value is 50 milliseconds.

echo transmit-interval (10-60000)

The minimum transmission interval (less jitter) that this system wants to use to send BFD echo packets. Defaults to 50ms.

echo-mode

Enables or disables the echo transmission mode. This mode is disabled by default. If you are not using distributed BFD then echo mode works only when the peer is also Soodar.

It is recommended that the transmission interval of control packets to be increased after enabling echo-mode to reduce bandwidth usage. For example: *transmit-interval 2000*.

Echo mode is not supported on multi-hop setups (see **RFC 5883** section 3).

shutdown

Enables or disables the peer. When the peer is disabled an 'administrative down' message is sent to the remote peer.

passive-mode

Mark session as passive: a passive session will not attempt to start the connection and will wait for control packets from peer before it begins replying.

This feature is useful when you have a router that acts as the central node of a star network and you want to avoid sending BFD control packets you don't need to.

The default is active-mode (or `no passive-mode`).

minimum-ttl (1-254)

For multi hop sessions only: configure the minimum expected TTL for an incoming BFD control packet.

This feature serves the purpose of thightening the packet validation requirements to avoid receiving BFD control packets from other sessions.

The default value is 254 (which means we only expect one hop between this system and the peer).

BFD Peer Specific Commands

label WORD

Labels a peer with the provided word. This word can be referenced later on other daemons to refer to a specific peer.

profile BFDPROF

Configure peer to use the profile configurations.

Notes:

- Profile configurations can be overridden on a peer basis by specifying non-default parameters in peer configuration node.
- Non existing profiles can be configured and they will only be applied once they start to exist.
- If the profile gets updated the new configuration will be applied to all peers with the profile without interruptions.

BGP BFD Configuration

The following commands are available inside the BGP configuration node.

neighbor <A.B.C.D|X:X::X:X|WORD> bfd

Listen for BFD events registered on the same target as this BGP neighbor. When BFD peer goes down it immediately asks BGP to shutdown the connection with its neighbor and, when it goes back up, notify BGP to try to connect to it.

neighbor <A.B.C.D|X:X::X:X|WORD> bfd check-control-plane-failure

Allow to write CBIT independence in BFD outgoing packets. Also allow to read both C-BIT value of BFD and lookup BGP peer status. This command is useful when a BFD down event is caught, while the BGP peer requested that local BGP keeps the remote BGP entries as staled if such issue is detected. This is the case when graceful restart is enabled, and it is wished to ignore the BD event while waiting for the remote router to restart.

Disabling this disables presence of CBIT independence in BFD outgoing packets and pays attention to BFD down notifications. This is the default.

neighbor <A.B.C.D|X:X::X:X|WORD> bfd profile BFDPROF

Same as command `neighbor <A.B.C.D|X:X::X:X|WORD> bfd`, but applies the BFD profile to the sessions it creates or that already exist.

IS-IS BFD Configuration

The following commands are available inside the interface configuration node.

isis bfd

Listen for BFD events on peers created on the interface. Every time a new neighbor is found a BFD peer is created to monitor the link status for fast convergence.

Note that there will be just one BFD session per interface. In case both IPv4 and IPv6 support are configured then just a IPv6 based session is created.

isis bfd profile BFDPROF

Use a BFD profile BFDPROF as provided in the BFD configuration.

OSPF BFD Configuration

The following commands are available inside the interface configuration node.

ip ospf bfd

Listen for BFD events on peers created on the interface. Every time a new neighbor is found a BFD peer is created to monitor the link status for fast convergence.

ip ospf bfd profile BFDPROF

Same as command `ip ospf bfd`, but applies the BFD profile to the sessions it creates or that already exist.

OSPF6 BFD Configuration

The following commands are available inside the interface configuration node.

ipv6 ospf6 bfd [profile BFDPROF]

Listen for BFD events on peers created on the interface. Every time a new neighbor is found a BFD peer is created to monitor the link status for fast convergence.

Optionally uses the BFD profile `BFDPROF` in the created sessions under that interface.

PIM BFD Configuration

The following commands are available inside the interface configuration node.

ip pim bfd [profile BFDPROF]

Listen for BFD events on peers created on the interface. Every time a new neighbor is found a BFD peer is created to monitor the link status for fast convergence.

Optionally uses the BFD profile `BFDPROF` in the created sessions under that interface.

RIP BFD configuration

The following commands are available inside the interface configuration node:

ip rip bfd

Automatically create BFD session for each RIP peer discovered in this interface. When the BFD session monitor signalize that the link is down the RIP peer is removed and all the learned routes associated with that peer are removed.

ip rip bfd profile BFD_PROFILE_NAME

Selects a BFD profile for the BFD sessions created in this interface.

The following command is available in the RIP router configuration node:

bfd default-profile BFD_PROFILE_NAME

Selects a default BFD profile for all sessions without a profile specified.

BFD Static Route Monitoring Configuration

A monitored static route conditions the installation to the RIB on the BFD session running state: when BFD session is up the route is installed to RIB, but when the BFD session is down it is removed from the RIB.

The following commands are available inside the configuration node:

ip route A.B.C.D/M A.B.C.D bfd [{multi-hop|source A.B.C.D|profile BFDPROF}]

Configure a static route for `A.B.C.D/M` using gateway `A.B.C.D` and use the gateway address as BFD peer destination address.

ipv6 route X:X::X:X/M [from X:X::X:X/M] X:X::X:X bfd [{multi-hop|source X:X::X:X|profile BFDPROF}]

Configure a static route for `X:X::X:X/M` using gateway `X:X::X:X` and use the gateway address as BFD peer destination address.

The static routes when uninstalled will no longer show up in the output of the command `show ip route` or `show ipv6 route`, instead we must use the BFD static route show command to see these monitored route status.

show bfd static route [json]

Show all monitored static routes and their status.

Example output:

```
Showing BFD monitored static routes:

  Route groups:
    rtg1 peer 172.16.0.1 (status: uninstalled):
        2001:db8::100/128

Next hops:
  VRF default IPv4 Unicast:
      192.168.100.0/24 peer 172.16.0.1 (status: uninstalled)

  VRF default IPv4 Multicast:

  VRF default IPv6 Unicast:
```

Note: It is advised to use *IP SLA* feature.

Configuration

Before applying `bfdd` rules to integrated daemons (like BGPd), we must create the corresponding peers inside the `bfd` configuration node.

Here is an example of BFD configuration:

```
bfd
 peer 192.168.0.1
   label home-peer
   no shutdown
 !
!
router bgp 65530
 neighbor 192.168.0.1 remote-as 65531
 neighbor 192.168.0.1 bfd
 neighbor 192.168.0.2 remote-as 65530
 neighbor 192.168.0.2 bfd
 neighbor 192.168.0.3 remote-as 65532
 neighbor 192.168.0.3 bfd
!
```

Peers can be identified by its address (use `multihop` when you need to specify a multi hop peer) or can be specified manually by a label.

Here are the available peer configurations:

```
bfd
 ! Configure a fast profile
 profile fast
  receive-interval 150
  transmit-interval 150
 !

 ! Configure peer with fast profile
 peer 192.168.0.6
  profile fast
  no shutdown
 !

! Configure peer with fast profile and override receive speed.
 peer 192.168.0.7
  profile fast
  receive-interval 500
  no shutdown
 !

 ! configure a peer on an specific interface
 peer 192.168.0.1 interface ge0
  no shutdown
```

(continues on next page)

(continued from previous page)

```
!

! configure a multihop peer
peer 192.168.0.2 multihop local-address 192.168.0.3
  shutdown
!

! configure a peer in a different vrf
peer 192.168.0.3 vrf foo
 shutdown
!

! configure a peer with every option possible
peer 192.168.0.4
 label peer-label
 detect-multiplier 50
 receive-interval 60000
 transmit-interval 3000
 shutdown
!

! configure a peer on an interface from a separate vrf
peer 192.168.0.5 interface ge1 vrf vrf2
 no shutdown
!

! remove a peer
no peer 192.168.0.3 vrf foo
```

Status

You can inspect the current BFD peer status with the following commands:

```
soodar# show bfd peers
BFD Peers:
        peer 192.168.0.1
                ID: 1
                Remote ID: 1
                Status: up
                Uptime: 1 minute(s), 51 second(s)
                Diagnostics: ok
                Remote diagnostics: ok
                Peer Type: dynamic
                Local timers:
                        Detect-multiplier: 3
                        Receive interval: 300ms
                        Transmission interval: 300ms
                        Echo receive interval: 50ms
                        Echo transmission interval: disabled
                Remote timers:
                        Detect-multiplier: 3
```

(continues on next page)

(continued from previous page)

```
                        Receive interval: 300ms
                        Transmission interval: 300ms
                        Echo receive interval: 50ms

        peer 192.168.1.1
                label: router3-peer
                ID: 2
                Remote ID: 2
                Status: up
                Uptime: 1 minute(s), 53 second(s)
                Diagnostics: ok
                Remote diagnostics: ok
                Peer Type: configured
                Local timers:
                        Detect-multiplier: 3
                        Receive interval: 300ms
                        Transmission interval: 300ms
                        Echo receive interval: 50ms
                        Echo transmission interval: disabled
                Remote timers:
                        Detect-multiplier: 3
                        Receive interval: 300ms
                        Transmission interval: 300ms
                        Echo receive interval: 50ms

soodar# show bfd peer 192.168.1.1
BFD Peer:
            peer 192.168.1.1
                label: router3-peer
                ID: 2
                Remote ID: 2
                Status: up
                Uptime: 3 minute(s), 4 second(s)
                Diagnostics: ok
                Remote diagnostics: ok
                Peer Type: dynamic
                Local timers:
                        Detect-multiplier: 3
                        Receive interval: 300ms
                        Transmission interval: 300ms
                        Echo receive interval: 50ms
                        Echo transmission interval: disabled
                Remote timers:
                        Detect-multiplier: 3
                        Receive interval: 300ms
                        Transmission interval: 300ms
                        Echo receive interval: 50ms

soodar# show bfd peer 192.168.0.1 json
{"multihop":false,"peer":"192.168.0.1","id":1,"remote-id":1,"status":"up","uptime":161,
↪"diagnostic":"ok","remote-diagnostic":"ok","receive-interval":300,"transmit-interval
↪":300,"echo-receive-interval":50,"echo-transmit-interval":0,"detect-multiplier":3,
```

(continues on next page)

(continued from previous page)

```
↪"remote-receive-interval":300,"remote-transmit-interval":300,"remote-echo-receive-
↪interval":50,"remote-detect-multiplier":3,"peer-type":"dynamic"}
```

You can inspect the current BFD peer status in brief with the following commands:

```
soodar# show bfd peers brief
Session count: 1
SessionId  LocalAddress         PeerAddress      Status
=========  ============         ===========      ======
1          192.168.0.1          192.168.0.2      up
```

You can also inspect peer session counters with the following commands:

```
soodar# show bfd peers counters
BFD Peers:
     peer 192.168.2.1 interface ge2
             Control packet input: 28 packets
             Control packet output: 28 packets
             Echo packet input: 0 packets
             Echo packet output: 0 packets
             Session up events: 1
             Session down events: 0
             Zebra notifications: 2

     peer 192.168.0.1
             Control packet input: 54 packets
             Control packet output: 103 packets
             Echo packet input: 965 packets
             Echo packet output: 966 packets
             Session up events: 1
             Session down events: 0
             Zebra notifications: 4

soodar# show bfd peer 192.168.0.1 counters
     peer 192.168.0.1
             Control packet input: 126 packets
             Control packet output: 247 packets
             Echo packet input: 2409 packets
             Echo packet output: 2410 packets
             Session up events: 1
             Session down events: 0
             Zebra notifications: 4

soodar# show bfd peer 192.168.0.1 counters json
{"multihop":false,"peer":"192.168.0.1","control-packet-input":348,"control-packet-output
↪":685,"echo-packet-input":6815,"echo-packet-output":6816,"session-up":1,"session-down
↪":0,"zebra-notifications":4}
```

You can also clear packet counters per session with the following commands, only the packet counters will be reset:

```
soodar# clear bfd peers counters

soodar# show bfd peers counters
```

(continues on next page)

(continued from previous page)

```
BFD Peers:
     peer 192.168.2.1 interface ge2
             Control packet input: 0 packets
             Control packet output: 0 packets
             Echo packet input: 0 packets
             Echo packet output: 0 packets
             Session up events: 1
             Session down events: 0
             Zebra notifications: 2

     peer 192.168.0.1
             Control packet input: 0 packets
             Control packet output: 0 packets
             Echo packet input: 0 packets
             Echo packet output: 0 packets
             Session up events: 1
             Session down events: 0
             Zebra notifications: 4
```

Debugging

By default only informational, warning and errors messages are going to be displayed. If you want to get debug messages and other diagnostics then make sure you have *debugging* level enabled:

```
config
log syslog debugging
```

You may also fine tune the debug messages by selecting one or more of the debug levels:

debug bfd network

Toggle network events: show messages about socket failures and unexpected BFD messages that may not belong to registered peers.

debug bfd peer

Toggle peer event log messages: show messages about peer creation/removal and state changes.

debug bfd zebra

Toggle zebra message events: show messages about interfaces, local addresses, VRF and daemon peer registrations.

1.5.2 BGP

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

Basic Concepts

Autonomous Systems

From **RFC 1930**:

> An AS is a connected group of one or more IP prefixes run by one or more network operators which has a SINGLE and CLEARLY DEFINED routing policy.

Each AS has an identifying number associated with it called an ASN (Autonomous System Number). This is a two octet value ranging in value from 1 to 65535. The AS numbers 64512 through 65535 are defined as private AS numbers. Private AS numbers must not be advertised on the global Internet.

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

See also:

RFC 1930

Address Families

Multiprotocol extensions enable BGP to carry routing information for multiple network layer protocols. BGP supports an Address Family Identifier (AFI) for IPv4 and IPv6. Support is also provided for multiple sets of per-AFI information via the BGP Subsequent Address Family Identifier (SAFI). FRR supports SAFIs for unicast information, labeled information (**RFC 3107** and **RFC 8277**), and Layer 3 VPN information (**RFC 4364** and **RFC 4659**).

Route Selection

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

1. **Weight check**

   Prefer higher local weight routes to lower routes.

2. **Local preference check**

   Prefer higher local preference routes to lower.

   If `bgp bestpath aigp` is enabled, and both paths that are compared have AIGP attribute, BGP uses AIGP tie-breaking unless both of the paths have the AIGP metric attribute. This means that the AIGP attribute is not evaluated during the best path selection process between two paths when one path does not have the AIGP attribute.

3. **Local route check**

   Prefer local routes (statics, aggregates, redistributed) to received routes.

4. **AS path length check**

   Prefer shortest hop-count AS_PATHs.

5. **Origin check**

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

6. **MED check**

   Where routes with a MED were received from the same AS, prefer the route with the lowest MED. *Multi-Exit Discriminator*.

7. **External check**

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

8. **IGP cost check**

   Prefer the route with the lower IGP cost.

9. **Multi-path check**

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

10. **Already-selected external check**

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

11. **Router-ID check**

    Prefer the route with the lowest *router-ID*. If the route has an *ORIGINATOR_ID* attribute, through iBGP reflection, then that router ID is used, otherwise the *router-ID* of the peer the route was received from is used.

12. **Cluster-List length check**

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

13. **Peer address**

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

Capability Negotiation

When adding IPv6 routing information exchange feature to BGP. There were some proposals. IETF (Internet Engineering Task Force) IDR (Inter Domain Routing) adopted a proposal called Multiprotocol Extension for BGP. The specification is described in **RFC 2283**. The protocol does not define new protocols. It defines new attributes to existing BGP. When it is used exchanging IPv6 routing information it is called BGP-4+. When it is used for exchanging multicast routing information it is called MBGP.

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

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

By default, FRR will bring up peering with minimal common capability for the both sides. For example, if the local router has unicast and multicast capabilities and the remote router only has unicast capability the local router will establish the connection with unicast only capability. When there are no common capabilities, FRR sends Unsupported Capability error and then resets the connection.

BGP Router Configuration

ASN and Router ID

First of all you must configure BGP router with the *router bgp ASN* command. The AS number is an identifier for the autonomous system. The AS identifier can either be a number or two numbers separated by a period. The BGP protocol uses the AS identifier for detecting whether the BGP connection is internal or external.

router bgp ASN

Enable a BGP protocol process with the specified ASN. After this statement you can input any *BGP Commands*.

bgp router-id A.B.C.D

This command specifies the router-ID. If *bgpd* connects to *zebra* it gets interface and address information. In that case default router ID value is selected as the largest IP Address of the interfaces. When *router zebra* is not enabled *bgpd* can't get interface information so *router-id* is set to 0.0.0.0. So please set router-id by hand.

Multiple Autonomous Systems

FRR's BGP implementation is capable of running multiple autonomous systems at once. Each configured AS corresponds to a *VRF*. In the past, to get the same functionality the network administrator had to run a new *bgpd* process; using VRFs allows multiple autonomous systems to be handled in a single process.

When using multiple autonomous systems, all router config blocks after the first one must specify a VRF to be the target of BGP's route selection. This VRF must be unique within respect to all other VRFs being used for the same purpose, i.e. two different autonomous systems cannot use the same VRF. However, the same AS can be used with different VRFs.

Note: The separated nature of VRFs makes it possible to peer a single *bgpd* process to itself, on one machine. Note that this can be done fully within BGP without a corresponding VRF in the kernel or Zebra, which enables some practical use cases such as *route reflectors* and route servers.

Configuration of additional autonomous systems, or of a router that targets a specific VRF, is accomplished with the following command:

router bgp ASN vrf VRFNAME

`VRFNAME` is matched against VRFs configured in the kernel. When `vrf VRFNAME` is not specified, the BGP protocol process belongs to the default VRF.

An example configuration with multiple autonomous systems might look like this:

```
router bgp 1
 neighbor 10.0.0.1 remote-as 20
 neighbor 10.0.0.2 remote-as 30
!
router bgp 2 vrf blue
 neighbor 10.0.0.3 remote-as 40
 neighbor 10.0.0.4 remote-as 50
!
router bgp 3 vrf red
 neighbor 10.0.0.5 remote-as 60
 neighbor 10.0.0.6 remote-as 70
...
```

See also:

VRF Route Leaking

See also:

VRF

Views

In addition to supporting multiple autonomous systems, FRR's BGP implementation also supports *views*.

BGP views are almost the same as normal BGP processes, except that routes selected by BGP are not installed into the kernel routing table. Each BGP view provides an independent set of routing information which is only distributed via BGP. Multiple views can be supported, and BGP view information is always independent from other routing protocols and Zebra/kernel routes. BGP views use the core instance (i.e., default VRF) for communication with peers.

router bgp AS-NUMBER view NAME

Make a new BGP view. You can use an arbitrary word for the NAME. Routes selected by the view are not installed into the kernel routing table.

With this command, you can setup Route Server like below.

```
!
router bgp 1 view 1
 neighbor 10.0.0.1 remote-as 2
 neighbor 10.0.0.2 remote-as 3
!
router bgp 2 view 2
 neighbor 10.0.0.3 remote-as 4
 neighbor 10.0.0.4 remote-as 5
```

show [ip] bgp view NAME

Display the routing table of BGP view NAME.

Route Selection

bgp bestpath as-path confed

This command specifies that the length of confederation path sets and sequences should should be taken into account during the BGP best path decision process.

bgp bestpath as-path multipath-relax

This command specifies that BGP decision process should consider paths of equal AS_PATH length candidates for multipath computation. Without the knob, the entire AS_PATH must match for multipath computation.

bgp bestpath compare-routerid

Ensure that when comparing routes where both are equal on most metrics, including local-pref, AS_PATH length, IGP cost, MED, that the tie is broken based on router-ID.

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

If a route has an *ORIGINATOR_ID* attribute because it has been reflected, that *ORIGINATOR_ID* will be used. Otherwise, the router-ID of the peer the route was received from will be used.

The advantage of this is that the route-selection (at this point) will be more deterministic. The disadvantage is that a few or even one lowest-ID router may attract all traffic to otherwise-equal paths because of this check. It

may increase the possibility of MED or IGP oscillation, unless other measures were taken to avoid these. The exact behaviour will be sensitive to the iBGP and reflection topology.

`bgp bestpath peer-type multipath-relax`

This command specifies that BGP decision process should consider paths from all peers for multipath computation. If this option is enabled, paths learned from any of eBGP, iBGP, or confederation neighbors will be multipath if they are otherwise considered equal cost.

`bgp bestpath aigp`

Use the bgp bestpath aigp command to evaluate the AIGP attribute during the best path selection process between two paths that have the AIGP attribute.

When bgp bestpath aigp is disabled, BGP does not use AIGP tie-breaking rules unless paths have the AIGP attribute.

Disabled by default.

`maximum-paths (1-128)`

Sets the maximum-paths value used for ecmp calculations for this bgp instance in EBGP. The maximum value listed, 128, can be limited by the ecmp cli for bgp or if the daemon was compiled with a lower ecmp value. This value can also be set in ipv4/ipv6 unicast/labeled unicast to only affect those particular afi/safi's.

`maximum-paths ibgp (1-128) [equal-cluster-length]`

Sets the maximum-paths value used for ecmp calculations for this bgp instance in IBGP. The maximum value listed, 128, can be limited by the ecmp cli for bgp or if the daemon was compiled with a lower ecmp value. This value can also be set in ipv4/ipv6 unicast/labeled unicast to only affect those particular afi/safi's.

Administrative Distance Metrics

`distance bgp (1-255) (1-255) (1-255)`

This command changes distance value of BGP. The arguments are the distance values for external routes, internal routes and local routes respectively.

`distance (1-255) A.B.C.D/M`

`distance (1-255) A.B.C.D/M WORD`

Sets the administrative distance for a particular route.

Require policy on EBGP

`bgp ebgp-requires-policy`

This command requires incoming and outgoing filters to be applied for eBGP sessions as part of RFC-8212 compliance. Without the incoming filter, no routes will be accepted. Without the outgoing filter, no routes will be announced.

This is enabled by default.

When you enable/disable this option you MUST clear the session.

When the incoming or outgoing filter is missing you will see "(Policy)" sign under `show bgp summary`:

```
exit1# show bgp summary

IPv4 Unicast Summary (VRF default):
BGP router identifier 10.10.10.1, local AS number 65001 vrf-id 0
```

(continues on next page)

(continued from previous page)

```
BGP table version 4
RIB entries 7, using 1344 bytes of memory
Peers 2, using 43 KiB of memory

Neighbor        V         AS   MsgRcvd   MsgSent   TblVer  InQ OutQ  Up/Down State/
↪PfxRcd   PfxSnt Desc
192.168.0.2     4      65002         8        10        0    0    0 00:03:09       ␣
↪    5 (Policy) N/A
fe80:1::2222    4      65002         9        11        0    0    0 00:03:09    ␣
↪(Policy) (Policy) N/A
```

Additionally a *show bgp neighbor* command would indicate in the *For address family:* block that:

```
exit1# show bgp neighbor
...
For address family: IPv4 Unicast
 Update group 1, subgroup 1
 Packet Queue length 0
 Inbound soft reconfiguration allowed
 Community attribute sent to this neighbor(all)
 Inbound updates discarded due to missing policy
 Outbound updates discarded due to missing policy
 0 accepted prefixes
```

Reject routes with AS_SET or AS_CONFED_SET types

bgp reject-as-sets

This command enables rejection of incoming and outgoing routes having AS_SET or AS_CONFED_SET type.

Suppress duplicate updates

bgp suppress-duplicates

For example, BGP routers can generate multiple identical announcements with empty community attributes if stripped at egress. This is an undesired behavior. Suppress duplicate updates if the route actually not changed. Default: enabled.

Send Hard Reset CEASE Notification for Administrative Reset

bgp hard-administrative-reset

Send Hard Reset CEASE Notification for 'Administrative Reset' events.

When disabled, and Graceful Restart Notification capability is exchanged between the peers, Graceful Restart procedures apply, and routes will be retained.

Enabled by default.

Disable checking if nexthop is connected on EBGP sessions

`bgp disable-ebgp-connected-route-check`

This command is used to disable the connection verification process for EBGP peering sessions that are reachable by a single hop but are configured on a loopback interface or otherwise configured with a non-directly connected IP address.

Route Flap Dampening

`bgp dampening (1-45) (1-20000) (1-50000) (1-255)`

This command enables BGP route-flap dampening and specifies dampening parameters.

half-life
Half-life time for the penalty

reuse-threshold
Value to start reusing a route

suppress-threshold
Value to start suppressing a route

max-suppress
Maximum duration to suppress a stable route

The route-flap damping algorithm is compatible with **RFC 2439**. The use of this command is not recommended nowadays.

At the moment, route-flap dampening is not working per VRF and is working only for IPv4 unicast and multicast.

See also:

https://www.ripe.net/publications/docs/ripe-378

Multi-Exit Discriminator

The BGP MED (Multi-Exit Discriminator) attribute has properties which can cause subtle convergence problems in BGP. These properties and problems have proven to be hard to understand, at least historically, and may still not be widely understood. The following attempts to collect together and present what is known about MED, to help operators and FRR users in designing and configuring their networks.

The BGP MED attribute is intended to allow one AS to indicate its preferences for its ingress points to another AS. The MED attribute will not be propagated on to another AS by the receiving AS - it is 'non-transitive' in the BGP sense.

E.g., if AS X and AS Y have 2 different BGP peering points, then AS X might set a MED of 100 on routes advertised at one and a MED of 200 at the other. When AS Y selects between otherwise equal routes to or via AS X, AS Y should prefer to take the path via the lower MED peering of 100 with AS X. Setting the MED allows an AS to influence the routing taken to it within another, neighbouring AS.

In this use of MED it is not really meaningful to compare the MED value on routes where the next AS on the paths differs. E.g., if AS Y also had a route for some destination via AS Z in addition to the routes from AS X, and AS Z had also set a MED, it wouldn't make sense for AS Y to compare AS Z's MED values to those of AS X. The MED values have been set by different administrators, with different frames of reference.

The default behaviour of BGP therefore is to not compare MED values across routes received from different neighbouring ASes. In FRR this is done by comparing the neighbouring, left-most AS in the received AS_PATHs of the routes and only comparing MED if those are the same.

Unfortunately, this behaviour of MED, of sometimes being compared across routes and sometimes not, depending on the properties of those other routes, means MED can cause the order of preference over all the routes to be undefined. That is, given routes A, B, and C, if A is preferred to B, and B is preferred to C, then a well-defined order should mean the preference is transitive (in the sense of orders[1]) and that A would be preferred to C.

However, when MED is involved this need not be the case. With MED it is possible that C is actually preferred over A. So A is preferred to B, B is preferred to C, but C is preferred to A. This can be true even where BGP defines a deterministic 'most preferred' route out of the full set of A,B,C. With MED, for any given set of routes there may be a deterministically preferred route, but there need not be any way to arrange them into any order of preference. With unmodified MED, the order of preference of routes literally becomes undefined.

That MED can induce non-transitive preferences over routes can cause issues. Firstly, it may be perceived to cause routing table churn locally at speakers; secondly, and more seriously, it may cause routing instability in iBGP topologies, where sets of speakers continually oscillate between different paths.

The first issue arises from how speakers often implement routing decisions. Though BGP defines a selection process that will deterministically select the same route as best at any given speaker, even with MED, that process requires evaluating all routes together. For performance and ease of implementation reasons, many implementations evaluate route preferences in a pair-wise fashion instead. Given there is no well-defined order when MED is involved, the best route that will be chosen becomes subject to implementation details, such as the order the routes are stored in. That may be (locally) non-deterministic, e.g.: it may be the order the routes were received in.

This indeterminism may be considered undesirable, though it need not cause problems. It may mean additional routing churn is perceived, as sometimes more updates may be produced than at other times in reaction to some event .

This first issue can be fixed with a more deterministic route selection that ensures routes are ordered by the neighbouring AS during selection. `bgp deterministic-med`. This may reduce the number of updates as routes are received, and may in some cases reduce routing churn. Though, it could equally deterministically produce the largest possible set of updates in response to the most common sequence of received updates.

A deterministic order of evaluation tends to imply an additional overhead of sorting over any set of n routes to a destination. The implementation of deterministic MED in FRR scales significantly worse than most sorting algorithms at present, with the number of paths to a given destination. That number is often low enough to not cause any issues, but where there are many paths, the deterministic comparison may quickly become increasingly expensive in terms of CPU.

Deterministic local evaluation can *not* fix the second, more major, issue of MED however. Which is that the non-transitive preference of routes MED can cause may lead to routing instability or oscillation across multiple speakers in iBGP topologies. This can occur with full-mesh iBGP, but is particularly problematic in non-full-mesh iBGP topologies that further reduce the routing information known to each speaker. This has primarily been documented with iBGP *route-reflection* topologies. However, any route-hiding technologies potentially could also exacerbate oscillation with MED.

This second issue occurs where speakers each have only a subset of routes, and there are cycles in the preferences between different combinations of routes - as the undefined order of preference of MED allows - and the routes are distributed in a way that causes the BGP speakers to 'chase' those cycles. This can occur even if all speakers use a deterministic order of evaluation in route selection.

E.g., speaker 4 in AS A might receive a route from speaker 2 in AS X, and from speaker 3 in AS Y; while speaker 5 in AS A might receive that route from speaker 1 in AS Y. AS Y might set a MED of 200 at speaker 1, and 100 at speaker 3. I.e, using ASN:ID:MED to label the speakers:

[1] For some set of objects to have an order, there *must* be some binary ordering relation that is defined for *every* combination of those objects, and that relation *must* be transitive. I.e.:, if the relation operator is <, and if a < b and b < c then that relation must carry over and it *must* be that a < c for the objects to have an order. The ordering relation may allow for equality, i.e. a < b and b < a may both be true and imply that a and b are equal in the order and not distinguished by it, in which case the set has a partial order. Otherwise, if there is an order, all the objects have a distinct place in the order and the set has a total order)

```
.
          /---------------\\
X:2------|--A:4-------A:5--|-Y:1:200
           Y:3:100--|-/   |
          \\---------------/
```

Assuming all other metrics are equal (AS_PATH, ORIGIN, 0 IGP costs), then based on the RFC4271 decision process speaker 4 will choose X:2 over Y:3:100, based on the lower ID of 2. Speaker 4 advertises X:2 to speaker 5. Speaker 5 will continue to prefer Y:1:200 based on the ID, and advertise this to speaker 4. Speaker 4 will now have the full set of routes, and the Y:1:200 it receives from 5 will beat X:2, but when speaker 4 compares Y:1:200 to Y:3:100 the MED check now becomes active as the ASes match, and now Y:3:100 is preferred. Speaker 4 therefore now advertises Y:3:100 to 5, which will also agrees that Y:3:100 is preferred to Y:1:200, and so withdraws the latter route from 4. Speaker 4 now has only X:2 and Y:3:100, and X:2 beats Y:3:100, and so speaker 4 implicitly updates its route to speaker 5 to X:2. Speaker 5 sees that Y:1:200 beats X:2 based on the ID, and advertises Y:1:200 to speaker 4, and the cycle continues.

The root cause is the lack of a clear order of preference caused by how MED sometimes is and sometimes is not compared, leading to this cycle in the preferences between the routes:

```
.
 /---> X:2 ---beats---> Y:3:100 --\\
|                                  |
|                                  |
 \\---beats--- Y:1:200 <---beats---/
```

This particular type of oscillation in full-mesh iBGP topologies can be avoided by speakers preferring already selected, external routes rather than choosing to update to new a route based on a post-MED metric (e.g. router-ID), at the cost of a non-deterministic selection process. FRR implements this, as do many other implementations, so long as it is not overridden by setting *`bgp bestpath compare-routerid`*, and see also *Route Selection*.

However, more complex and insidious cycles of oscillation are possible with iBGP route-reflection, which are not so easily avoided. These have been documented in various places. See, e.g.:

- [bgp-route-osci-cond]
- [stable-flexible-ibgp]
- [ibgp-correctness]

for concrete examples and further references.

There is as of this writing *no* known way to use MED for its original purpose; *and* reduce routing information in iBGP topologies; *and* be sure to avoid the instability problems of MED due the non-transitive routing preferences it can induce; in general on arbitrary networks.

There may be iBGP topology specific ways to reduce the instability risks, even while using MED, e.g.: by constraining the reflection topology and by tuning IGP costs between route-reflector clusters, see **RFC 3345** for details. In the near future, the Add-Path extension to BGP may also solve MED oscillation while still allowing MED to be used as intended, by distributing "best-paths per neighbour AS". This would be at the cost of distributing at least as many routes to all speakers as a full-mesh iBGP would, if not more, while also imposing similar CPU overheads as the "Deterministic MED" feature at each Add-Path reflector.

More generally, the instability problems that MED can introduce on more complex, non-full-mesh, iBGP topologies may be avoided either by:

- Setting *`bgp always-compare-med`*, however this allows MED to be compared across values set by different neighbour ASes, which may not produce coherent desirable results, of itself.

- Effectively ignoring MED by setting MED to the same value (e.g.: 0) using `set metric METRIC` on all received routes, in combination with setting *`bgp always-compare-med`* on all speakers. This is the simplest and most performant way to avoid MED oscillation issues, where an AS is happy not to allow neighbours to inject this problematic metric.

As MED is evaluated after the AS_PATH length check, another possible use for MED is for intra-AS steering of routes with equal AS_PATH length, as an extension of the last case above. As MED is evaluated before IGP metric, this can allow cold-potato routing to be implemented to send traffic to preferred hand-offs with neighbours, rather than the closest hand-off according to the IGP metric.

Note that even if action is taken to address the MED non-transitivity issues, other oscillations may still be possible. E.g., on IGP cost if iBGP and IGP topologies are at cross-purposes with each other - see the Flavel and Roughan paper above for an example. Hence the guideline that the iBGP topology should follow the IGP topology.

`bgp deterministic-med`

Carry out route-selection in way that produces deterministic answers locally, even in the face of MED and the lack of a well-defined order of preference it can induce on routes. Without this option the preferred route with MED may be determined largely by the order that routes were received in.

Setting this option will have a performance cost that may be noticeable when there are many routes for each destination. Currently in FRR it is implemented in a way that scales poorly as the number of routes per destination increases.

The default is that this option is not set.

Note that there are other sources of indeterminism in the route selection process, specifically, the preference for older and already selected routes from eBGP peers, *Route Selection*.

`bgp always-compare-med`

Always compare the MED on routes, even when they were received from different neighbouring ASes. Setting this option makes the order of preference of routes more defined, and should eliminate MED induced oscillations.

If using this option, it may also be desirable to use `set metric METRIC` to set MED to 0 on routes received from external neighbours.

This option can be used, together with `set metric METRIC` to use MED as an intra-AS metric to steer equal-length AS_PATH routes to, e.g., desired exit points.

Graceful Restart

BGP graceful restart functionality as defined in RFC-4724 defines the mechanisms that allows BGP speaker to continue to forward data packets along known routes while the routing protocol information is being restored.

Usually, when BGP on a router restarts, all the BGP peers detect that the session went down and then came up. This "down/up" transition results in a "routing flap" and causes BGP route re-computation, generation of BGP routing updates, and unnecessary churn to the forwarding tables.

The following functionality is provided by graceful restart:

1. The feature allows the restarting router to indicate to the helping peer the routes it can preserve in its forwarding plane during control plane restart by sending graceful restart capability in the OPEN message sent during session establishment.
2. The feature allows helping router to advertise to all other peers the routes received from the restarting router which are preserved in the forwarding plane of the restarting router during control plane restart.

```
(R1)----------------------------------------------------------------(R2)
```

(continues on next page)

(continued from previous page)

```
1. BGP Graceful Restart Capability exchanged between R1 & R2.

<------------------------------------------------------------------------>

2. Kill BGP Process at R1.

------------------------------------------------------------------------>

3. R2 Detects the above BGP Restart & verifies BGP Restarting
  Capability of R1.

4. Start BGP Process at R1.

5. Re-establish the BGP session between R1 & R2.

<------------------------------------------------------------------------>

6. R2 Send initial route updates, followed by End-Of-Rib.

<------------------------------------------------------------------------

7. R1 was waiting for End-Of-Rib from R2 & which has been received
  now.

8. R1 now runs BGP Best-Path algorithm. Send Initial BGP  Update,
  followed by End-Of Rib

<------------------------------------------------------------------------>
```

BGP-GR Preserve-Forwarding State

BGP OPEN message carrying optional capabilities for Graceful Restart has 8 bit "Flags for Address Family" for given AFI and SAFI. This field contains bit flags relating to routes that were advertised with the given AFI and SAFI.

```
0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
|F|   Reserved  |
+-+-+-+-+-+-+-+-+
```

The most significant bit is defined as the Forwarding State (F) bit, which can be used to indicate whether the forwarding state for routes that were advertised with the given AFI and SAFI has indeed been preserved during the previous BGP restart. When set (value 1), the bit indicates that the forwarding state has been preserved. The remaining bits are reserved and MUST be set to zero by the sender and ignored by the receiver.

bgp graceful-restart preserve-fw-state

FRR gives us the option to enable/disable the "F" flag using this specific vty command. However, it doesn't have the option to enable/disable this flag only for specific AFI/SAFI i.e. when this command is used, it applied to all the supported AFI/SAFI combinations for this peer.

End-of-RIB (EOR) message

An UPDATE message with no reachable Network Layer Reachability Information (NLRI) and empty withdrawn NLRI is specified as the End-of-RIB marker that can be used by a BGP speaker to indicate to its peer the completion of the initial routing update after the session is established.

For the IPv4 unicast address family, the End-of-RIB marker is an UPDATE message with the minimum length. For any other address family, it is an UPDATE message that contains only the MP_UNREACH_NLRI attribute with no withdrawn routes for that <AFI, SAFI>.

Although the End-of-RIB marker is specified for the purpose of BGP graceful restart, it is noted that the generation of such a marker upon completion of the initial update would be useful for routing convergence in general, and thus the practice is recommended.

Route Selection Deferral Timer

Specifies the time the restarting router defers the route selection process after restart.

Restarting Router : The usage of route election deferral timer is specified in https://tools.ietf.org/html/rfc4724#section-4.1

Once the session between the Restarting Speaker and the Receiving Speaker is re-established, the Restarting Speaker will receive and process BGP messages from its peers.

However, it MUST defer route selection for an address family until it either.

1. Receives the End-of-RIB marker from all its peers (excluding the ones with the "Restart State" bit set in the received capability and excluding the ones that do not advertise the graceful restart capability).
2. The Selection_Deferral_Timer timeout.

bgp graceful-restart select-defer-time (0-3600)

This is command, will set deferral time to value specified.

bgp graceful-restart rib-stale-time (1-3600)

This is command, will set the time for which stale routes are kept in RIB.

bgp graceful-restart restart-time (0-4095)

Set the time to wait to delete stale routes before a BGP open message is received.

Using with Long-lived Graceful Restart capability, this is recommended setting this timer to 0 and control stale routes with `bgp long-lived-graceful-restart stale-time`.

Default value is 120.

bgp graceful-restart stalepath-time (1-4095)

This is command, will set the max time (in seconds) to hold onto restarting peer's stale paths.

It also controls Enhanced Route-Refresh timer.

If this command is configured and the router does not receive a Route-Refresh EoRR message, the router removes the stale routes from the BGP table after the timer expires. The stale path timer is started when the router receives a Route-Refresh BoRR message.

bgp graceful-restart notification

Indicate Graceful Restart support for BGP NOTIFICATION messages.

After changing this parameter, you have to reset the peers in order to advertise N-bit in Graceful Restart capability.

Without Graceful-Restart Notification capability (N-bit not set), GR is not activated when receiving CEASE/HOLDTIME expire notifications.

When sending `CEASE/Administrative Reset` (`clear bgp`), the session is closed and routes are not retained. When N-bit is set and `bgp hard-administrative-reset` is turned off Graceful-Restart is activated and routes are retained.

Enabled by default.

BGP Per Peer Graceful Restart

Ability to enable and disable graceful restart, helper and no GR at all mode functionality at peer level.

So bgp graceful restart can be enabled at modes global BGP level or at per peer level. There are two FSM, one for BGP GR global mode and other for peer per GR.

Default global mode is helper and default peer per mode is inherit from global. If per peer mode is configured, the GR mode of this particular peer will override the global mode.

BGP GR Global Mode Commands

`bgp graceful-restart`
: This command will enable BGP graceful restart functionality at the global level.

`bgp graceful-restart disable`
: This command will disable both the functionality graceful restart and helper mode.

BGP GR Peer Mode Commands

`neighbor A.B.C.D graceful-restart`
: This command will enable BGP graceful restart functionality at the peer level.

`neighbor A.B.C.D graceful-restart-helper`
: This command will enable BGP graceful restart helper only functionality at the peer level.

`neighbor A.B.C.D graceful-restart-disable`
: This command will disable the entire BGP graceful restart functionality at the peer level.

Long-lived Graceful Restart

Currently, only restarter mode is supported. This capability is advertised only if graceful restart capability is negotiated.

`bgp long-lived-graceful-restart stale-time (1-16777215)`
: Specifies the maximum time to wait before purging long-lived stale routes for helper routers.

  Default is 0, which means the feature is off by default. Only graceful restart takes into account.

Administrative Shutdown

bgp shutdown [message MSG...]

Administrative shutdown of all peers of a bgp instance. Drop all BGP peers, but preserve their configurations. The peers are notified in accordance with RFC 8203 by sending a `NOTIFICATION` message with error code `Cease` and subcode `Administrative Shutdown` prior to terminating connections. This global shutdown is independent of the neighbor shutdown, meaning that individually shut down peers will not be affected by lifting it.

An optional shutdown message *MSG* can be specified.

Networks

network A.B.C.D/M

This command adds the announcement network.

```
router bgp 1
 address-family ipv4 unicast
  network 10.0.0.0/8
 exit-address-family
```

This configuration example says that network 10.0.0.0/8 will be announced to all neighbors. Some vendors' routers don't advertise routes if they aren't present in their IGP routing tables; *bgpd* doesn't care about IGP routes when announcing its routes.

bgp network import-check

This configuration modifies the behavior of the network statement. If you have this configured the underlying network must exist in the rib. If you have the [no] form configured then BGP will not check for the networks existence in the rib. default is the network must exist.

IPv6 Support

neighbor A.B.C.D activate

This configuration modifies whether to enable an address family for a specific neighbor. By default only the IPv4 unicast address family is enabled.

```
router bgp 1
 address-family ipv6 unicast
  neighbor 2001:0DB8::1 activate
  network 2001:0DB8:5009::/64
 exit-address-family
```

This configuration example says that network 2001:0DB8:5009::/64 will be announced and enables the neighbor 2001:0DB8::1 to receive this announcement.

By default, only the IPv4 unicast address family is announced to all neighbors. Using the 'no bgp default ipv4-unicast' configuration overrides this default so that all address families need to be enabled explicitly.

```
router bgp 1
 no bgp default ipv4-unicast
 neighbor 10.10.10.1 remote-as 2
 neighbor 2001:0DB8::1 remote-as 3
```

(continues on next page)

(continued from previous page)

```
address-family ipv4 unicast
 neighbor 10.10.10.1 activate
 network 192.168.1.0/24
exit-address-family
address-family ipv6 unicast
 neighbor 2001:0DB8::1 activate
 network 2001:0DB8:5009::/64
exit-address-family
```

This configuration demonstrates how the 'no bgp default ipv4-unicast' might be used in a setup with two upstreams where each of the upstreams should only receive either IPv4 or IPv6 announcements.

Using the `bgp default ipv6-unicast` configuration, IPv6 unicast address family is enabled by default for all new neighbors.

Route Aggregation

Route Aggregation-IPv4 Address Family

aggregate-address A.B.C.D/M

This command specifies an aggregate address.

In order to advertise an aggregated prefix, a more specific (longer) prefix MUST exist in the BGP table. For example, if you want to create an `aggregate-address 10.0.0.0/24`, you should make sure you have something like `10.0.0.5/32` or `10.0.0.0/26`, or any other smaller prefix in the BGP table. The routing information table (RIB) is not enough, you have to redistribute them into the BGP table.

aggregate-address A.B.C.D/M route-map NAME

Apply a route-map for an aggregated prefix.

aggregate-address A.B.C.D/M origin <egp|igp|incomplete>

Override ORIGIN for an aggregated prefix.

aggregate-address A.B.C.D/M as-set

This command specifies an aggregate address. Resulting routes include AS set.

aggregate-address A.B.C.D/M summary-only

This command specifies an aggregate address.

Longer prefixes advertisements of more specific routes to all neighbors are suppressed.

aggregate-address A.B.C.D/M matching-MED-only

Configure the aggregated address to only be created when the routes MED match, otherwise no aggregated route will be created.

aggregate-address A.B.C.D/M suppress-map NAME

Similar to *summary-only*, but will only suppress more specific routes that are matched by the selected route-map.

This configuration example sets up an `aggregate-address` under the ipv4 address-family.

```
router bgp 1
 address-family ipv4 unicast
  aggregate-address 10.0.0.0/8
  aggregate-address 20.0.0.0/8 as-set
```

(continues on next page)

(continued from previous page)

```
 aggregate-address 40.0.0.0/8 summary-only
 aggregate-address 50.0.0.0/8 route-map aggr-rmap
exit-address-family
```

Route Aggregation-IPv6 Address Family

aggregate-address X:X::X:X/M

This command specifies an aggregate address.

aggregate-address X:X::X:X/M route-map NAME

Apply a route-map for an aggregated prefix.

aggregate-address X:X::X:X/M origin <egp|igp|incomplete>

Override ORIGIN for an aggregated prefix.

aggregate-address X:X::X:X/M as-set

This command specifies an aggregate address. Resulting routes include AS set.

aggregate-address X:X::X:X/M summary-only

This command specifies an aggregate address.

Longer prefixes advertisements of more specific routes to all neighbors are suppressed

aggregate-address X:X::X:X/M matching-MED-only

Configure the aggregated address to only be created when the routes MED match, otherwise no aggregated route will be created.

aggregate-address X:X::X:X/M suppress-map NAME

Similar to *summary-only*, but will only suppress more specific routes that are matched by the selected route-map.

This configuration example sets up an `aggregate-address` under the ipv6 address-family.

```
router bgp 1
 address-family ipv6 unicast
  aggregate-address 10::0/64
  aggregate-address 20::0/64 as-set
  aggregate-address 40::0/64 summary-only
  aggregate-address 50::0/64 route-map aggr-rmap
 exit-address-family
```

Redistribution

Redistribution configuration should be placed under the `address-family` section for the specific AF to redistribute into. Protocol availability for redistribution is determined by BGP AF; for example, you cannot redistribute OSPFv3 into `address-family ipv4 unicast` as OSPFv3 supports IPv6.

**redistribute <connected | isis | kernel | ospf | ospf6 | rip | ripng | **
static|table>[metric (0-4294967295)] [route-map WORD]

Redistribute routes from other protocols into BGP.

bgp update-delay MAX-DELAY ESTABLISH-WAIT

This feature is used to enable read-only mode on BGP process restart or when a BGP process is cleared using 'clear ip bgp *'. Note that this command is configured at the global level and applies to all bgp instances/vrfs. It cannot be used at the same time as the "update-delay" command described below, which is entered in each bgp instance/vrf desired to delay update installation and advertisements. The global and per-vrf approaches to defining update-delay are mutually exclusive.

When applicable, read-only mode would begin as soon as the first peer reaches Established status and a timer for max-delay seconds is started. During this mode BGP doesn't run any best-path or generate any updates to its peers. This mode continues until:

1. All the configured peers, except the shutdown peers, have sent explicit EOR (End-Of-RIB) or an implicit-EOR. The first keep-alive after BGP has reached Established is considered an implicit-EOR. If the establish-wait optional value is given, then BGP will wait for peers to reach established from the beginning of the update-delay till the establish-wait period is over, i.e. the minimum set of established peers for which EOR is expected would be peers established during the establish-wait window, not necessarily all the configured neighbors.
2. max-delay period is over.

On hitting any of the above two conditions, BGP resumes the decision process and generates updates to its peers.

Default max-delay is 0, i.e. the feature is off by default.

update-delay MAX-DELAY

update-delay MAX-DELAY ESTABLISH-WAIT

This feature is used to enable read-only mode on BGP process restart or when a BGP process is cleared using 'clear ip bgp *'. Note that this command is configured under the specific bgp instance/vrf that the feature is enabled for. It cannot be used at the same time as the global "bgp update-delay" described above, which is entered at the global level and applies to all bgp instances. The global and per-vrf approaches to defining update-delay are mutually exclusive.

When applicable, read-only mode would begin as soon as the first peer reaches Established status and a timer for max-delay seconds is started. During this mode BGP doesn't run any best-path or generate any updates to its peers. This mode continues until:

1. All the configured peers, except the shutdown peers, have sent explicit EOR (End-Of-RIB) or an implicit-EOR. The first keep-alive after BGP has reached Established is considered an implicit-EOR. If the establish-wait optional value is given, then BGP will wait for peers to reach established from the beginning of the update-delay till the establish-wait period is over, i.e. the minimum set of established peers for which EOR is expected would be peers established during the establish-wait window, not necessarily all the configured neighbors.
2. max-delay period is over.

On hitting any of the above two conditions, BGP resumes the decision process and generates updates to its peers.

Default max-delay is 0, i.e. the feature is off by default.

table-map ROUTE-MAP-NAME

This feature is used to apply a route-map on route updates from BGP to Zebra. All the applicable match operations are allowed, such as match on prefix, next-hop, communities, etc. Set operations for this attach-point are limited to metric and next-hop only. Any operation of this feature does not affect BGPs internal RIB.

Supported for ipv4 and ipv6 address families. It works on multi-paths as well, however, metric setting is based on the best-path only.

Peers

Defining Peers

neighbor PEER remote-as ASN

Creates a new neighbor whose remote-as is ASN. PEER can be an IPv4 address or an IPv6 address or an interface to use for the connection.

```
router bgp 1
 neighbor 10.0.0.1 remote-as 2
```

In this case my router, in AS-1, is trying to peer with AS-2 at 10.0.0.1.

This command must be the first command used when configuring a neighbor. If the remote-as is not specified, *bgpd* will complain like this:

```
can't find neighbor 10.0.0.1
```

neighbor PEER remote-as internal

Create a peer as you would when you specify an ASN, except that if the peers ASN is different than mine as specified under the *router bgp ASN* command the connection will be denied.

neighbor PEER remote-as external

Create a peer as you would when you specify an ASN, except that if the peers ASN is the same as mine as specified under the *router bgp ASN* command the connection will be denied.

bgp listen range <A.B.C.D/M|X:X::X:X/M> peer-group PGNAME

Accept connections from any peers in the specified prefix. Configuration from the specified peer-group is used to configure these peers.

Note: When using BGP listen ranges, if the associated peer group has TCP MD5 authentication configured, your kernel must support this on prefixes. On Linux, this support was added in kernel version 4.14. If your kernel does not support this feature you will get a warning in the log file, and the listen range will only accept connections from peers without MD5 configured.

Additionally, we have observed that when using this option at scale (several hundred peers) the kernel may hit its option memory limit. In this situation you will see error messages like:

```
bgpd: sockopt_tcp_signature: setsockopt(23): Cannot allocate memory
```

In this case you need to increase the value of the sysctl `net.core.optmem_max` to allow the kernel to allocate the necessary option memory.

---

bgp listen limit <1-65535>

Define the maximum number of peers accepted for one BGP instance. This limit is set to 100 by default. Increasing this value will really be possible if more file descriptors are available in the BGP process.

coalesce-time (0-4294967295)

The time in milliseconds that BGP will delay before deciding what peers can be put into an update-group together in order to generate a single update for them. The default time is 1000.

Configuring Peers

neighbor PEER shutdown [message MSG...] [rtt (1-65535) [count (1-255)]]

Shutdown the peer. We can delete the neighbor's configuration by `no neighbor PEER remote-as ASN` but all configuration of the neighbor will be deleted. When you want to preserve the configuration, but want to drop the BGP peer, use this syntax.

Optionally you can specify a shutdown message *MSG*.

Also, you can specify optionally `rtt` in milliseconds to automatically shutdown the peer if round-trip-time becomes higher than defined.

Additional `count` parameter is the number of keepalive messages to count before shutdown the peer if round-trip-time becomes higher than defined.

neighbor PEER disable-connected-check

Allow peerings between directly connected eBGP peers using loopback addresses.

neighbor PEER disable-link-bw-encoding- ieee

By default bandwidth in extended communities is carried encoded as IEEE floating-point format, which is according to the draft.

Older versions have the implementation where extended community bandwidth value is carried encoded as uint32. To enable backward compatibility we need to disable IEEE floating-point encoding option per-peer.

neighbor PEER extended-optional-parameters

Force Extended Optional Parameters Length format to be used for OPEN messages.

By default, it's disabled. If the standard optional parameters length is higher than one-octet (255), then extended format is enabled automatically.

For testing purposes, extended format can be enabled with this command.

neighbor PEER ebgp-multihop

Specifying `ebgp-multihop` allows sessions with eBGP neighbors to establish when they are multiple hops away. When the neighbor is not directly connected and this knob is not enabled, the session will not establish.

If the peer's IP address is not in the RIB and is reachable via the default route, then you have to enable `ip nht resolve-via-default`.

neighbor PEER description ...

Set description of the peer.

neighbor PEER interface IFNAME

When you connect to a BGP peer over an IPv6 link-local address, you have to specify the IFNAME of the interface used for the connection. To specify IPv4 session addresses, see the `neighbor PEER update-source` command below.

neighbor PEER interface remote-as <internal|external|ASN>

Configure an unnumbered BGP peer. `PEER` should be an interface name. The session will be established via IPv6 link locals. Use `internal` for iBGP and `external` for eBGP sessions, or specify an ASN if you wish.

neighbor PEER next-hop-self [force]

This command specifies an announced route's nexthop as being equivalent to the address of the bgp router if it is learned via eBGP. This will also bypass third-party next-hops in favor of the local bgp address. If the optional keyword `force` is specified the modification is done also for routes learned via iBGP.

neighbor PEER attribute-unchanged [{as-path|next-hop|med}]

This command specifies attributes to be left unchanged for advertisements sent to a peer. Use this to leave the next-hop unchanged in ipv6 configurations, as the route-map directive to leave the next-hop unchanged is only available for ipv4.

neighbor PEER update-source <IFNAME|ADDRESS>

Specify the IPv4 source address to use for the BGP session to this neighbour, may be specified as either an IPv4 address directly or as an interface name (in which case the *zebra* daemon MUST be running in order for *bgpd* to be able to retrieve interface state).

```
router bgp 64555
 neighbor foo update-source 192.168.0.1
 neighbor bar update-source loopback0
```

neighbor PEER default-originate [route-map WORD]

bgpd's default is to not announce the default route (0.0.0.0/0) even if it is in routing table. When you want to announce default routes to the peer, use this command.

If `route-map` keyword is specified, then the default route will be originated only if route-map conditions are met. For example, announce the default route only if `10.10.10.10/32` route exists and set an arbitrary community for a default route.

```
router bgp 64555
 address-family ipv4 unicast
  neighbor 192.168.255.1 default-originate route-map default
!
ip prefix-list p1 seq 5 permit 10.10.10.10/32
!
route-map default permit 10
 match ip address prefix-list p1
 set community 123:123
!
```

neighbor PEER port PORT

neighbor PEER password PASSWORD

Set a MD5 password to be used with the tcp socket that is being used to connect to the remote peer. Please note if you are using this command with a large number of peers on linux you should consider modifying the *net.core.optmem_max* sysctl to a larger value to avoid out of memory errors from the linux kernel.

neighbor PEER send-community

neighbor PEER weight WEIGHT

This command specifies a default *weight* value for the neighbor's routes.

neighbor PEER maximum-prefix NUMBER [force]

Sets a maximum number of prefixes we can receive from a given peer. If this number is exceeded, the BGP session will be destroyed.

In practice, it is generally preferable to use a prefix-list to limit what prefixes are received from the peer instead of using this knob. Tearing down the BGP session when a limit is exceeded is far more destructive than merely rejecting undesired prefixes. The prefix-list method is also much more granular and offers much smarter matching criterion than number of received prefixes, making it more suited to implementing policy.

If `force` is set, then ALL prefixes are counted for maximum instead of accepted only. This is useful for cases where an inbound filter is applied, but you want maximum-prefix to act on ALL (including filtered) prefixes. This option requires *soft-reconfiguration inbound* to be enabled for the peer.

neighbor PEER maximum-prefix-out NUMBER

Sets a maximum number of prefixes we can send to a given peer.

Since sent prefix count is managed by update-groups, this option creates a separate update-group for outgoing updates.

neighbor PEER local-as AS-NUMBER [no-prepend] [replace-as]

Specify an alternate AS for this BGP process when interacting with the specified peer. With no modifiers, the specified local-as is prepended to the received AS_PATH when receiving routing updates from the peer, and prepended to the outgoing AS_PATH (after the process local AS) when transmitting local routes to the peer.

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

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

Note that replace-as can only be specified if no-prepend is.

This command is only allowed for eBGP peers.

neighbor <A.B.C.D|X:X::X:X|WORD> as-override

Override AS number of the originating router with the local AS number.

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

This command is only allowed for eBGP peers.

neighbor <A.B.C.D|X:X::X:X|WORD> allowas-in [<(1-10)|origin>]

Accept incoming routes with AS path containing AS number with the same value as the current system AS.

This is used when you want to use the same AS number in your sites, but you can't connect them directly. This is an alternative to *neighbor WORD as-override*.

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

The parameter *origin* configures BGP to only accept routes originated with the same AS number as the system.

This command is only allowed for eBGP peers.

neighbor <A.B.C.D|X:X::X:X|WORD> addpath-tx-all-paths

Configure BGP to send all known paths to neighbor in order to preserve multi path capabilities inside a network.

neighbor <A.B.C.D|X:X::X:X|WORD> addpath-tx-bestpath-per-AS

Configure BGP to send best known paths to neighbor in order to preserve multi path capabilities inside a network.

neighbor <A.B.C.D|X:X::X:X|WORD> disable-addpath-rx

Do not accept additional paths from this neighbor.

neighbor PEER ttl-security hops NUMBER

This command enforces Generalized TTL Security Mechanism (GTSM), as specified in RFC 5082. With this command, only neighbors that are the specified number of hops away will be allowed to become neighbors. This command is mutually exclusive with *ebgp-multihop*.

neighbor PEER capability extended-nexthop

Allow bgp to negotiate the extended-nexthop capability with it's peer. If you are peering over a v6 LL address then this capability is turned on automatically. If you are peering over a v6 Global Address then turning on this command will allow BGP to install v4 routes with v6 nexthops if you do not have v4 configured on interfaces.

neighbor <A.B.C.D|X:X::X:X|WORD> accept-own

Enable handling of self-originated VPN routes containing `accept-own` community.

This feature allows you to handle self-originated VPN routes, which a BGP speaker receives from a route-reflector. A 'self-originated' route is one that was originally advertised by the speaker itself. As per **RFC 4271**, a BGP speaker rejects advertisements that originated the speaker itself. However, the BGP ACCEPT_OWN mechanism enables a router to accept the prefixes it has advertised, when reflected from a route-reflector that modifies certain attributes of the prefix.

A special community called `accept-own` is attached to the prefix by the route-reflector, which is a signal to the receiving router to bypass the ORIGINATOR_ID and NEXTHOP/MP_REACH_NLRI check.

Default: disabled.

neighbor <A.B.C.D|X:X::X:X|WORD> path-attribute discard (1-255)...

Drops specified path attributes from BGP UPDATE messages from the specified neighbor.

If you do not want specific attributes, you can drop them using this command, and let the BGP proceed by ignoring those attributes.

neighbor <A.B.C.D|X:X::X:X|WORD> path-attribute treat-as-withdraw (1-255)...

Received BGP UPDATES that contain specified path attributes are treat-as-withdraw. If there is an existing prefix in the BGP routing table, it will be removed.

neighbor <A.B.C.D|X:X::X:X|WORD> graceful-shutdown

Mark all routes from this neighbor as less preferred by setting `graceful-shutdown` community, and local-preference to 0.

bgp fast-external-failover

This command causes bgp to take down ebgp peers immediately when a link flaps. *bgp fast-external-failover* is the default and will not be displayed as part of a *show run*. The no form of the command turns off this ability.

bgp default ipv4-unicast

This command allows the user to specify that the IPv4 Unicast address family is turned on by default or not. This command defaults to on and is not displayed. The *no bgp default ipv4-unicast* form of the command is displayed.

bgp default ipv4-vpn

This command allows the user to specify that the IPv4 MPLS VPN address family is turned on by default or not. This command defaults to off and is not displayed. The *bgp default ipv4-vpn* form of the command is displayed.

bgp default ipv6-unicast

This command allows the user to specify that the IPv6 Unicast address family is turned on by default or not. This command defaults to off and is not displayed. The *bgp default ipv6-unicast* form of the command is displayed.

bgp default ipv6-vpn

This command allows the user to specify that the IPv6 MPLS VPN address family is turned on by default or not. This command defaults to off and is not displayed. The *bgp default ipv6-vpn* form of the command is displayed.

bgp default show-hostname

This command shows the hostname of the peer in certain BGP commands outputs. It's easier to troubleshoot if you have a number of BGP peers.

bgp default show-nexthop-hostname

This command shows the hostname of the next-hop in certain BGP commands outputs. It's easier to troubleshoot if you have a number of BGP peers and a number of routes to check.

neighbor PEER advertisement-interval (0-600)

Setup the minimum route advertisement interval(mrai) for the peer in question. This number is between 0 and 600 seconds, with the default advertisement interval being 0.

neighbor PEER timers (0-65535) (0-65535)

Set keepalive and hold timers for a neighbor. The first value is keepalive and the second is hold time.

neighbor PEER timers connect (1-65535)

Set connect timer for a neighbor. The connect timer controls how long BGP waits between connection attempts to a neighbor.

neighbor PEER timers delayopen (1-240)

This command allows the user enable the *RFC 4271 <https://tools.ietf.org/html/rfc4271/>* DelayOpenTimer with the specified interval or disable it with the negating command for the peer. By default, the DelayOpenTimer is disabled. The timer interval may be set to a duration of 1 to 240 seconds.

bgp minimum-holdtime (1-65535)

This command allows user to prevent session establishment with BGP peers with lower holdtime less than configured minimum holdtime. When this command is not set, minimum holdtime does not work.

bgp tcp-keepalive (1-65535) (1-65535) (1-30)

This command allows user to configure TCP keepalive with new BGP peers. Each parameter respectively stands for TCP keepalive idle timer (seconds), interval (seconds), and maximum probes. By default, TCP keepalive is disabled.

Displaying Information about Peers

show bgp <afi> <safi> neighbors WORD bestpath-routes [detail] [json] [wide]

For the given neighbor, WORD, that is specified list the routes selected by BGP as having the best path.

If `detail` option is specified, the detailed version of all routes will be displayed. The same format as `show [ip] bgp [afi] [safi] PREFIX` will be used, but for the whole table of received, advertised or filtered prefixes.

If `json` option is specified, output is displayed in JSON format.

If `wide` option is specified, then the prefix table's width is increased to fully display the prefix and the nexthop.

Peer Filtering

neighbor PEER distribute-list NAME [in|out]

This command specifies a distribute-list for the peer. *direct* is `in` or `out`.

neighbor PEER prefix-list NAME [in|out]

neighbor PEER filter-list NAME [in|out]

neighbor PEER route-map NAME [in|out]

Apply a route-map on the neighbor. *direct* must be *in* or *out*.

bgp route-reflector allow-outbound-policy

By default, attribute modification via route-map policy out is not reflected on reflected routes. This option allows the modifications to be reflected as well. Once enabled, it affects all reflected routes.

neighbor PEER sender-as-path-loop-detection

Enable the detection of sender side AS path loops and filter the bad routes before they are sent.

This setting is disabled by default.

Peer Groups

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

neighbor WORD peer-group

This command defines a new peer group.

neighbor PEER peer-group PGNAME

This command bind specific peer to peer group WORD.

neighbor PEER solo

This command is used to indicate that routes advertised by the peer should not be reflected back to the peer. This command only is only meaningful when there is a single peer defined in the peer-group.

show [ip] bgp peer-group [json]

This command displays configured BGP peer-groups.

```
soodar# show bgp peer-group

BGP peer-group test1, remote AS 65001
Peer-group type is external
Configured address-families: IPv4 Unicast; IPv6 Unicast;
1 IPv4 listen range(s)
   192.168.100.0/24
2 IPv6 listen range(s)
   2001:db8:1::/64
   2001:db8:2::/64
Peer-group members:
   192.168.200.1  Active
   2001:db8::1  Active

BGP peer-group test2
Peer-group type is external
Configured address-families: IPv4 Unicast;
```

Optional `json` parameter is used to display JSON output.

```
{
  "test1":{
    "remoteAs":65001,
    "type":"external",
    "addressFamiliesConfigured":[
      "IPv4 Unicast",
      "IPv6 Unicast"
    ],
    "dynamicRanges":{
      "IPv4":{
        "count":1,
        "ranges":[
          "192.168.100.0\/24"
        ]
```

(continues on next page)

(continued from previous page)


```
      },
      "IPv6":{
        "count":2,
        "ranges":[
          "2001:db8:1::\/64",
          "2001:db8:2::\/64"
        ]
      }
    },
    "members":{
      "192.168.200.1":{
        "status":"Active"
      },
      "2001:db8::1":{
        "status":"Active"
      }
    }
  },
  "test2":{
     "type":"external",
     "addressFamiliesConfigured":[
       "IPv4 Unicast"
     ]
  }
}
```


Capability Negotiation

neighbor PEER strict-capability-match

Strictly compares remote capabilities and local capabilities. If capabilities are different, send Unsupported Capability error then reset connection.

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

neighbor PEER dont-capability-negotiate

Suppress sending Capability Negotiation as OPEN message optional parameter to the peer. This command only affects the peer is configured other than IPv4 unicast configuration.

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

You may prefer locally configured capabilities more than the negotiated capabilities even though remote peer sends capabilities. If the peer is configured by *override-capability*, *bgpd* ignores received capabilities then override negotiated capabilities with configured values.

Additionally the operator should be reminded that this feature fundamentally disables the ability to use widely deployed BGP features. BGP unnumbered, hostname support, AS4, Addpath, Route Refresh, ORF, Dynamic Capabilities, and graceful restart.

neighbor PEER override-capability

Override the result of Capability Negotiation with local configuration. Ignore remote peer's capability value.

neighbor PEER capability software-version

Send the software version in the BGP OPEN message to the neighbor. This is very useful in environments with a large amount of peers with different versions of FRR or any other vendor.

Disabled by default.

neighbor PEER aigp

Send and receive AIGP attribute for this neighbor. This is valid only for eBGP neighbors.

Disabled by default. iBGP neighbors have this option enabled implicitly.

AS Path Access Lists

AS path access list is user defined AS path.

bgp as-path access-list WORD [seq (0-4294967295)] permit|deny LINE

This command defines a new AS path access list.

show bgp as-path-access-list [json]

Display all BGP AS Path access lists.

If the `json` option is specified, output is displayed in JSON format.

show bgp as-path-access-list WORD [json]

Display the specified BGP AS Path access list.

If the `json` option is specified, output is displayed in JSON format.

Bogon ASN filter policy configuration example

```
bgp as-path access-list 99 permit _0_
bgp as-path access-list 99 permit _23456_
bgp as-path access-list 99 permit _1310[0-6][0-9]_|_13107[0-1]_
bgp as-path access-list 99 seq 20 permit ^65
```

Using AS Path in Route Map

match as-path WORD

For a given as-path, WORD, match it on the BGP as-path given for the prefix and if it matches do normal route-map actions. The no form of the command removes this match from the route-map.

set as-path prepend AS-PATH

Prepend the given string of AS numbers to the AS_PATH of the BGP path's NLRI. The no form of this command removes this set operation from the route-map.

set as-path prepend last-as NUM

Prepend the existing last AS number (the leftmost ASN) to the AS_PATH. The no form of this command removes this set operation from the route-map.

set as-path replace <any|ASN>

Replace a specific AS number to local AS number. `any` replaces each AS number in the AS-PATH with the local AS number.

Communities Attribute

The BGP communities attribute is widely used for implementing policy routing. Network operators can manipulate BGP communities attribute based on their network policy. BGP communities attribute is defined in RFC 1997 and RFC 1998. It is an optional transitive attribute, therefore local policy can travel through different autonomous system.

The communities attribute is a set of communities values. Each community value is 4 octet long. The following format is used to define the community value.

AS:VAL
: This format represents 4 octet communities value. `AS` is high order 2 octet in digit format. `VAL` is low order 2 octet in digit format. This format is useful to define AS oriented policy value. For example, `7675:80` can be used when AS 7675 wants to pass local policy value 80 to neighboring peer.

graceful-shutdown
: `graceful-shutdown` represents well-known communities value `GRACEFUL_SHUTDOWN` `0xFFFF0000` `65535:0`. RFC 8326 implements the purpose Graceful BGP Session Shutdown to reduce the amount of lost traffic when taking BGP sessions down for maintenance. The use of the community needs to be supported from your peers side to actually have any effect.

accept-own
: `accept-own` represents well-known communities value `ACCEPT_OWN` `0xFFFF0001` `65535:1`. RFC 7611 implements a way to signal to a router to accept routes with a local nexthop address. This can be the case when doing policing and having traffic having a nexthop located in another VRF but still local interface to the router. It is recommended to read the RFC for full details.

route-filter-translated-v4
: `route-filter-translated-v4` represents well-known communities value `ROUTE_FILTER_TRANSLATED_v4` `0xFFFF0002` `65535:2`.

route-filter-v4
: `route-filter-v4` represents well-known communities value `ROUTE_FILTER_v4` `0xFFFF0003` `65535:3`.

route-filter-translated-v6
: `route-filter-translated-v6` represents well-known communities value `ROUTE_FILTER_TRANSLATED_v6` `0xFFFF0004` `65535:4`.

route-filter-v6
: `route-filter-v6` represents well-known communities value `ROUTE_FILTER_v6` `0xFFFF0005` `65535:5`.

llgr-stale
: `llgr-stale` represents well-known communities value `LLGR_STALE` `0xFFFF0006` `65535:6`. Assigned and intended only for use with routers supporting the Long-lived Graceful Restart Capability as described in [Draft-IETF-uttaro-idr-bgp-persistence]. Routers receiving routes with this community may (depending on implementation) choose allow to reject or modify routes on the presence or absence of this community.

no-llgr
: `no-llgr` represents well-known communities value `NO_LLGR` `0xFFFF0007` `65535:7`. Assigned and intended only for use with routers supporting the Long-lived Graceful Restart Capability as described in [Draft-IETF-uttaro-idr-bgp-persistence]. Routers receiving routes with this community may (depending on implementation) choose allow to reject or modify routes on the presence or absence of this community.

accept-own-nexthop
: `accept-own-nexthop` represents well-known communities value `accept-own-nexthop` `0xFFFF0008` `65535:8`. [Draft-IETF-agrewal-idr-accept-own-nexthop] describes how to tag and label VPN routes to be able to send traffic between VRFs via an internal layer 2 domain on the same PE device. Refer to [Draft-IETF-agrewal-idr-accept-own-nexthop] for full details.

blackhole
: `blackhole` represents well-known communities value `BLACKHOLE` `0xFFFF029A` `65535:666`. RFC 7999 doc-

uments sending prefixes to EBGP peers and upstream for the purpose of blackholing traffic. Prefixes tagged with the this community should normally not be re-advertised from neighbors of the originating network. Upon receiving `BLACKHOLE` community from a BGP speaker, `NO_ADVERTISE` community is added automatically.

no-export
`no-export` represents well-known communities value `NO_EXPORT` `0xFFFFFF01`. All routes carry this value must not be advertised to outside a BGP confederation boundary. If neighboring BGP peer is part of BGP confederation, the peer is considered as inside a BGP confederation boundary, so the route will be announced to the peer.

no-advertise
`no-advertise` represents well-known communities value `NO_ADVERTISE` `0xFFFFFF02`. All routes carry this value must not be advertise to other BGP peers.

local-AS
`local-AS` represents well-known communities value `NO_EXPORT_SUBCONFED` `0xFFFFFF03`. All routes carry this value must not be advertised to external BGP peers. Even if the neighboring router is part of confederation, it is considered as external BGP peer, so the route will not be announced to the peer.

no-peer
`no-peer` represents well-known communities value `NOPEER` `0xFFFFFF04` `65535:65284`. **RFC 3765** is used to communicate to another network how the originating network want the prefix propagated.

When the communities attribute is received duplicate community values in the attribute are ignored and value is sorted in numerical order.

Community Lists

Community lists are user defined lists of community attribute values. These lists can be used for matching or manipulating the communities attribute in UPDATE messages.

There are two types of community list:

standard
This type accepts an explicit value for the attribute.

expanded
This type accepts a regular expression. Because the regex must be interpreted on each use expanded community lists are slower than standard lists.

bgp community-list standard NAME permit|deny COMMUNITY

This command defines a new standard community list. `COMMUNITY` is communities value. The `COMMUNITY` is compiled into community structure. We can define multiple community list under same name. In that case match will happen user defined order. Once the community list matches to communities attribute in BGP updates it return permit or deny by the community list definition. When there is no matched entry, deny will be returned. When `COMMUNITY` is empty it matches to any routes.

bgp community-list expanded NAME permit|deny COMMUNITY

This command defines a new expanded community list. `COMMUNITY` is a string expression of communities attribute. `COMMUNITY` can be a regular expression (*BGP Regular Expressions*) to match the communities attribute in BGP updates. The expanded community is only used to filter, not *set* actions.

Deprecated since version 5.0: It is recommended to use the more explicit versions of this command.

bgp community-list NAME permit|deny COMMUNITY

When the community list type is not specified, the community list type is automatically detected. If `COMMUNITY`

can be compiled into communities attribute, the community list is defined as a standard community list. Otherwise it is defined as an expanded community list. This feature is left for backward compatibility. Use of this feature is not recommended.

Note that all community lists share the same namespace, so it's not necessary to specify `standard` or `expanded`; these modifiers are purely aesthetic.

show bgp community-list [NAME detail]

Displays community list information. When `NAME` is specified the specified community list's information is shown.

```
# show bgp community-list
Named Community standard list CLIST
permit 7675:80 7675:100 no-export
deny internet
  Named Community expanded list EXPAND
permit :

  # show bgp community-list CLIST detail
  Named Community standard list CLIST
permit 7675:80 7675:100 no-export
deny internet
```

Numbered Community Lists

When number is used for BGP community list name, the number has special meanings. Community list number in the range from 1 to 99 is standard community list. Community list number in the range from 100 to 500 is expanded community list. These community lists are called as numbered community lists. On the other hand normal community lists is called as named community lists.

bgp community-list (1-99) permit|deny COMMUNITY

This command defines a new community list. The argument to (1-99) defines the list identifier.

bgp community-list (100-500) permit|deny COMMUNITY

This command defines a new expanded community list. The argument to (100-500) defines the list identifier.

Community alias

BGP community aliases are useful to quickly identify what communities are set for a specific prefix in a human-readable format. Especially handy for a huge amount of communities. Accurately defined aliases can help you faster spot things on the wire.

bgp community alias NAME ALIAS

This command creates an alias name for a community that will be used later in various CLI outputs in a human-readable format.

```
soodar# show run | include bgp community alias
bgp community alias 65001:14 community-1
bgp community alias 65001:123:1 lcommunity-1

soodar# show ip bgp 172.16.16.1/32
BGP routing table entry for 172.16.16.1/32, version 21
Paths: (2 available, best #2, table default)
```

(continues on next page)

(continued from previous page)

```
Advertised to non peer-group peers:
65030
  192.168.0.2 from 192.168.0.2 (172.16.16.1)
    Origin incomplete, metric 0, valid, external, best (Neighbor IP)
    Community: 65001:12 65001:13 community-1 65001:65534
    Large Community: lcommunity-1 65001:123:2
    Last update: Fri Apr 16 12:51:27 2021
```

show bgp [afi] [safi] [all] alias WORD [wide|json]

Display prefixes with matching BGP community alias.

Using Communities in Route Maps

In *Route Maps* we can match on or set the BGP communities attribute. Using this feature network operator can implement their network policy based on BGP communities attribute.

The following commands can be used in route maps:

match alias WORD

This command performs match to BGP updates using community alias WORD. When the one of BGP communities value match to the one of community alias value in community alias, it is match.

match community WORD exact-match [exact-match]

This command perform match to BGP updates using community list WORD. When the one of BGP communities value match to the one of communities value in community list, it is match. When *exact-match* keyword is specified, match happen only when BGP updates have completely same communities value specified in the community list.

set community <none|COMMUNITY> additive

This command sets the community value in BGP updates. If the attribute is already configured, the newly provided value replaces the old one unless the `additive` keyword is specified, in which case the new value is appended to the existing value.

If `none` is specified as the community value, the communities attribute is not sent.

It is not possible to set an expanded community list.

set comm-list WORD delete

This command remove communities value from BGP communities attribute. The `word` is community list name. When BGP route's communities value matches to the community list `word`, the communities value is removed. When all of communities value is removed eventually, the BGP update's communities attribute is completely removed.

Example Configuration

The following configuration is exemplary of the most typical usage of BGP communities attribute. In the example, AS 7675 provides an upstream Internet connection to AS 100. When the following configuration exists in AS 7675, the network operator of AS 100 can set local preference in AS 7675 network by setting BGP communities attribute to the updates.

```
router bgp 7675
 neighbor 192.168.0.1 remote-as 100
```

(continues on next page)

(continued from previous page)

```
 address-family ipv4 unicast
  neighbor 192.168.0.1 route-map RMAP in
 exit-address-family
!
bgp community-list 70 permit 7675:70
bgp community-list 80 permit 7675:80
bgp community-list 90 permit 7675:90
!
route-map RMAP permit 10
 match community 70
 set local-preference 70
!
route-map RMAP permit 20
 match community 80
 set local-preference 80
!
route-map RMAP permit 30
 match community 90
 set local-preference 90
```

The following configuration announces `10.0.0.0/8` from AS 100 to AS 7675. The route has communities value `7675:80` so when above configuration exists in AS 7675, the announced routes' local preference value will be set to 80.

```
router bgp 100
 network 10.0.0.0/8
 neighbor 192.168.0.2 remote-as 7675
 address-family ipv4 unicast
  neighbor 192.168.0.2 route-map RMAP out
 exit-address-family
!
ip prefix-list PLIST permit 10.0.0.0/8
!
route-map RMAP permit 10
 match ip address prefix-list PLIST
 set community 7675:80
```

The following configuration is an example of BGP route filtering using communities attribute. This configuration only permit BGP routes which has BGP communities value (`0:80` and `0:90`) or `0:100`. The network operator can set special internal communities value at BGP border router, then limit the BGP route announcements into the internal network.

```
router bgp 7675
 neighbor 192.168.0.1 remote-as 100
 address-family ipv4 unicast
  neighbor 192.168.0.1 route-map RMAP in
 exit-address-family
!
bgp community-list 1 permit 0:80 0:90
bgp community-list 1 permit 0:100
!
route-map RMAP permit in
 match community 1
```

The following example filters BGP routes which have a community value of `1:1`. When there is no match community-list returns `deny`. To avoid filtering all routes, a `permit` line is set at the end of the community-list.

```
router bgp 7675
 neighbor 192.168.0.1 remote-as 100
 address-family ipv4 unicast
  neighbor 192.168.0.1 route-map RMAP in
 exit-address-family
!
bgp community-list standard FILTER deny 1:1
bgp community-list standard FILTER permit
!
route-map RMAP permit 10
 match community FILTER
```

The following configuration is an example of communities value deletion. With this configuration the community values `100:1` and `100:2` are removed from BGP updates. For communities value deletion, only `permit` community-list is used. `deny` community-list is ignored.

```
router bgp 7675
 neighbor 192.168.0.1 remote-as 100
 address-family ipv4 unicast
  neighbor 192.168.0.1 route-map RMAP in
 exit-address-family
!
bgp community-list standard DEL permit 100:1 100:2
!
route-map RMAP permit 10
 set comm-list DEL delete
```

Extended Communities Attribute

BGP extended communities attribute is introduced with MPLS VPN/BGP technology. MPLS VPN/BGP expands capability of network infrastructure to provide VPN functionality. At the same time it requires a new framework for policy routing. With BGP Extended Communities Attribute we can use Route Target or Site of Origin for implementing network policy for MPLS VPN/BGP.

BGP Extended Communities Attribute is similar to BGP Communities Attribute. It is an optional transitive attribute. BGP Extended Communities Attribute can carry multiple Extended Community value. Each Extended Community value is eight octet length.

BGP Extended Communities Attribute provides an extended range compared with BGP Communities Attribute. Adding to that there is a type field in each value to provides community space structure.

There are two format to define Extended Community value. One is AS based format the other is IP address based format.

`AS:VAL`
: This is a format to define AS based Extended Community value. `AS` part is 2 octets Global Administrator subfield in Extended Community value. `VAL` part is 4 octets Local Administrator subfield. `7675:100` represents AS 7675 policy value 100.

`IP-Address:VAL`
: This is a format to define IP address based Extended Community value. `IP-Address` part is 4 octets Global Administrator subfield. `VAL` part is 2 octets Local Administrator subfield.

Extended Community Lists

bgp extcommunity-list standard NAME permit|deny EXTCOMMUNITY

This command defines a new standard extcommunity-list. *extcommunity* is extended communities value. The *extcommunity* is compiled into extended community structure. We can define multiple extcommunity-list under same name. In that case match will happen user defined order. Once the extcommunity-list matches to extended communities attribute in BGP updates it return permit or deny based upon the extcommunity-list definition. When there is no matched entry, deny will be returned. When *extcommunity* is empty it matches to any routes.

bgp extcommunity-list expanded NAME permit|deny LINE

This command defines a new expanded extcommunity-list. *line* is a string expression of extended communities attribute. *line* can be a regular expression (*BGP Regular Expressions*) to match an extended communities attribute in BGP updates.

Note that all extended community lists shares a single name space, so it's not necessary to specify their type when creating or destroying them.

show bgp extcommunity-list [NAME detail]

This command displays current extcommunity-list information. When *name* is specified the community list's information is shown.

BGP Extended Communities in Route Map

match extcommunity WORD

set extcommunity none

This command resets the extended community value in BGP updates. If the attribute is already configured or received from the peer, the attribute is discarded and set to none. This is useful if you need to strip incoming extended communities.

set extcommunity rt EXTCOMMUNITY

This command sets Route Target value.

set extcommunity nt EXTCOMMUNITY

This command sets Node Target value.

If the receiving BGP router supports Node Target Extended Communities, it will install the route with the community that contains it's own local BGP Identifier. Otherwise, it's not installed.

set extcommunity soo EXTCOMMUNITY

This command sets Site of Origin value.

set extcommunity bandwidth <(1-25600) | cumulative | num-multipaths> [non-transitive]

This command sets the BGP link-bandwidth extended community for the prefix (best path) for which it is applied. The link-bandwidth can be specified as an `explicit value` (specified in Mbps), or the router can be told to use the `cumulative bandwidth` of all multipaths for the prefix or to compute it based on the `number of multipaths`. The link bandwidth extended community is encoded as `transitive` unless the set command explicitly configures it as `non-transitive`.

See also:

wecmp_linkbw

Note that the extended expanded community is only used for *match* rule, not for *set* actions.

Large Communities Attribute

The BGP Large Communities attribute was introduced in Feb 2017 with **RFC 8092**.

The BGP Large Communities Attribute is similar to the BGP Communities Attribute except that it has 3 components instead of two and each of which are 4 octets in length. Large Communities bring additional functionality and convenience over traditional communities, specifically the fact that the `GLOBAL` part below is now 4 octets wide allowing seamless use in networks using 4-byte ASNs.

`GLOBAL:LOCAL1:LOCAL2`

This is the format to define Large Community values. Referencing **RFC 8195** the values are commonly referred to as follows:

- The `GLOBAL` part is a 4 octet Global Administrator field, commonly used as the operators AS number.
- The `LOCAL1` part is a 4 octet Local Data Part 1 subfield referred to as a function.
- The `LOCAL2` part is a 4 octet Local Data Part 2 field and referred to as the parameter subfield.

As an example, `65551:1:10` represents AS 65551 function 1 and parameter 10. The referenced RFC above gives some guidelines on recommended usage.

Large Community Lists

Two types of large community lists are supported, namely *standard* and *expanded*.

`bgp large-community-list standard NAME permit|deny LARGE-COMMUNITY`

This command defines a new standard large-community-list. *large-community* is the Large Community value. We can add multiple large communities under same name. In that case the match will happen in the user defined order. Once the large-community-list matches the Large Communities attribute in BGP updates it will return permit or deny based upon the large-community-list definition. When there is no matched entry, a deny will be returned. When *large-community* is empty it matches any routes.

`bgp large-community-list expanded NAME permit|deny LINE`

This command defines a new expanded large-community-list. Where *line* is a string matching expression, it will be compared to the entire Large Communities attribute as a string, with each large-community in order from lowest to highest. *line* can also be a regular expression which matches this Large Community attribute.

Note that all community lists share the same namespace, so it's not necessary to specify `standard` or `expanded`; these modifiers are purely aesthetic.

`show bgp large-community-list`

`show bgp large-community-list NAME detail`

This command display current large-community-list information. When *name* is specified the community list information is shown.

`show ip bgp large-community-info`

This command displays the current large communities in use.

Large Communities in Route Map

`match large-community LINE [exact-match]`

Where *line* can be a simple string to match, or a regular expression. It is very important to note that this match occurs on the entire large-community string as a whole, where each large-community is ordered from lowest to highest. When *exact-match* keyword is specified, match happen only when BGP updates have completely same large communities value specified in the large community list.

`set large-community LARGE-COMMUNITY`

`set large-community LARGE-COMMUNITY LARGE-COMMUNITY`

`set large-community LARGE-COMMUNITY additive`

These commands are used for setting large-community values. The first command will overwrite any large-communities currently present. The second specifies two large-communities, which overwrites the current large-community list. The third will add a large-community value without overwriting other values. Multiple large-community values can be specified.

Note that the large expanded community is only used for *match* rule, not for *set* actions.

BGP Roles and Only to Customers

BGP roles are defined in **RFC 9234** and provide an easy way to route leaks prevention, detection and mitigation.

To enable its mechanics, you must set your local role to reflect your type of peering relationship with your neighbor. Possible values of `LOCAL-ROLE` are:

- provider
- rs-server
- rs-client
- customer
- peer

The local Role value is negotiated with the new BGP Role capability with a built-in check of the corresponding value. In case of mismatch the new OPEN Roles Mismatch Notification <2, 11> would be sent.

The correct Role pairs are:

- Provider - Customer
- Peer - Peer
- RS-Server - RS-Client

```
soodar# show bgp neighbor | include Role
  Local Role: customer
  Neighbor Role: provider
    Role: advertised and received
```

If strict-mode is set BGP session won't become established until BGP neighbor set local Role on its side. This configuration parameter is defined in **RFC 9234** and used to enforce corresponding configuration at your counter-part side. Default value - disabled.

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

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

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

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

neighbor PEER local-role LOCAL-ROLE [strict-mode]

This command set your local-role to LOCAL-ROLE: <provider|rs-server|rs-client|customer|peer>.

This role helps to detect and prevent route leaks.

If strict-mode is set, your neighbor must send you Capability with the value of his role (by setting local-role on his side). Otherwise, a Role Mismatch Notification will be sent.

Labeled unicast

bgpd supports labeled information, as per **RFC 3107**.

By default, locally advertised prefixes use the *implicit-null* label to encode in the outgoing NLRI. The following command uses the *explicit-null* label value for all the BGP instances.

L3VPN VRFs

bgpd supports L3VPN (Layer 3 Virtual Private Networks) VRFs (Virtual Routing and Forwarding) for IPv4 **RFC 4364** and IPv6 **RFC 4659**. L3VPN routes, and their associated VRF MPLS labels, can be distributed to VPN SAFI neighbors in the *default*, i.e., non VRF, BGP instance. VRF MPLS labels are reached using *core* MPLS labels which are distributed using LDP or BGP labeled unicast. *bgpd* also supports inter-VRF route leaking.

VRF Route Leaking

BGP routes may be leaked (i.e. copied) between a unicast VRF RIB and the VPN SAFI RIB of the default VRF for use in MPLS-based L3VPNs. Unicast routes may also be leaked between any VRFs (including the unicast RIB of the default BGP instanced). A shortcut syntax is also available for specifying leaking from one VRF to another VRF using the default instance's VPN RIB as the intermediary. A common application of the VRF-VRF feature is to connect a customer's private routing domain to a provider's VPN service. Leaking is configured from the point of view of an individual VRF: `import` refers to routes leaked from VPN to a unicast VRF, whereas `export` refers to routes leaked from a unicast VRF to VPN.

Required parameters

Routes exported from a unicast VRF to the VPN RIB must be augmented by two parameters:

- an RD (Route Distinguisher)
- an RTLIST (Route-target List)

Configuration for these exported routes must, at a minimum, specify these two parameters.

Routes imported from the VPN RIB to a unicast VRF are selected according to their RTLISTs. Routes whose RTLIST contains at least one route-target in common with the configured import RTLIST are leaked. Configuration for these imported routes must specify an RTLIST to be matched.

The RD, which carries no semantic value, is intended to make the route unique in the VPN RIB among all routes of its prefix that originate from all the customers and sites that are attached to the provider's VPN service. Accordingly, each site of each customer is typically assigned an RD that is unique across the entire provider network.

The RTLIST is a set of route-target extended community values whose purpose is to specify route-leaking policy. Typically, a customer is assigned a single route-target value for import and export to be used at all customer sites. This configuration specifies a simple topology wherein a customer has a single routing domain which is shared across all its sites. More complex routing topologies are possible through use of additional route-targets to augment the leaking of sets of routes in various ways.

When using the shortcut syntax for vrf-to-vrf leaking, the RD and RT are auto-derived.

General configuration

Configuration of route leaking between a unicast VRF RIB and the VPN SAFI RIB of the default VRF is accomplished via commands in the context of a VRF address-family:

rd vpn export AS:NN|IP:nn

Specifies the route distinguisher to be added to a route exported from the current unicast VRF to VPN.

rt vpn import|export|both RTLIST...

Specifies the route-target list to be attached to a route (export) or the route-target list to match against (import) when exporting/importing between the current unicast VRF and VPN.

The RTLIST is a space-separated list of route-targets, which are BGP extended community values as described in *Extended Communities Attribute*.

label vpn export allocation-mode per-vrf|per-nexthop

Select how labels are allocated in the given VRF. By default, the *per-vrf* mode is selected, and one label is used for all prefixes from the VRF. The *per-nexthop* will use a unique label for all prefixes that are reachable via the same nexthop.

label vpn export (0..1048575)|auto

Enables an MPLS label to be attached to a route exported from the current unicast VRF to VPN. If the value specified is `auto`, the label value is automatically assigned from a pool maintained by the Zebra daemon. If Zebra is not running, or if this command is not configured, automatic label assignment will not complete, which will block corresponding route export.

nexthop vpn export A.B.C.D|X:X::X:X

Specifies an optional nexthop value to be assigned to a route exported from the current unicast VRF to VPN. If left unspecified, the nexthop will be set to 0.0.0.0 or 0:0::0:0 (self).

route-map vpn import|export MAP

Specifies an optional route-map to be applied to routes imported or exported between the current unicast VRF and VPN.

import|export vpn

Enables import or export of routes between the current unicast VRF and VPN.

import vrf VRFNAME

Shortcut syntax for specifying automatic leaking from vrf VRFNAME to the current VRF using the VPN RIB as intermediary. The RD and RT are auto derived and should not be specified explicitly for either the source or destination VRF's.

This shortcut syntax mode is not compatible with the explicit *import vpn* and *export vpn* statements for the two VRF's involved. The CLI will disallow attempts to configure incompatible leaking modes.

bgp retain route-target all

It is possible to retain or not VPN prefixes that are not imported by local VRF configuration. This can be done via the following command in the context of the global VPNv4/VPNv6 family. This command defaults to on and is not displayed. The *no bgp retain route-target all* form of the command is displayed.

neighbor <A.B.C.D|X:X::X:X|WORD> soo EXTCOMMUNITY

Without this command, SoO extended community attribute is configured using an inbound route map that sets the SoO value during the update process. With the introduction of the new BGP per-neighbor Site-of-Origin (SoO) feature, two new commands configured in sub-modes under router configuration mode simplify the SoO value configuration.

If we configure SoO per neighbor at PEs, the SoO community is automatically added for all routes from the CPEs. Routes are validated and prevented from being sent back to the same CPE (e.g.: multi-site). This is especially needed when using `as-override` or `allowas-in` to prevent routing loops.

mpls bgp forwarding

It is possible to permit BGP install VPN prefixes without transport labels, by issuing the following command under the interface configuration context. This configuration will install VPN prefixes originated from an e-bgp session, and with the next-hop directly connected.

Debugging

show debug

Show all enabled debugs.

show bgp listeners

Display Listen sockets and the vrf that created them. Useful for debugging of when listen is not working and this is considered a developer debug statement.

debug bgp allow-martian

Enable or disable BGP accepting martian nexthops from a peer. Please note this is not an actual debug command and this command is also being deprecated and will be removed soon. The new command is `bgp allow-martian-nexthop`

debug bgp bfd

Enable or disable debugging for BFD events. This will show BFD integration library messages and BGP BFD integration messages that are mostly state transitions and validation problems.

debug bgp conditional-advertisement

Enable or disable debugging of BGP conditional advertisement.

debug bgp neighbor-events

Enable or disable debugging for neighbor events. This provides general information on BGP events such as peer connection / disconnection, session establishment / teardown, and capability negotiation.

debug bgp updates

Enable or disable debugging for BGP updates. This provides information on BGP UPDATE messages transmitted and received between local and remote instances.

debug bgp keepalives

Enable or disable debugging for BGP keepalives. This provides information on BGP KEEPALIVE messages transmitted and received between local and remote instances.

debug bgp bestpath <A.B.C.D/M|X:X::X:X/M>

Enable or disable debugging for bestpath selection on the specified prefix.

`debug bgp nht`
: Enable or disable debugging of BGP nexthop tracking.

`debug bgp update-groups`
: Enable or disable debugging of dynamic update groups. This provides general information on group creation, deletion, join and prune events.

`debug bgp zebra`
: Enable or disable debugging of communications between *bgpd* and *zebra*.

Other BGP Commands

The following are available in the top level *enable* mode:

`clear bgp \*`
: Clear all peers.

`clear bgp ipv4|ipv6 \*`
: Clear all peers with this address-family activated.

`clear bgp ipv4|ipv6 unicast \*`
: Clear all peers with this address-family and sub-address-family activated.

`clear bgp ipv4|ipv6 PEER`
: Clear peers with address of X.X.X.X and this address-family activated.

`clear bgp ipv4|ipv6 unicast PEER`
: Clear peer with address of X.X.X.X and this address-family and sub-address-family activated.

`clear bgp ipv4|ipv6 PEER soft|in|out`
: Clear peer using soft reconfiguration in this address-family.

`clear bgp ipv4|ipv6 unicast PEER soft|in|out`
: Clear peer using soft reconfiguration in this address-family and sub-address-family.

`clear bgp [ipv4|ipv6] [unicast] PEER|\* message-stats`
: Clear BGP message statistics for a specified peer or for all peers, optionally filtered by activated address-family and sub-address-family.

The following are available in the `router bgp` mode:

`write-quanta (1-64)`
: BGP message Tx I/O is vectored. This means that multiple packets are written to the peer socket at the same time each I/O cycle, in order to minimize system call overhead. This value controls how many are written at a time. Under certain load conditions, reducing this value could make peer traffic less 'bursty'. In practice, leave this settings on the default (64) unless you truly know what you are doing.

`read-quanta (1-10)`
: Unlike Tx, BGP Rx traffic is not vectored. Packets are read off the wire one at a time in a loop. This setting controls how many iterations the loop runs for. As with write-quanta, it is best to leave this setting on the default.

The following command is available in `config` mode as well as in the `router bgp` mode:

`bgp graceful-shutdown`
: The purpose of this command is to initiate BGP Graceful Shutdown which is described in **RFC 8326**. The use case for this is to minimize or eliminate the amount of traffic loss in a network when a planned maintenance activity such as software upgrade or hardware replacement is to be performed on a router. The feature works by

re-announcing routes to eBGP peers with the GRACEFUL_SHUTDOWN community included. Peers are then expected to treat such paths with the lowest preference. This happens automatically on a receiver running FRR; with other routing protocol stacks, an inbound policy may have to be configured. In FRR, triggering graceful shutdown also results in announcing a LOCAL_PREF of 0 to iBGP peers.

Graceful shutdown can be configured per BGP instance or globally for all of BGP. These two options are mutually exclusive. The no form of the command causes graceful shutdown to be stopped, and routes will be re-announced without the GRACEFUL_SHUTDOWN community and/or with the usual LOCAL_PREF value. Note that if this option is saved to the startup configuration, graceful shutdown will remain in effect across restarts of *bgpd* and will need to be explicitly disabled.

bgp input-queue-limit (1-4294967295)

Set the BGP Input Queue limit for all peers when messaging parsing. Increase this only if you have the memory to handle large queues of messages at once.

bgp output-queue-limit (1-4294967295)

Set the BGP Output Queue limit for all peers when messaging parsing. Increase this only if you have the memory to handle large queues of messages at once.

Displaying BGP Information

The following four commands display the IPv6 and IPv4 routing tables, depending on whether or not the `ip` keyword is used. Actually, `show ip bgp` command was used on older *Quagga* routing daemon project, while `show bgp` command is the new format. The choice has been done to keep old format with IPv4 routing table, while new format displays IPv6 routing table.

show ip bgp [all] [wide|json [detail]]

show ip bgp A.B.C.D [json]

show bgp [all] [wide|json [detail]]

show bgp X:X::X:X [json]

These commands display BGP routes. When no route is specified, the default is to display all BGP routes.

```
BGP table version is 0, local router ID is 10.1.1.1
   Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
   Origin codes: i - IGP, e - EGP, ? - incomplete

Network    Next Hop      Metric LocPrf Weight Path
   \*> 1.1.1.1/32       0.0.0.0      0   32768 i

   Total number of prefixes 1
```

If `wide` option is specified, then the prefix table's width is increased to fully display the prefix and the nexthop.

This is especially handy dealing with IPv6 prefixes and if `[no] bgp default show-nexthop-hostname` is enabled.

If `all` option is specified, `ip` keyword is ignored, show bgp all and show ip bgp all commands display routes for all AFIs and SAFIs.

If `json` option is specified, output is displayed in JSON format.

If `detail` option is specified after `json`, more verbose JSON output will be displayed.

Some other commands provide additional options for filtering the output.

show [ip] bgp regexp LINE

This command displays BGP routes using AS path regular expression (*BGP Regular Expressions*).

show [ip] bgp [all] summary [wide] [json]

Show a bgp peer summary for the specified address family.

The old command structure `show ip bgp` may be removed in the future and should no longer be used. In order to reach the other BGP routing tables other than the IPv6 routing table given by `show bgp`, the new command structure is extended with `show bgp [afi] [safi]`.

`wide` option gives more output like `LocalAS` and extended `Desc` to 64 characters.

```
soodar# show ip bgp summary wide

IPv4 Unicast Summary (VRF default):
BGP router identifier 192.168.100.1, local AS number 65534 vrf-id 0
BGP table version 3
RIB entries 5, using 920 bytes of memory
Peers 1, using 27 KiB of memory

Neighbor        V         AS    LocalAS   MsgRcvd   MsgSent   TblVer  InQ OutQ␣
↪ Up/Down State/PfxRcd   PfxSnt Desc
192.168.0.2     4      65030        123        15        22        0    0    0␣
↪00:07:00            0        1 us-east1-rs1.frrouting.org

Total number of neighbors 1
soodar#
```

If PfxRcd and/or PfxSnt is shown as `(Policy)`, that means that the EBGP default policy is turned on, but you don't have any filters applied for incoming/outgoing directions.

See also:

Require policy on EBGP

show bgp vrfs [<VRFNAME$vrf_name>] [json]

The command displays all bgp vrf instances basic info like router-id, configured and established neighbors, evpn related basic info like l3vni, router-mac, vxlan-interface. User can get that information as JSON format when `json` keyword at the end of cli is presented.

```
soodar# show bgp vrfs
Type  Id     routerId          #PeersCfg  #PeersEstb  Name
             L3-VNI            RouterMAC              Interface
DFLT  0      17.0.0.6          3          3           default
             0                 00:00:00:00:00:00      unknown
 VRF  21     17.0.0.6          0          0           sym_1
             8888              34:11:12:22:22:01      vlan4034_l3
 VRF  32     17.0.0.6          0          0           sym_2
             8889              34:11:12:22:22:01      vlan4035_l3

Total number of VRFs (including default): 3
```

show bgp [afi] [safi] [all] [wide|json]

show bgp [<ipv4|ipv6> <unicast|vpn|labeled-unicast>]

These commands display BGP routes for the specific routing table indicated by the selected afi and the selected safi. If no afi and no safi value is given, the command falls back to the default IPv6 routing table

show bgp [afi] [safi] [all] summary [json]

Show a bgp peer summary for the specified address family, and subsequent address-family.

show bgp [afi] [safi] [all] summary failed [json]

Show a bgp peer summary for peers that are not successfully exchanging routes for the specified address family, and subsequent address-family.

show bgp [afi] [safi] [all] summary established [json]

Show a bgp peer summary for peers that are successfully exchanging routes for the specified address family, and subsequent address-family.

show bgp [afi] [safi] [all] summary neighbor [PEER] [json]

Show a bgp summary for the specified peer, address family, and subsequent address-family. The neighbor filter can be used in combination with the failed, established filters.

show bgp [afi] [safi] [all] summary remote-as <internal|external|ASN> [json]

Show a bgp peer summary for the specified remote-as ASN or type (`internal` for iBGP and `external` for eBGP sessions), address family, and subsequent address-family. The remote-as filter can be used in combination with the failed, established filters.

show bgp [afi] [safi] [all] summary terse [json]

Shorten the output. Do not show the following information about the BGP instances: the number of RIB entries, the table version and the used memory. The `terse` option can be used in combination with the remote-as, neighbor, failed and established filters, and with the `wide` option as well.

show bgp [afi] [safi] [neighbor [PEER] [routes|advertised-routes | received-routes] \ [<A.B.C.D/M|X:X::X:X/M> | detail] [json]

This command shows information on a specific BGP peer of the relevant afi and safi selected.

The `routes` keyword displays only routes in this address-family's BGP table that were received by this peer and accepted by inbound policy.

The `advertised-routes` keyword displays only the routes in this address-family's BGP table that were permitted by outbound policy and advertised to to this peer.

The `received-routes` keyword displays all routes belonging to this address-family (prior to inbound policy) that were received by this peer.

If a specific prefix is specified, the detailed version of that prefix will be displayed.

If `detail` option is specified, the detailed version of all routes will be displayed. The same format as `show [ip] bgp [afi] [safi] PREFIX` will be used, but for the whole table of received, advertised or filtered prefixes.

If `json` option is specified, output is displayed in JSON format.

show bgp [<view|vrf> VIEWVRFNAME] [afi] [safi] neighbors PEER received prefix-filter \ [json]

Display Address Prefix ORFs received from this peer.

show bgp [afi] [safi] [all] dampening dampened-paths [wide|json]

Display paths suppressed due to dampening of the selected afi and safi selected.

show bgp [afi] [safi] [all] dampening flap-statistics [wide|json]

Display flap statistics of routes of the selected afi and safi selected.

show bgp [afi] [safi] [all] dampening parameters [json]

Display details of configured dampening parameters of the selected afi and safi.

If the `json` option is specified, output is displayed in JSON format.

show bgp [afi] [safi] [all] version (1-4294967295) [wide|json]

Display prefixes with matching version numbers. The version number and above having prefixes will be listed here.

It helps to identify which prefixes were installed at some point.

show bgp [afi] [safi] statistics

Display statistics of routes of the selected afi and safi.

show bgp statistics-all

Display statistics of routes of all the afi and safi.

show [ip] bgp [afi] [safi] [all] cidr-only [wide|json]

Display routes with non-natural netmasks.

show [ip] bgp [afi] [safi] [all] prefix-list WORD [wide|json]

Display routes that match the specified prefix-list.

If `wide` option is specified, then the prefix table's width is increased to fully display the prefix and the nexthop.

If the `json` option is specified, output is displayed in JSON format.

show [ip] bgp [afi] [safi] [all] access-list WORD [wide|json]

Display routes that match the specified access-list.

show [ip] bgp [afi] [safi] [all] filter-list WORD [wide|json]

Display routes that match the specified AS-Path filter-list.

If `wide` option is specified, then the prefix table's width is increased to fully display the prefix and the nexthop.

If the `json` option is specified, output is displayed in JSON format.

show [ip] bgp [afi] [safi] [all] route-map WORD [wide|json]

Display routes that match the specified route-map.

If `wide` option is specified, then the prefix table's width is increased to fully display the prefix and the nexthop.

If the `json` option is specified, output is displayed in JSON format.

show [ip] bgp [afi] [safi] [all] <A.B.C.D/M|X:X::X:X/M> longer-prefixes [wide|json]

Displays the specified route and all more specific routes.

If `wide` option is specified, then the prefix table's width is increased to fully display the prefix and the nexthop.

If the `json` option is specified, output is displayed in JSON format.

show [ip] bgp [afi] [safi] [all] self-originate [wide|json]

Display self-originated routes.

If `wide` option is specified, then the prefix table's width is increased to fully display the prefix and the nexthop.

If the `json` option is specified, output is displayed in JSON format.

show [ip] bgp [afi] [safi] [all] neighbors A.B.C.D [advertised-routes | received-routes|filtered-routes] \ [<A.B.C.D/M|X:X::X:X/M> | detail] [json|wide]

Display the routes advertised to a BGP neighbor or received routes from neighbor or filtered routes received from neighbor based on the option specified.

If `wide` option is specified, then the prefix table's width is increased to fully display the prefix and the nexthop.

This is especially handy dealing with IPv6 prefixes and if `[no] bgp default show-nexthop-hostname` is enabled.

If `all` option is specified, `ip` keyword is ignored and, routes displayed for all AFIs and SAFIs. if afi is specified, with `all` option, routes will be displayed for each SAFI in the selcted AFI

If a specific prefix is specified, the detailed version of that prefix will be displayed.

If `detail` option is specified, the detailed version of all routes will be displayed. The same format as `show [ip] bgp [afi] [safi] PREFIX` will be used, but for the whole table of received, advertised or filtered prefixes.

If `json` option is specified, output is displayed in JSON format.

show [ip] bgp [afi] [safi] [all] detail-routes

Display the detailed version of all routes. The same format as using `show [ip] bgp [afi] [safi] PREFIX`, but for the whole BGP table.

If `all` option is specified, `ip` keyword is ignored and, routes displayed for all AFIs and SAFIs.

If `afi` is specified, with `all` option, routes will be displayed for each SAFI in the selected AFI.

show [ip] bgp [<view|vrf> VIEWVRFNAME] [afi] [safi] detail [json]

Display the detailed version of all routes from the specified bgp vrf table for a given afi + safi.

If no vrf is specified, then it is assumed as a default vrf and routes are displayed from default vrf table.

If `all` option is specified as vrf name, then all bgp vrf tables routes from a given afi+safi are displayed in the detailed output of routes.

If `json` option is specified, detailed output is displayed in JSON format.

Following are sample output for few examples of how to use this command.

```
torm-23# sh bgp ipv4 unicast detail (OR) sh bgp vrf default ipv4 unicast detail

!--- Output suppressed.

BGP routing table entry for 172.16.16.1/32
Paths: (1 available, best #1, table default)
  Not advertised to any peer
  Local, (Received from a RR-client)
    172.16.16.1 (metric 20) from torm-22(172.16.16.1) (192.168.0.10)
      Origin IGP, metric 0, localpref 100, valid, internal
      Last update: Fri May  8 12:54:05 2023
BGP routing table entry for 172.16.16.2/32
Paths: (1 available, best #1, table default)
  Not advertised to any peer
  Local
    0.0.0.0 from 0.0.0.0 (172.16.16.2)
      Origin incomplete, metric 0, weight 32768, valid, sourced, bestpath-from-AS Local,␣
↪best (First path received)
      Last update: Wed May  8 12:54:41 2023

Displayed  2 routes and 2 total paths
```

```
torm-23# sh bgp vrf all detail

Instance default:

!--- Output suppressed.
```

(continues on next page)

(continued from previous page)

```
BGP routing table entry for 172.16.16.1/32
Paths: (1 available, best #1, table default)
  Not advertised to any peer
  Local, (Received from a RR-client)
    172.16.16.1 (metric 20) from torm-22(172.16.16.1) (192.168.0.10)
      Origin IGP, metric 0, localpref 100, valid, internal
      Last update: Fri May  8 12:44:05 2023
BGP routing table entry for 172.16.16.2/32
Paths: (1 available, best #1, table default)
  Not advertised to any peer
  Local
    0.0.0.0 from 0.0.0.0 (172.16.16.2)
      Origin incomplete, metric 0, weight 32768, valid, sourced, bestpath-from-AS Local,␣
↪best (First path received)
      Last update: Wed May  8 12:45:01 2023

Displayed  2 routes and 2 total paths

Instance vrf3:

!--- Output suppressed.

BGP routing table entry for 192.168.0.2/32
Paths: (1 available, best #1, vrf vrf3)
  Not advertised to any peer
  Imported from 172.16.16.1:12:[2]:[0]:[48]:[00:02:00:00:00:58]:[32]:[192.168.0.2], VNI␣
↪1008/4003
  Local
    172.16.16.1 from torm-22(172.16.16.1) (172.16.16.1) announce-nh-self
      Origin IGP, localpref 100, valid, internal, bestpath-from-AS Local, best (First␣
↪path received)
      Extended Community: RT:65000:1008 ET:8 Rmac:00:02:00:00:00:58
      Last update: Fri May  8 02:41:55 2023
BGP routing table entry for 192.168.1.2/32
Paths: (1 available, best #1, vrf vrf3)
  Not advertised to any peer
  Imported from 172.16.16.1:13:[2]:[0]:[48]:[00:02:00:00:00:58]:[32]:[192.168.1.2], VNI␣
↪1009/4003
  Local
    172.16.16.1 from torm-22(172.16.16.1) (172.16.16.1) announce-nh-self
      Origin IGP, localpref 100, valid, internal, bestpath-from-AS Local, best (First␣
↪path received)
      Extended Community: RT:65000:1009 ET:8 Rmac:00:02:00:00:00:58
      Last update: Fri May  8 02:41:55 2023

Displayed  2 routes and 2 total paths
```

```
torm-23# sh bgp vrf vrf3 ipv4 unicast detail

!--- Output suppressed.

BGP routing table entry for 192.168.0.2/32
```

(continues on next page)

(continued from previous page)

```
Paths: (1 available, best #1, vrf vrf3)
  Not advertised to any peer
  Imported from 172.16.16.1:12:[2]:[0]:[48]:[00:02:00:00:00:58]:[32]:[192.168.0.2], VNI␣
↪1008/4003
  Local
    172.16.16.1 from torm-22(172.16.16.1) (172.16.16.1) announce-nh-self
      Origin IGP, localpref 100, valid, internal, bestpath-from-AS Local, best (First␣
↪path received)
      Extended Community: RT:65000:1008 ET:8 Rmac:00:02:00:00:00:58
      Last update: Fri May  8 02:23:35 2023
BGP routing table entry for 192.168.1.2/32
Paths: (1 available, best #1, vrf vrf3)
  Not advertised to any peer
  Imported from 172.16.16.1:13:[2]:[0]:[48]:[00:02:00:00:00:58]:[32]:[192.168.1.2], VNI␣
↪1009/4003
  Local
    172.16.16.1 from torm-22(172.16.16.1) (172.16.16.1) announce-nh-self
      Origin IGP, localpref 100, valid, internal, bestpath-from-AS Local, best (First␣
↪path received)
      Extended Community: RT:65000:1009 ET:8 Rmac:00:02:00:00:00:58
      Last update: Fri May  8 02:23:55 2023

Displayed  2 routes and 2 total paths
```

Displaying Routes by Community Attribute

The following commands allow displaying routes based on their community attribute.

show [ip] bgp <ipv4|ipv6> [all] community [wide|json]

show [ip] bgp <ipv4|ipv6> [all] community COMMUNITY [wide|json]

show [ip] bgp <ipv4|ipv6> [all] community COMMUNITY exact-match [wide|json]

These commands display BGP routes which have the community attribute. attribute. When `COMMUNITY` is specified, BGP routes that match that community are displayed. When *exact-match* is specified, it display only routes that have an exact match.

show [ip] bgp <ipv4|ipv6> community-list WORD [json]

show [ip] bgp <ipv4|ipv6> community-list WORD exact-match [json]

These commands display BGP routes for the address family specified that match the specified community list. When *exact-match* is specified, it displays only routes that have an exact match.

If `wide` option is specified, then the prefix table's width is increased to fully display the prefix and the nexthop.

This is especially handy dealing with IPv6 prefixes and if `[no] bgp default show-nexthop-hostname` is enabled.

If `all` option is specified, `ip` keyword is ignored and, routes displayed for all AFIs and SAFIs. if afi is specified, with `all` option, routes will be displayed for each SAFI in the selcted AFI

If `json` option is specified, output is displayed in JSON format.

show bgp labelpool <chunks|inuse|ledger | requests|summary> [json]

These commands display information about the BGP labelpool used for the association of MPLS labels with routes for L3VPN and Labeled Unicast

If `chunks` option is specified, output shows the current list of label chunks granted to BGP by Zebra, indicating the start and end label in each chunk

If `inuse` option is specified, output shows the current inuse list of label to prefix mappings

If `ledger` option is specified, output shows ledger list of all label requests made per prefix

If `requests` option is specified, output shows current list of label requests which have not yet been fulfilled by the labelpool

If `summary` option is specified, output is a summary of the counts for the chunks, inuse, ledger and requests list along with the count of outstanding chunk requests to Zebra and the number of zebra reconnects that have happened

If `json` option is specified, output is displayed in JSON format.

Displaying Routes by Large Community Attribute

The following commands allow displaying routes based on their large community attribute.

show [ip] bgp <ipv4|ipv6> large-community

show [ip] bgp <ipv4|ipv6> large-community LARGE-COMMUNITY

show [ip] bgp <ipv4|ipv6> large-community LARGE-COMMUNITY exact-match

show [ip] bgp <ipv4|ipv6> large-community LARGE-COMMUNITY json

These commands display BGP routes which have the large community attribute. attribute. When `LARGE-COMMUNITY` is specified, BGP routes that match that large community are displayed. When *exact-match* is specified, it display only routes that have an exact match. When *json* is specified, it display routes in json format.

show [ip] bgp <ipv4|ipv6> large-community-list WORD

show [ip] bgp <ipv4|ipv6> large-community-list WORD exact-match

show [ip] bgp <ipv4|ipv6> large-community-list WORD json

These commands display BGP routes for the address family specified that match the specified large community list. When *exact-match* is specified, it displays only routes that have an exact match. When *json* is specified, it display routes in json format.

Displaying Routes by AS Path

show bgp ipv4|ipv6 regexp LINE

This commands displays BGP routes that matches a regular expression *line* (*BGP Regular Expressions*).

show [ip] bgp ipv4 vpn

show [ip] bgp ipv6 vpn

Print active IPV4 or IPV6 routes advertised via the VPN SAFI.

show bgp ipv4 vpn summary

show bgp ipv6 vpn summary

Print a summary of neighbor connections for the specified AFI/SAFI combination.

Displaying Routes by Route Distinguisher

show bgp [<ipv4|ipv6> vpn [route]] rd <all|RD>

For L3VPN address-families, routes can be displayed on a per-RD (Route Distinguisher) basis or for all RD's.

Displaying Update Group Information

show bgp update-groups [advertise-queue| advertised-routes |packet-queue]

Display Information about each individual update-group being used. If SUBGROUP-ID is specified only display about that particular group. If advertise-queue is specified the list of routes that need to be sent to the peers in the update-group is displayed, advertised-routes means the list of routes we have sent to the peers in the update-group and packet-queue specifies the list of packets in the queue to be sent.

show bgp update-groups statistics

Display Information about update-group events in FRR.

Displaying Nexthop Information

show [ip] bgp [<view|vrf> VIEWVRFNAME] nexthop ipv4 [A.B.C.D] [detail] [json]

show [ip] bgp [<view|vrf> VIEWVRFNAME] nexthop ipv6 [X:X::X:X] [detail] [json]

show [ip] bgp [<view|vrf> VIEWVRFNAME] nexthop [<A.B.C.D|X:X::X:X>] [detail] [json]

show [ip] bgp <view|vrf> all nexthop [json]

Display information about nexthops to bgp neighbors. If a certain nexthop is specified, also provides information about paths associated with the nexthop. With detail option provides information about gates of each nexthop.

show [ip] bgp [<view|vrf> VIEWVRFNAME] import-check-table [detail] [json]

Display information about nexthops from table that is used to check network's existence in the rib for network statements.

AS-notation support

By default, the ASN value output follows how the BGP ASN instance is expressed in the configuration. Three as-notation outputs are available:

- plain output: both AS4B and AS2B use a single number. ` router bgp 65536`.
- dot output: AS4B values are using two numbers separated by a period. *router bgp 1.1* means that the AS number is 65536.
- dot+ output: AS2B and AS4B values are using two numbers separated by a period. *router bgp 0.5* means that the AS number is 5.

The below option permits forcing the as-notation output:

router bgp ASN as-notation dot|dot+|plain

The chosen as-notation format will override the BGP ASN output.

Route Reflector

BGP routers connected inside the same AS through BGP belong to an internal BGP session, or IBGP. In order to prevent routing table loops, IBGP does not advertise IBGP-learned routes to other routers in the same session. As such, IBGP requires a full mesh of all peers. For large networks, this quickly becomes unscalable. Introducing route reflectors removes the need for the full-mesh.

When route reflectors are configured, these will reflect the routes announced by the peers configured as clients. A route reflector client is configured with:

neighbor PEER route-reflector-client

To avoid single points of failure, multiple route reflectors can be configured.

A cluster is a collection of route reflectors and their clients, and is used by route reflectors to avoid looping.

bgp cluster-id A.B.C.D

bgp session-dscp (0-63)

This command allows bgp to control, at a global level, the TCP dscp values in the TCP header.

Routing Policy

You can set different routing policy for a peer. For example, you can set different filter for a peer.

```
!
router bgp 1 view 1
 neighbor 10.0.0.1 remote-as 2
 address-family ipv4 unicast
  neighbor 10.0.0.1 distribute-list 1 in
 exit-address-family
!
router bgp 1 view 2
 neighbor 10.0.0.1 remote-as 2
 address-family ipv4 unicast
  neighbor 10.0.0.1 distribute-list 2 in
 exit-address-family
```

This means BGP update from a peer 10.0.0.1 goes to both BGP view 1 and view 2. When the update is inserted into view 1, distribute-list 1 is applied. On the other hand, when the update is inserted into view 2, distribute-list 2 is applied.

BGP Regular Expressions

BGP regular expressions are based on *POSIX 1003.2* regular expressions. The following description is just a quick subset of the POSIX regular expressions.

.*
: Matches any single character.

*****
: Matches 0 or more occurrences of pattern.

+
: Matches 1 or more occurrences of pattern.

?
: Match 0 or 1 occurrences of pattern.

^
: Matches the beginning of the line.

$
: Matches the end of the line.

_
: The _ character has special meanings in BGP regular expressions. It matches to space and comma , and AS set delimiter { and } and AS confederation delimiter (and). And it also matches to the beginning of the line and the end of the line. So _ can be used for AS value boundaries match. This character technically evaluates to (^|[,{}()]|$).

Miscellaneous Configuration Examples

Example of a session to an upstream, advertising only one prefix to it.

```
router bgp 64512
 bgp router-id 10.236.87.1
 neighbor upstream peer-group
 neighbor upstream remote-as 64515
 neighbor upstream capability dynamic
 neighbor 10.1.1.1 peer-group upstream
 neighbor 10.1.1.1 description ACME ISP

 address-family ipv4 unicast
  network 10.236.87.0/24
  neighbor upstream prefix-list pl-allowed-adv out
 exit-address-family
!
ip prefix-list pl-allowed-adv seq 5 permit 82.195.133.0/25
ip prefix-list pl-allowed-adv seq 10 deny any
```

A more complex example including upstream, peer and customer sessions advertising global prefixes and NO_EXPORT prefixes and providing actions for customer routes based on community values. Extensive use is made of route-maps and the 'call' feature to support selective advertising of prefixes. This example is intended as guidance only, it has NOT been tested and almost certainly contains silly mistakes, if not serious flaws.

```
router bgp 64512
 bgp router-id 10.236.87.1
 neighbor upstream capability dynamic
 neighbor cust capability dynamic
 neighbor peer capability dynamic
 neighbor 10.1.1.1 remote-as 64515
 neighbor 10.1.1.1 peer-group upstream
 neighbor 10.2.1.1 remote-as 64516
 neighbor 10.2.1.1 peer-group upstream
 neighbor 10.3.1.1 remote-as 64517
 neighbor 10.3.1.1 peer-group cust-default
 neighbor 10.3.1.1 description customer1
 neighbor 10.4.1.1 remote-as 64518
 neighbor 10.4.1.1 peer-group cust
 neighbor 10.4.1.1 description customer2
 neighbor 10.5.1.1 remote-as 64519
 neighbor 10.5.1.1 peer-group peer
```

(continues on next page)

(continued from previous page)

```
 neighbor 10.5.1.1 description peer AS 1
 neighbor 10.6.1.1 remote-as 64520
 neighbor 10.6.1.1 peer-group peer
 neighbor 10.6.1.1 description peer AS 2

 address-family ipv4 unicast
  network 10.123.456.0/24
  network 10.123.456.128/25 route-map rm-no-export
  neighbor upstream route-map rm-upstream-out out
  neighbor cust route-map rm-cust-in in
  neighbor cust route-map rm-cust-out out
  neighbor cust send-community both
  neighbor peer route-map rm-peer-in in
  neighbor peer route-map rm-peer-out out
  neighbor peer send-community both
  neighbor 10.3.1.1 prefix-list pl-cust1-network in
  neighbor 10.4.1.1 prefix-list pl-cust2-network in
  neighbor 10.5.1.1 prefix-list pl-peer1-network in
  neighbor 10.6.1.1 prefix-list pl-peer2-network in
 exit-address-family
!
ip prefix-list pl-default permit 0.0.0.0/0
!
ip prefix-list pl-upstream-peers permit 10.1.1.1/32
ip prefix-list pl-upstream-peers permit 10.2.1.1/32
!
ip prefix-list pl-cust1-network permit 10.3.1.0/24
ip prefix-list pl-cust1-network permit 10.3.2.0/24
!
ip prefix-list pl-cust2-network permit 10.4.1.0/24
!
ip prefix-list pl-peer1-network permit 10.5.1.0/24
ip prefix-list pl-peer1-network permit 10.5.2.0/24
ip prefix-list pl-peer1-network permit 192.168.0.0/24
!
ip prefix-list pl-peer2-network permit 10.6.1.0/24
ip prefix-list pl-peer2-network permit 10.6.2.0/24
ip prefix-list pl-peer2-network permit 192.168.1.0/24
ip prefix-list pl-peer2-network permit 192.168.2.0/24
ip prefix-list pl-peer2-network permit 172.16.1/24
!
bgp as-path access-list seq 5 asp-own-as permit ^$
bgp as-path access-list seq 10 asp-own-as permit _64512_
!
! ###################################################################
! Match communities we provide actions for, on routes receives from
! customers. Communities values of <our-ASN>:X, with X, have actions:
!
! 100 - blackhole the prefix
! 200 - set no_export
! 300 - advertise only to other customers
! 400 - advertise only to upstreams
```

(continues on next page)

(continued from previous page)

```
! 500 - set no_export when advertising to upstreams
! 2X00 - set local_preference to X00
!
! blackhole the prefix of the route
bgp community-list standard cm-blackhole permit 64512:100
!
! set no-export community before advertising
bgp community-list standard cm-set-no-export permit 64512:200
!
! advertise only to other customers
bgp community-list standard cm-cust-only permit 64512:300
!
! advertise only to upstreams
bgp community-list standard cm-upstream-only permit 64512:400
!
! advertise to upstreams with no-export
bgp community-list standard cm-upstream-noexport permit 64512:500
!
! set local-pref to least significant 3 digits of the community
bgp community-list standard cm-prefmod-100 permit 64512:2100
bgp community-list standard cm-prefmod-200 permit 64512:2200
bgp community-list standard cm-prefmod-300 permit 64512:2300
bgp community-list standard cm-prefmod-400 permit 64512:2400
bgp community-list expanded cme-prefmod-range permit 64512:2...
!
! Informational communities
!
! 3000 - learned from upstream
! 3100 - learned from customer
! 3200 - learned from peer
!
bgp community-list standard cm-learnt-upstream permit 64512:3000
bgp community-list standard cm-learnt-cust permit 64512:3100
bgp community-list standard cm-learnt-peer permit 64512:3200
!
! ################################################################
! Utility route-maps
!
! These utility route-maps generally should not used to permit/deny
! routes, i.e. they do not have meaning as filters, and hence probably
! should be used with 'on-match next'. These all finish with an empty
! permit entry so as not interfere with processing in the caller.
!
route-map rm-no-export permit 10
 set community additive no-export
route-map rm-no-export permit 20
!
route-map rm-blackhole permit 10
 description blackhole, up-pref and ensure it cannot escape this AS
 set ip next-hop 127.0.0.1
 set local-preference 10
 set community additive no-export
```

(continues on next page)

(continued from previous page)

```
route-map rm-blackhole permit 20
!
! Set local-pref as requested
route-map rm-prefmod permit 10
 match community cm-prefmod-100
 set local-preference 100
route-map rm-prefmod permit 20
 match community cm-prefmod-200
 set local-preference 200
route-map rm-prefmod permit 30
 match community cm-prefmod-300
 set local-preference 300
route-map rm-prefmod permit 40
 match community cm-prefmod-400
 set local-preference 400
route-map rm-prefmod permit 50
!
! Community actions to take on receipt of route.
route-map rm-community-in permit 10
 description check for blackholing, no point continuing if it matches.
 match community cm-blackhole
 call rm-blackhole
route-map rm-community-in permit 20
 match community cm-set-no-export
 call rm-no-export
 on-match next
route-map rm-community-in permit 30
 match community cme-prefmod-range
 call rm-prefmod
route-map rm-community-in permit 40
!
! ######################################################################
! Community actions to take when advertising a route.
! These are filtering route-maps,
!
! Deny customer routes to upstream with cust-only set.
route-map rm-community-filt-to-upstream deny 10
 match community cm-learnt-cust
 match community cm-cust-only
route-map rm-community-filt-to-upstream permit 20
!
! Deny customer routes to other customers with upstream-only set.
route-map rm-community-filt-to-cust deny 10
 match community cm-learnt-cust
 match community cm-upstream-only
route-map rm-community-filt-to-cust permit 20
!
! ######################################################################
! The top-level route-maps applied to sessions. Further entries could
! be added obviously..
!
! Customers
```

(continues on next page)

(continued from previous page)

```
route-map rm-cust-in permit 10
 call rm-community-in
 on-match next
route-map rm-cust-in permit 20
 set community additive 64512:3100
route-map rm-cust-in permit 30
!
route-map rm-cust-out permit 10
 call rm-community-filt-to-cust
 on-match next
route-map rm-cust-out permit 20
!
! Upstream transit ASes
route-map rm-upstream-out permit 10
 description filter customer prefixes which are marked cust-only
 call rm-community-filt-to-upstream
 on-match next
route-map rm-upstream-out permit 20
 description only customer routes are provided to upstreams/peers
 match community cm-learnt-cust
!
! Peer ASes
! outbound policy is same as for upstream
route-map rm-peer-out permit 10
 call rm-upstream-out
!
route-map rm-peer-in permit 10
 set community additive 64512:3200
```

Example of how to set up a 6-Bone connection.

```
! bgpd configuration
! ==================
!
! MP-BGP configuration
!
router bgp 7675
 bgp router-id 10.0.0.1
 neighbor 3ffe:1cfa:0:2:2a0:c9ff:fe9e:f56 remote-as `as-number`
!
 address-family ipv6
 network 3ffe:506::/32
 neighbor 3ffe:1cfa:0:2:2a0:c9ff:fe9e:f56 activate
 neighbor 3ffe:1cfa:0:2:2a0:c9ff:fe9e:f56 route-map set-nexthop out
 neighbor 3ffe:1cfa:0:2:2c0:4fff:fe68:a231 remote-as `as-number`
 neighbor 3ffe:1cfa:0:2:2c0:4fff:fe68:a231 route-map set-nexthop out
 exit-address-family
!
ipv6 access-list all permit any
!
! Set output nexthop address.
!
```

(continues on next page)

(continued from previous page)

```
route-map set-nexthop permit 10
 match ipv6 address all
 set ipv6 nexthop global 3ffe:1cfa:0:2:2c0:4fff:fe68:a225
 set ipv6 nexthop local fe80::2c0:4fff:fe68:a225
!
log syslog
!
```

BGP tcp-mss support

TCP provides a mechanism for the user to specify the max segment size. setsockopt API is used to set the max segment size for TCP session. We can configure this as part of BGP neighbor configuration.

This document explains how to avoid ICMP vulnerability issues by limiting TCP max segment size when you are using MTU discovery. Using MTU discovery on TCP paths is one method of avoiding BGP packet fragmentation.

TCP negotiates a maximum segment size (MSS) value during session connection establishment between two peers. The MSS value negotiated is primarily based on the maximum transmission unit (MTU) of the interfaces to which the communicating peers are directly connected. However, due to variations in link MTU on the path taken by the TCP packets, some packets in the network that are well within the MSS value might be fragmented when the packet size exceeds the link's MTU.

This feature is supported with TCP over IPv4 and TCP over IPv6.

CLI Configuration:

Below configuration can be done in router bgp mode and allows the user to configure the tcp-mss value per neighbor. The configuration gets applied only after hard reset is performed on that neighbor. If we configure tcp-mss on both the neighbors then both neighbors need to be reset.

The configuration takes effect based on below rules, so there is a configured tcp-mss and a synced tcp-mss value per TCP session.

By default if the configuration is not done then the TCP max segment size is set to the Maximum Transmission unit (MTU) – (IP/IP6 header size + TCP header size + ethernet header). For IPv4 its MTU – (20 bytes IP header + 20 bytes TCP header + 12 bytes ethernet header) and for IPv6 its MTU – (40 bytes IPv6 header + 20 bytes TCP header + 12 bytes ethernet header).

If the config is done then it reduces 12-14 bytes for the ether header and uses it after synchronizing in TCP handshake.

neighbor <A.B.C.D|X:X::X:X|WORD> tcp-mss (1-65535)

When tcp-mss is configured kernel reduces 12-14 bytes for ethernet header. E.g. if tcp-mss is configured as 150 the synced value will be 138.

Note: configured and synced value is different since TCP module will reduce 12 bytes for ethernet header.

Running config:

```
soodar# show running-config
Building configuration...

Current configuration:
!
router bgp 100
 bgp router-id 192.0.2.1
 neighbor 198.51.100.2 remote-as 100
 neighbor 198.51.100.2 tcp-mss 150       => new entry
 neighbor 2001:DB8::2 remote-as 100
 neighbor 2001:DB8::2 tcp-mss 400        => new entry
```

Show command:

```
soodar# show bgp neighbors 198.51.100.2
BGP neighbor is 198.51.100.2, remote AS 100, local AS 100, internal link
Hostname: frr
  BGP version 4, remote router ID 192.0.2.2, local router ID 192.0.2.1
  BGP state = Established, up for 02:15:28
  Last read 00:00:28, Last write 00:00:28
  Hold time is 180, keepalive interval is 60 seconds
  Configured tcp-mss is 150, synced tcp-mss is 138     => new display
```

```
soodar# show bgp neighbors 2001:DB8::2
BGP neighbor is 2001:DB8::2, remote AS 100, local AS 100, internal link
Hostname: frr
  BGP version 4, remote router ID 192.0.2.2, local router ID 192.0.2.1
  BGP state = Established, up for 02:16:34
  Last read 00:00:34, Last write 00:00:34
  Hold time is 180, keepalive interval is 60 seconds
  Configured tcp-mss is 400, synced tcp-mss is 388      => new display
```

Show command json output:

```
soodar# show bgp neighbors 2001:DB8::2 json
{
  "2001:DB8::2":{
    "remoteAs":100,
    "localAs":100,
    "nbrInternalLink":true,
    "hostname":"frr",
    "bgpVersion":4,
    "remoteRouterId":"192.0.2.2",
    "localRouterId":"192.0.2.1",
    "bgpState":"Established",
    "bgpTimerUpMsec":8349000,
    "bgpTimerUpString":"02:19:09",
```

(continues on next page)

(continued from previous page)

```
    "bgpTimerUpEstablishedEpoch":1613054251,
    "bgpTimerLastRead":9000,
    "bgpTimerLastWrite":9000,
    "bgpInUpdateElapsedTimeMsecs":8347000,
    "bgpTimerHoldTimeMsecs":180000,
    "bgpTimerKeepAliveIntervalMsecs":60000,
    "bgpTcpMssConfigured":400,                                  => new entry
    "bgpTcpMssSynced":388,                                    => new entry
```

```
soodar# show bgp neighbors 198.51.100.2 json
{
  "198.51.100.2":{
    "remoteAs":100,
    "localAs":100,
    "nbrInternalLink":true,
    "hostname":"frr",
    "bgpVersion":4,
    "remoteRouterId":"192.0.2.2",
    "localRouterId":"192.0.2.1",
    "bgpState":"Established",
    "bgpTimerUpMsec":8370000,
    "bgpTimerUpString":"02:19:30",
    "bgpTimerUpEstablishedEpoch":1613054251,
    "bgpTimerLastRead":30000,
    "bgpTimerLastWrite":30000,
    "bgpInUpdateElapsedTimeMsecs":8368000,
    "bgpTimerHoldTimeMsecs":180000,
    "bgpTimerKeepAliveIntervalMsecs":60000,
    "bgpTcpMssConfigured":150,                                  => new entry
    "bgpTcpMssSynced":138,                                    => new entry
```

Configuring FRR as a Route Server

The purpose of a Route Server is to centralize the peerings between BGP speakers. For example if we have an exchange point scenario with four BGP speakers, each of which maintaining a BGP peering with the other three (fig-topologies-full), we can convert it into a centralized scenario where each of the four establishes a single BGP peering against the Route Server (fig-topologies-rs).

We will first describe briefly the Route Server model implemented by FRR. We will explain the commands that have been added for configuring that model. And finally we will show a full example of FRR configured as Route Server.

Description of the Route Server model

First we are going to describe the normal processing that BGP announcements suffer inside a standard BGP speaker, as shown in fig-normal-processing, it consists of three steps:

- When an announcement is received from some peer, the *In* filters configured for that peer are applied to the announcement. These filters can reject the announcement, accept it unmodified, or accept it with some of its attributes modified.
- The announcements that pass the *In* filters go into the Best Path Selection process, where they are compared to other announcements referred to the same destination that have been received from different peers (in case such

other announcements exist). For each different destination, the announcement which is selected as the best is inserted into the BGP speaker's Loc-RIB.

- The routes which are inserted in the Loc-RIB are considered for announcement to all the peers (except the one from which the route came). This is done by passing the routes in the Loc-RIB through the *Out* filters corresponding to each peer. These filters can reject the route, accept it unmodified, or accept it with some of its attributes modified. Those routes which are accepted by the *Out* filters of a peer are announced to that peer.

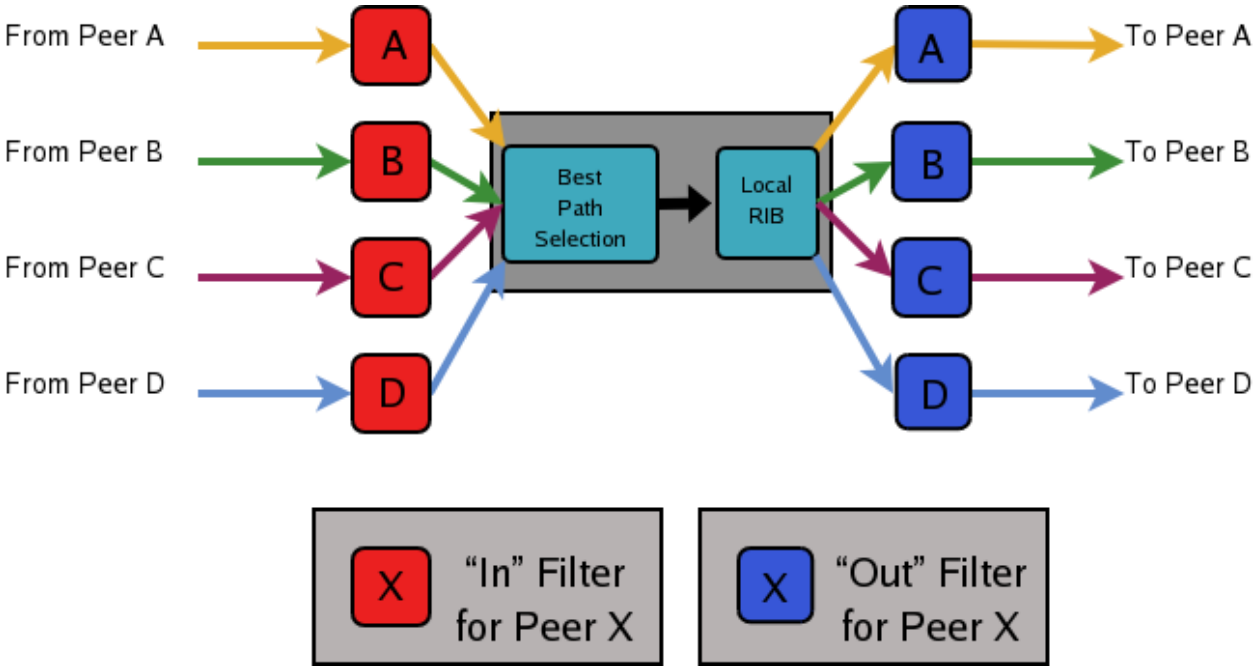


Fig. 1: Announcement processing inside a 'normal' BGP speaker

Of course we want that the routing tables obtained in each of the routers are the same when using the route server than when not. But as a consequence of having a single BGP peering (against the route server), the BGP speakers can no longer distinguish from/to which peer each announce comes/goes.

This means that the routers connected to the route server are not able to apply by themselves the same input/output filters as in the full mesh scenario, so they have to delegate those functions to the route server.

Even more, the 'best path' selection must be also performed inside the route server on behalf of its clients. The reason is that if, after applying the filters of the announcer and the (potential) receiver, the route server decides to send to some client two or more different announcements referred to the same destination, the client will only retain the last one, considering it as an implicit withdrawal of the previous announcements for the same destination. This is the expected behavior of a BGP speaker as defined in **RFC 1771**, and even though there are some proposals of mechanisms that permit multiple paths for the same destination to be sent through a single BGP peering, none are currently supported by most existing BGP implementations.

As a consequence a route server must maintain additional information and perform additional tasks for a RS-client that those necessary for common BGP peerings. Essentially a route server must:

- Maintain a separated Routing Information Base (Loc-RIB) for each peer configured as RS-client, containing the routes selected as a result of the 'Best Path Selection' process that is performed on behalf of that RS-client.
- Whenever it receives an announcement from a RS-client, it must consider it for the Loc-RIBs of the other RS-clients.
  - This means that for each of them the route server must pass the announcement through the appropriate *Out* filter of the announcer.

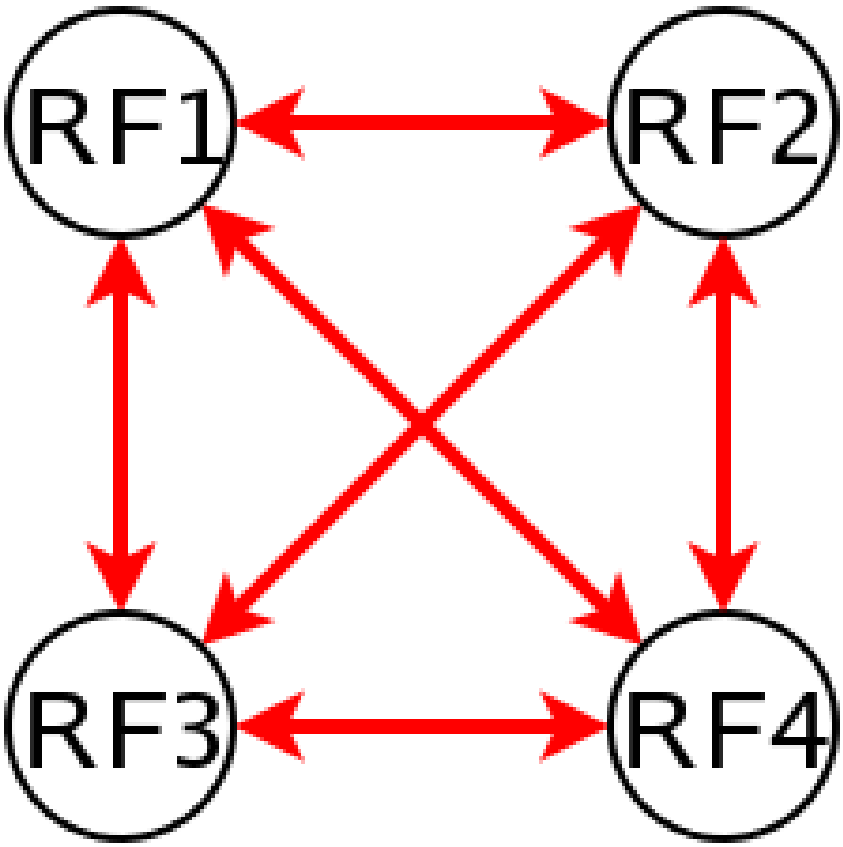


Fig. 2: Full Mesh

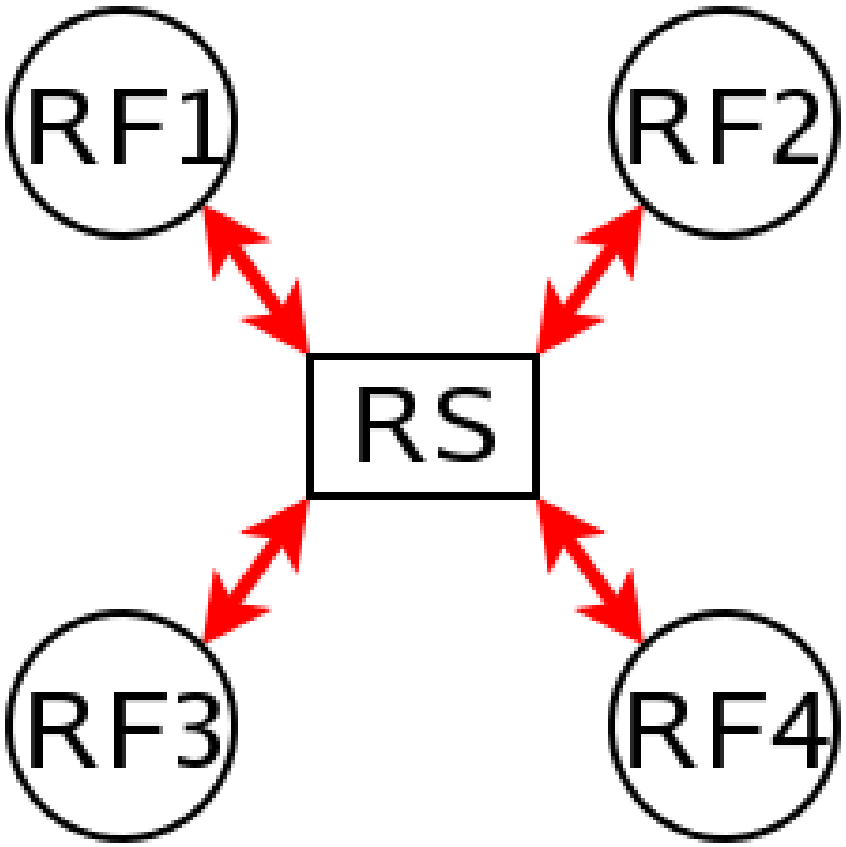


Fig. 3: Route server and clients

- Then through the appropriate *In* filter of the potential receiver.
- Only if the announcement is accepted by both filters it will be passed to the 'Best Path Selection' process.
- Finally, it might go into the Loc-RIB of the receiver.

When we talk about the 'appropriate' filter, both the announcer and the receiver of the route must be taken into account. Suppose that the route server receives an announcement from client A, and the route server is considering it for the Loc-RIB of client B. The filters that should be applied are the same that would be used in the full mesh scenario, i.e., first the *Out* filter of router A for announcements going to router B, and then the *In* filter of router B for announcements coming from router A.

We call 'Export Policy' of a RS-client to the set of *Out* filters that the client would use if there was no route server. The same applies for the 'Import Policy' of a RS-client and the set of *In* filters of the client if there was no route server.

It is also common to demand from a route server that it does not modify some BGP attributes (next-hop, as-path and MED) that are usually modified by standard BGP speakers before announcing a route.

The announcement processing model implemented by FRR is shown in fig-rs-processing. The figure shows a mixture of RS-clients (B, C and D) with normal BGP peers (A). There are some details that worth additional comments:

- Announcements coming from a normal BGP peer are also considered for the Loc-RIBs of all the RS-clients. But logically they do not pass through any export policy.
- Those peers that are configured as RS-clients do not receive any announce from the *Main* Loc-RIB.
- Apart from import and export policies, *In* and *Out* filters can also be set for RS-clients. *In* filters might be useful when the route server has also normal BGP peers. On the other hand, *Out* filters for RS-clients are probably unnecessary, but we decided not to remove them as they do not hurt anybody (they can always be left empty).

Commands for configuring a Route Server

Now we will describe the commands that have been added to frr in order to support the route server features.

`neighbor PEER-GROUP route-server-client`

`neighbor A.B.C.D route-server-client`

`neighbor X:X::X:X route-server-client`

This command configures the peer given by *peer*, *A.B.C.D* or *X:X::X:X* as an RS-client.

Actually this command is not new, it already existed in standard FRR. It enables the transparent mode for the specified peer. This means that some BGP attributes (as-path, next-hop and MED) of the routes announced to that peer are not modified.

With the route server patch, this command, apart from setting the transparent mode, creates a new Loc-RIB dedicated to the specified peer (those named *Loc-RIB for X* in fig-rs-processing.). Starting from that moment, every announcement received by the route server will be also considered for the new Loc-RIB.

`neigbor A.B.C.D|X.X::X.X|peer-group route-map WORD in|out`

This set of commands can be used to specify the route-map that represents the Import or Export policy of a peer which is configured as a RS-client (with the previous command).

`match peer A.B.C.D|X:X::X:X`

This is a new *match* statement for use in route-maps, enabling them to describe import/export policies. As we said before, an import/export policy represents a set of input/output filters of the RS-client. This statement makes possible that a single route-map represents the full set of filters that a BGP speaker would use for its different peers in a non-RS scenario.

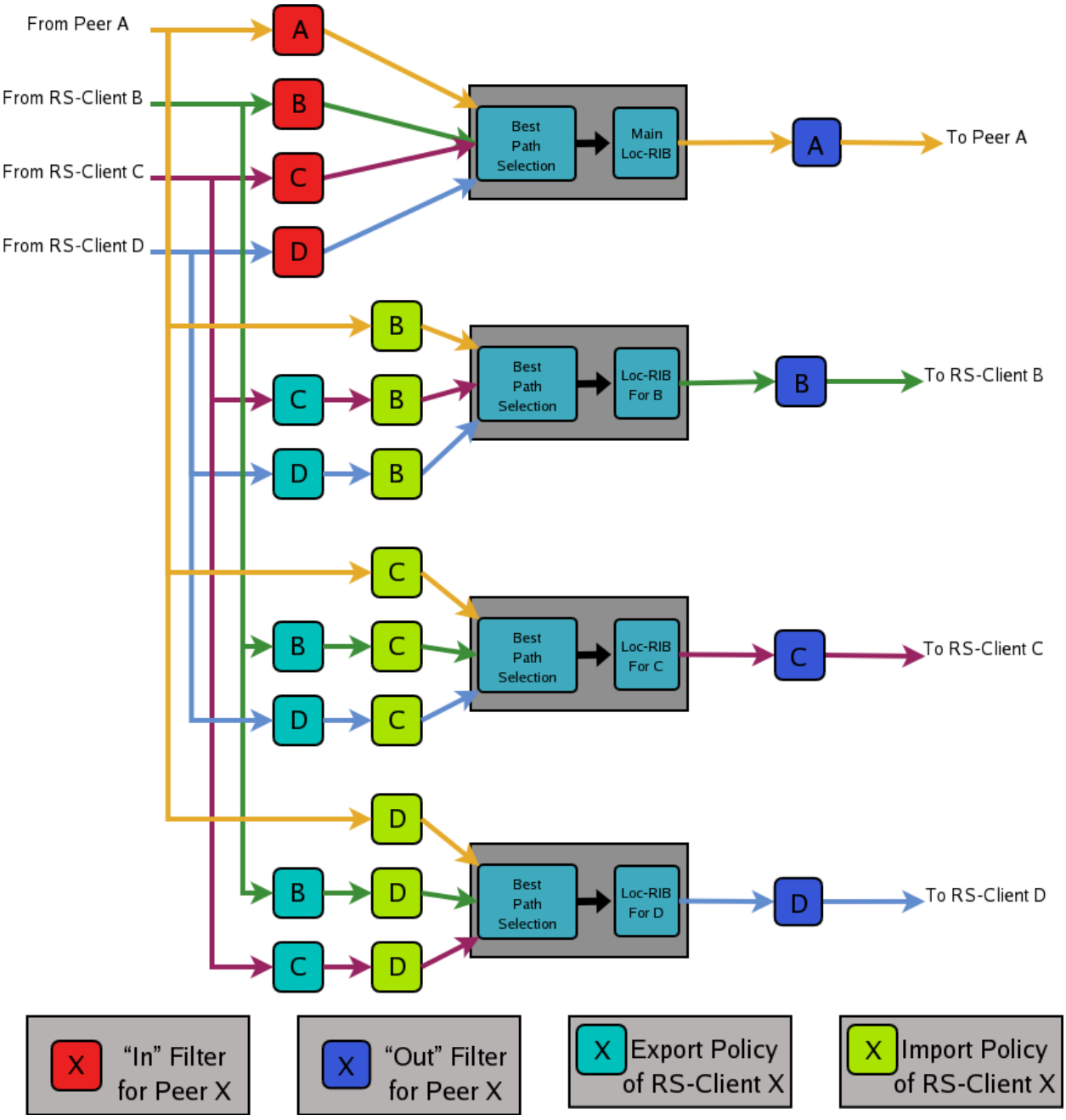


Fig. 4: Announcement processing model implemented by the Route Server

The *match peer* statement has different semantics whether it is used inside an import or an export route-map. In the first case the statement matches if the address of the peer who sends the announce is the same that the address specified by {A.B.C.D|X:X::X:X}. For export route-maps it matches when {A.B.C.D|X:X::X:X} is the address of the RS-Client into whose Loc-RIB the announce is going to be inserted (how the same export policy is applied before different Loc-RIBs is shown in fig-rs-processing.).

call WORD

This command (also used inside a route-map) jumps into a different route-map, whose name is specified by *WORD*. When the called route-map finishes, depending on its result the original route-map continues or not. Apart from being useful for making import/export route-maps easier to write, this command can also be used inside any normal (in or out) route-map.

Example of Route Server Configuration

Finally we are going to show how to configure a FRR daemon to act as a Route Server. For this purpose we are going to present a scenario without route server, and then we will show how to use the configurations of the BGP routers to generate the configuration of the route server.

All the configuration files shown in this section have been taken from scenarios which were tested using the VNUML tool http://www.dit.upm.es/vnuml,VNUML.

Configuration of the BGP routers without Route Server

We will suppose that our initial scenario is an exchange point with three BGP capable routers, named RA, RB and RC. Each of the BGP speakers generates some routes (with the *network* command), and establishes BGP peerings against the other two routers. These peerings have In and Out route-maps configured, named like 'PEER-X-IN' or 'PEER-X-OUT'. For example the configuration file for router RA could be the following:

```
#Configuration for router 'RA'
!
hostname RA
password ****
!
router bgp 65001
  no bgp default ipv4-unicast
  neighbor 2001:0DB8::B remote-as 65002
  neighbor 2001:0DB8::C remote-as 65003
!
  address-family ipv6
    network 2001:0DB8:AAAA:1::/64
    network 2001:0DB8:AAAA:2::/64
    network 2001:0DB8:0000:1::/64
    network 2001:0DB8:0000:2::/64
    neighbor 2001:0DB8::B activate
    neighbor 2001:0DB8::B soft-reconfiguration inbound
    neighbor 2001:0DB8::B route-map PEER-B-IN in
    neighbor 2001:0DB8::B route-map PEER-B-OUT out
    neighbor 2001:0DB8::C activate
    neighbor 2001:0DB8::C soft-reconfiguration inbound
    neighbor 2001:0DB8::C route-map PEER-C-IN in
    neighbor 2001:0DB8::C route-map PEER-C-OUT out
  exit-address-family
```

(continues on next page)

(continued from previous page)

```
!
ipv6 prefix-list COMMON-PREFIXES seq  5 permit 2001:0DB8:0000::/48 ge 64 le 64
ipv6 prefix-list COMMON-PREFIXES seq 10 deny any
!
ipv6 prefix-list PEER-A-PREFIXES seq  5 permit 2001:0DB8:AAAA::/48 ge 64 le 64
ipv6 prefix-list PEER-A-PREFIXES seq 10 deny any
!
ipv6 prefix-list PEER-B-PREFIXES seq  5 permit 2001:0DB8:BBBB::/48 ge 64 le 64
ipv6 prefix-list PEER-B-PREFIXES seq 10 deny any
!
ipv6 prefix-list PEER-C-PREFIXES seq  5 permit 2001:0DB8:CCCC::/48 ge 64 le 64
ipv6 prefix-list PEER-C-PREFIXES seq 10 deny any
!
route-map PEER-B-IN permit 10
  match ipv6 address prefix-list COMMON-PREFIXES
  set metric 100
route-map PEER-B-IN permit 20
  match ipv6 address prefix-list PEER-B-PREFIXES
  set community 65001:11111
!
route-map PEER-C-IN permit 10
  match ipv6 address prefix-list COMMON-PREFIXES
  set metric 200
route-map PEER-C-IN permit 20
  match ipv6 address prefix-list PEER-C-PREFIXES
  set community 65001:22222
!
route-map PEER-B-OUT permit 10
  match ipv6 address prefix-list PEER-A-PREFIXES
!
route-map PEER-C-OUT permit 10
  match ipv6 address prefix-list PEER-A-PREFIXES
!
line vty
!
```

Configuration of the BGP routers with Route Server

To convert the initial scenario into one with route server, first we must modify the configuration of routers RA, RB and RC. Now they must not peer between them, but only with the route server. For example, RA's configuration would turn into:

```
# Configuration for router 'RA'
!
hostname RA
password ****
!
router bgp 65001
  no bgp default ipv4-unicast
  neighbor 2001:0DB8::FFFF remote-as 65000
!
```

(continues on next page)

(continued from previous page)

```
  address-family ipv6
    network 2001:0DB8:AAAA:1::/64
    network 2001:0DB8:AAAA:2::/64
    network 2001:0DB8:0000:1::/64
    network 2001:0DB8:0000:2::/64

    neighbor 2001:0DB8::FFFF activate
    neighbor 2001:0DB8::FFFF soft-reconfiguration inbound
  exit-address-family
!
line vty
!
```

Which is logically much simpler than its initial configuration, as it now maintains only one BGP peering and all the filters (route-maps) have disappeared.

Configuration of the Route Server itself

As we said when we described the functions of a route server (description-of-the-route-server-model), it is in charge of all the route filtering. To achieve that, the In and Out filters from the RA, RB and RC configurations must be converted into Import and Export policies in the route server.

This is a fragment of the route server configuration (we only show the policies for client RA):

```
# Configuration for Route Server ('RS')
!
hostname RS
password ix
!
router bgp 65000 view RS
  no bgp default ipv4-unicast
  neighbor 2001:0DB8::A  remote-as 65001
  neighbor 2001:0DB8::B  remote-as 65002
  neighbor 2001:0DB8::C  remote-as 65003
!
  address-family ipv6
    neighbor 2001:0DB8::A activate
    neighbor 2001:0DB8::A route-server-client
    neighbor 2001:0DB8::A route-map RSCLIENT-A-IMPORT in
    neighbor 2001:0DB8::A route-map RSCLIENT-A-EXPORT out
    neighbor 2001:0DB8::A soft-reconfiguration inbound

    neighbor 2001:0DB8::B activate
    neighbor 2001:0DB8::B route-server-client
    neighbor 2001:0DB8::B route-map RSCLIENT-B-IMPORT in
    neighbor 2001:0DB8::B route-map RSCLIENT-B-EXPORT out
    neighbor 2001:0DB8::B soft-reconfiguration inbound

    neighbor 2001:0DB8::C activate
    neighbor 2001:0DB8::C route-server-client
    neighbor 2001:0DB8::C route-map RSCLIENT-C-IMPORT in
    neighbor 2001:0DB8::C route-map RSCLIENT-C-EXPORT out
```

(continues on next page)

(continued from previous page)

```
    neighbor 2001:0DB8::C soft-reconfiguration inbound
  exit-address-family
!
ipv6 prefix-list COMMON-PREFIXES seq  5 permit 2001:0DB8:0000::/48 ge 64 le 64
ipv6 prefix-list COMMON-PREFIXES seq 10 deny any
!
ipv6 prefix-list PEER-A-PREFIXES seq  5 permit 2001:0DB8:AAAA::/48 ge 64 le 64
ipv6 prefix-list PEER-A-PREFIXES seq 10 deny any
!
ipv6 prefix-list PEER-B-PREFIXES seq  5 permit 2001:0DB8:BBBB::/48 ge 64 le 64
ipv6 prefix-list PEER-B-PREFIXES seq 10 deny any
!
ipv6 prefix-list PEER-C-PREFIXES seq  5 permit 2001:0DB8:CCCC::/48 ge 64 le 64
ipv6 prefix-list PEER-C-PREFIXES seq 10 deny any
!
route-map RSCLIENT-A-IMPORT permit 10
  match peer 2001:0DB8::B
  call A-IMPORT-FROM-B
route-map RSCLIENT-A-IMPORT permit 20
  match peer 2001:0DB8::C
  call A-IMPORT-FROM-C
!
route-map A-IMPORT-FROM-B permit 10
  match ipv6 address prefix-list COMMON-PREFIXES
  set metric 100
route-map A-IMPORT-FROM-B permit 20
  match ipv6 address prefix-list PEER-B-PREFIXES
  set community 65001:11111
!
route-map A-IMPORT-FROM-C permit 10
  match ipv6 address prefix-list COMMON-PREFIXES
  set metric 200
route-map A-IMPORT-FROM-C permit 20
  match ipv6 address prefix-list PEER-C-PREFIXES
  set community 65001:22222
!
route-map RSCLIENT-A-EXPORT permit 10
  match peer 2001:0DB8::B
  match ipv6 address prefix-list PEER-A-PREFIXES
route-map RSCLIENT-A-EXPORT permit 20
  match peer 2001:0DB8::C
  match ipv6 address prefix-list PEER-A-PREFIXES
!
...
...
...
```

If you compare the initial configuration of RA with the route server configuration above, you can see how easy it is to generate the Import and Export policies for RA from the In and Out route-maps of RA's original configuration.

When there was no route server, RA maintained two peerings, one with RB and another with RC. Each of this peerings had an In route-map configured. To build the Import route-map for client RA in the route server, simply add route-map entries following this scheme:

```
route-map <NAME> permit 10
    match peer <Peer Address>
    call <In Route-Map for this Peer>
route-map <NAME> permit 20
    match peer <Another Peer Address>
    call <In Route-Map for this Peer>
```

This is exactly the process that has been followed to generate the route-map RSCLIENT-A-IMPORT. The route-maps that are called inside it (A-IMPORT-FROM-B and A-IMPORT-FROM-C) are exactly the same than the In route-maps from the original configuration of RA (PEER-B-IN and PEER-C-IN), only the name is different.

The same could have been done to create the Export policy for RA (route-map RSCLIENT-A-EXPORT), but in this case the original Out route-maps where so simple that we decided not to use the *call WORD* commands, and we integrated all in a single route-map (RSCLIENT-A-EXPORT).

The Import and Export policies for RB and RC are not shown, but the process would be identical.

Further considerations about Import and Export route-maps

The current version of the route server patch only allows to specify a route-map for import and export policies, while in a standard BGP speaker apart from route-maps there are other tools for performing input and output filtering (access-lists, community-lists, ...). But this does not represent any limitation, as all kinds of filters can be included in import/export route-maps. For example suppose that in the non-route-server scenario peer RA had the following filters configured for input from peer B:

```
neighbor 2001:0DB8::B prefix-list LIST-1 in
neighbor 2001:0DB8::B filter-list LIST-2 in
neighbor 2001:0DB8::B route-map PEER-B-IN in
...
...
route-map PEER-B-IN permit 10
  match ipv6 address prefix-list COMMON-PREFIXES
  set local-preference 100
route-map PEER-B-IN permit 20
  match ipv6 address prefix-list PEER-B-PREFIXES
  set community 65001:11111
```

It is possible to write a single route-map which is equivalent to the three filters (the community-list, the prefix-list and the route-map). That route-map can then be used inside the Import policy in the route server. Lets see how to do it:

```
neighbor 2001:0DB8::A route-map RSCLIENT-A-IMPORT in
...
!
...
route-map RSCLIENT-A-IMPORT permit 10
  match peer 2001:0DB8::B
  call A-IMPORT-FROM-B
...
...
!
route-map A-IMPORT-FROM-B permit 1
  match ipv6 address prefix-list LIST-1
  match as-path LIST-2
```

(continues on next page)

(continued from previous page)

```
  on-match goto 10
route-map A-IMPORT-FROM-B deny 2
route-map A-IMPORT-FROM-B permit 10
  match ipv6 address prefix-list COMMON-PREFIXES
  set local-preference 100
route-map A-IMPORT-FROM-B permit 20
  match ipv6 address prefix-list PEER-B-PREFIXES
  set community 65001:11111
!
...
...
```

The route-map A-IMPORT-FROM-B is equivalent to the three filters (LIST-1, LIST-2 and PEER-B-IN). The first entry of route-map A-IMPORT-FROM-B (sequence number 1) matches if and only if both the prefix-list LIST-1 and the filter-list LIST-2 match. If that happens, due to the 'on-match goto 10' statement the next route-map entry to be processed will be number 10, and as of that point route-map A-IMPORT-FROM-B is identical to PEER-B-IN. If the first entry does not match, *on-match goto 10*' will be ignored and the next processed entry will be number 2, which will deny the route.

Thus, the result is the same that with the three original filters, i.e., if either LIST-1 or LIST-2 rejects the route, it does not reach the route-map PEER-B-IN. In case both LIST-1 and LIST-2 accept the route, it passes to PEER-B-IN, which can reject, accept or modify the route.

Weighted ECMP using BGP link bandwidth

Overview

In normal equal cost multipath (ECMP), the route to a destination has multiple next hops and traffic is expected to be equally distributed across these next hops. In practice, flow-based hashing is used so that all traffic associated with a particular flow uses the same next hop, and by extension, the same path across the network.

Weighted ECMP using BGP link bandwidth introduces support for network-wide unequal cost multipathing (UCMP) to an IP destination. The unequal cost load balancing is implemented by the forwarding plane based on the weights associated with the next hops of the IP prefix. These weights are computed based on the bandwidths of the corresponding multipaths which are encoded in the `BGP link bandwidth extended community` as specified in [**?**]. Exchange of an appropriate BGP link bandwidth value for a prefix across the network results in network-wide unequal cost multipathing.

One of the primary use cases of this capability is in the data center when a service (represented by its anycast IP) has an unequal set of resources across the regions (e.g., PODs) of the data center and the network itself provides the load balancing function instead of an external load balancer. Refer to [**?**] and **RFC 7938** for details on this use case. This use case is applicable in a pure L3 network as well as in a EVPN network.

The traditional use case for BGP link bandwidth to load balance traffic to the exit routers in the AS based on the bandwidth of their external eBGP peering links is also supported.

Design Principles

Next hop weight computation and usage

As described, in UCMP, there is a weight associated with each next hop of an IP prefix, and traffic is expected to be distributed across the next hops in proportion to their weight. The weight of a next hop is a simple factoring of the bandwidth of the corresponding path against the total bandwidth of all multipaths, mapped to the range 1 to 100. What happens if not all the paths in the multipath set have link bandwidth associated with them? In such a case, in adherence to [**?**], the behavior reverts to standard ECMP among all the multipaths, with the link bandwidth being effectively ignored.

Note that there is no change to either the BGP best path selection algorithm or to the multipath computation algorithm; the mapping of link bandwidth to weight happens at the time of installation of the route in the RIB.

If data forwarding is implemented by means of the Linux kernel, the next hop's weight is used in the hash calculation. The kernel uses the Hash threshold algorithm and use of the next hop weight is built into it; next hops need not be expanded to achieve UCMP. UCMP for IPv4 is available in older Linux kernels too, while UCMP for IPv6 is available from the 4.16 kernel onwards.

If data forwarding is realized in hardware, common implementations expand the next hops (i.e., they are repeated) in the ECMP container in proportion to their weight. For example, if the weights associated with 3 next hops for a particular route are 50, 25 and 25 and the ECMP container has a size of 16 next hops, the first next hop will be repeated 8 times and the other 2 next hops repeated 4 times each. Other implementations are also possible.

Unequal cost multipath across a network

For the use cases listed above, it is not sufficient to support UCMP on just one router (e.g., egress router), or individually, on multiple routers; UCMP must be deployed across the entire network. This is achieved by employing the BGP link-bandwidth extended community.

At the router which originates the BGP link bandwidth, there has to be user configuration to trigger it, which is described below. Receiving routers would use the received link bandwidth from their downstream routers to determine the next hop weight as described in the earlier section. Further, if the received link bandwidth is a transitive attribute, it would be propagated to eBGP peers, with the additional change that if the next hop is set to oneself, the cumulative link bandwidth of all downstream paths is propagated to other routers. In this manner, the entire network will know how to distribute traffic to an anycast service across the network.

The BGP link-bandwidth extended community is encoded in bytes-per-second. In the use case where UCMP must be based on the number of paths, a reference bandwidth of 1 Mbps is used. So, for example, if there are 4 equal cost paths to an anycast IP, the encoded bandwidth in the extended community will be 500,000. The actual value itself doesn't matter as long as all routers originating the link-bandwidth are doing it in the same way.

Configuration Guide

The configuration for weighted ECMP using BGP link bandwidth requires one essential step - using a route-map to inject the link bandwidth extended community. An additional option is provided to control the processing of received link bandwidth.

Injecting link bandwidth into the network

At the "entry point" router that is injecting the prefix to which weighted load balancing must be performed, a route-map must be configured to attach the link bandwidth extended community.

For the use case of providing weighted load balancing for an anycast service, this configuration will typically need to be applied at the TOR or Leaf router that is connected to servers which provide the anycast service and the bandwidth would be based on the number of multipaths for the destination.

For the use case of load balancing to the exit router, the exit router should be configured with the route map specifying the a bandwidth value that corresponds to the bandwidth of the link connecting to its eBGP peer in the adjoining AS. In addition, the link bandwidth extended community must be explicitly configured to be non-transitive.

The complete syntax of the route-map set command can be found at *BGP Extended Communities in Route Map*

This route-map is supported only at two attachment points: (a) the outbound route-map attached to a peer or peer-group, per address-family (b) the EVPN advertise route-map used to inject IPv4 or IPv6 unicast routes into EVPN as type-5 routes.

Since the link bandwidth origination is done by using a route-map, it can be constrained to certain prefixes (e.g., only for anycast services) or it can be generated for all prefixes. Further, when the route-map is used in the neighbor context, the link bandwidth usage can be constrained to certain peers only.

A sample configuration is shown below and illustrates link bandwidth advertisement towards the "SPINE" peer-group for anycast IPs in the range 192.168.x.x

```
ip prefix-list anycast_ip seq 10 permit 192.168.0.0/16 le 32
route-map anycast_ip permit 10
 match ip address prefix-list anycast_ip
 set extcommunity bandwidth num-multipaths
route-map anycast_ip permit 20
!
router bgp 65001
 neighbor SPINE peer-group
 neighbor SPINE remote-as external
 neighbor 172.16.35.1 peer-group SPINE
 neighbor 172.16.36.1 peer-group SPINE
 !
 address-family ipv4 unicast
  network 110.0.0.1/32
  network 192.168.44.1/32
  neighbor SPINE route-map anycast_ip out
 exit-address-family
!
```

Controlling link bandwidth processing on the receiver

There is no configuration necessary to process received link bandwidth and translate it into the weight associated with the corresponding next hop; that happens by default. If some of the multipaths do not have the link bandwidth extended community, the default behavior is to revert to normal ECMP as recommended in [**?**].

The operator can change these behaviors with the following configuration:

bgp bestpath bandwidth <ignore | skip-missing | default-weight-for-missing>

The different options imply behavior as follows:

- ignore: Ignore link bandwidth completely for route installation (i.e., do regular ECMP, not weighted)
- skip-missing: Skip paths without link bandwidth and do UCMP among the others (if at least some paths have link-bandwidth)
- default-weight-for-missing: Assign a low default weight (value 1) to paths not having link bandwidth

This configuration is per BGP instance similar to other BGP route-selection controls; it operates on both IPv4-unicast and IPv6-unicast routes in that instance. In an EVPN network, this configuration (if required) should be implemented in the tenant VRF and is again applicable for IPv4-unicast and IPv6-unicast, including the ones sourced from EVPN type-5 routes.

A sample snippet of FRR configuration on a receiver to skip paths without link bandwidth and do weighted ECMP among the other paths (if some of them have link bandwidth) is as shown below.

```
router bgp 65021
 bgp bestpath as-path multipath-relax
 bgp bestpath bandwidth skip-missing
 neighbor LEAF peer-group
 neighbor LEAF remote-as external
 neighbor 172.16.35.2 peer-group LEAF
 neighbor 172.16.36.2 peer-group LEAF
 !
 address-family ipv4 unicast
  network 130.0.0.1/32
 exit-address-family
!
```

Stopping the propagation of the link bandwidth outside a domain

The link bandwidth extended community will get automatically propagated with the prefix to EBGP peers, if it is encoded as a transitive attribute by the originator. If this propagation has to be stopped outside of a particular domain (e.g., stopped from being propagated to routers outside of the data center core network), the mechanism available is to disable the advertisement of all BGP extended communities on the specific peering/s. In other words, the propagation cannot be blocked just for the link bandwidth extended community. The configuration to disable all extended communities can be applied to a peer or peer-group (per address-family).

Of course, the other common way to stop the propagation of the link bandwidth outside the domain is to block the prefixes themselves from being advertised and possibly, announce only an aggregate route. This would be quite common in a EVPN network.

BGP link bandwidth and UCMP monitoring & troubleshooting

Existing operational commands to display the BGP routing table for a specific prefix will show the link bandwidth extended community also, if present.

An example of an IPv4-unicast route received with the link bandwidth attribute from two peers is shown below:

```
CLI# show bgp ipv4 unicast 192.168.10.1/32
BGP routing table entry for 192.168.10.1/32
Paths: (2 available, best #2, table default)
  Advertised to non peer-group peers:
  l1(swp1) l2(swp2) l3(swp3) l4(swp4)
  65002
```

(continues on next page)

(continued from previous page)

```
    fe80::202:ff:fe00:1b from l2(swp2) (110.0.0.2)
    (fe80::202:ff:fe00:1b) (used)
      Origin IGP, metric 0, valid, external, multipath, bestpath-from-AS 65002
      Extended Community: LB:65002:125000000 (1000.000 Mbps)
      Last update: Thu Feb 20 18:34:16 2020

  65001
    fe80::202:ff:fe00:15 from l1(swp1) (110.0.0.1)
    (fe80::202:ff:fe00:15) (used)
      Origin IGP, metric 0, valid, external, multipath, bestpath-from-AS 65001, best␣
↪(Older Path)
      Extended Community: LB:65001:62500000 (500.000 Mbps)
      Last update: Thu Feb 20 18:22:34 2020
```

The weights associated with the next hops of a route can be seen by querying the RIB for a specific route.

For example, the next hop weights corresponding to the link bandwidths in the above example is illustrated below:

```
spine1# show ip route 192.168.10.1/32
Routing entry for 192.168.10.1/32
  Known via "bgp", distance 20, metric 0, best
  Last update 00:00:32 ago
  * fe80::202:ff:fe00:1b, via swp2, weight 66
  * fe80::202:ff:fe00:15, via swp1, weight 33
```

For troubleshooting, existing debug logs `debug bgp updates`, `debug bgp bestpath <prefix>`, `debug bgp zebra` and `debug zebra kernel` can be used.

A debug log snippet when `debug bgp zebra` is enabled and a route is installed by BGP in the RIB with next hop weights is shown below:

```
2020-02-29T06:26:19.927754+00:00 leaf1 bgpd[5459]: bgp_zebra_announce: p=192.168.150.1/
↪32, bgp_is_valid_label: 0
2020-02-29T06:26:19.928096+00:00 leaf1 bgpd[5459]: Tx route add VRF 33 192.168.150.1/32␣
↪metric 0 tag 0 count 2
2020-02-29T06:26:19.928289+00:00 leaf1 bgpd[5459]:   nhop [1]: 110.0.0.6 if 35 VRF 33 wt␣
↪50   RMAC 0a:11:2f:7d:35:20
2020-02-29T06:26:19.928479+00:00 leaf1 bgpd[5459]:   nhop [2]: 110.0.0.5 if 35 VRF 33 wt␣
↪50   RMAC 32:1e:32:a3:6c:bf
2020-02-29T06:26:19.928668+00:00 leaf1 bgpd[5459]: bgp_zebra_announce: 192.168.150.1/32:␣
↪announcing to zebra (recursion NOT set)
```

References

BGP fast-convergence support

Whenever BGP peer address becomes unreachable we must bring down the BGP session immediately. Currently only single-hop EBGP sessions are brought down immediately.IBGP and multi-hop EBGP sessions wait for hold-timer expiry to bring down the sessions.

This new configuration option helps user to teardown BGP sessions immediately whenever peer becomes unreachable.

bgp fast-convergence

This configuration is available at the bgp level. When enabled, configuration is applied to all the neighbors configured in that bgp instance.

```
router bgp 64496
 neighbor 10.0.0.2 remote-as 64496
 neighbor fd00::2 remote-as 64496
 bgp fast-convergence
!
address-family ipv4 unicast
 redistribute static
exit-address-family
!
address-family ipv6 unicast
 neighbor fd00::2 activate
exit-address-family
```

1.5.3 LDP

The *ldpd* daemon is a standardised protocol that permits exchanging MPLS label information between MPLS devices. The LDP protocol creates peering between devices, so as to exchange that label information. This information is stored in MPLS table of *zebra*, and it injects that MPLS information in the underlying system (Linux kernel or OpenBSD system for instance). *ldpd* provides necessary options to create a Layer 2 VPN across MPLS network. For instance, it is possible to interconnect several sites that share the same broadcast domain.

FRR implements LDP as described in **RFC 5036**; other LDP standard are the following ones: **RFC 6720**, **RFC 6667**, **RFC 5919**, **RFC 5561**, **RFC 7552**, **RFC 4447**. Because MPLS is already available, FRR also supports **RFC 3031**.

Understanding LDP principles

Let's first introduce some definitions that permit understand better the LDP protocol:

- *LSR* : Labeled Switch Router. Networking devices handling labels used to forward traffic between and through them.
- ***LER***
  [Labeled Edge Router. A Labeled edge router is located at the edge of] an MPLS network, generally between an IP network and an MPLS network.

LDP aims at sharing label information across devices. It tries to establish peering with remote LDP capable devices, first by discovering using UDP port 646 , then by peering using TCP port 646. Once the TCP session is established, the label information is shared, through label advertisements.

There are different methods to send label advertisement modes. The implementation actually supports the following : Liberal Label Retention + Downstream Unsolicited + Independent Control. The other advertising modes are depicted below, and compared with the current implementation.

- Liberal label retention versus conservative mode In liberal mode, every label sent by every LSR is stored in the MPLS table. In conservative mode, only the label that was sent by the best next hop (determined by the IGP metric) for that particular FEC is stored in the MPLS table.
- Independent LSP Control versus ordered LSP Control MPLS has two ways of binding labels to FEC's; either through ordered LSP control, or independent LSP control. Ordered LSP control only binds a label to a FEC if it is the egress LSR, or the router received a label binding for a FEC from the next hop router. In this mode, an MPLS router will create a label binding for each FEC and distribute it to its neighbors so long as he has a entry

in the RIB for the destination. In the other mode, label bindings are made without any dependencies on another router advertising a label for a particular FEC. Each router makes it own independent decision to create a label for each FEC. By default IOS uses Independent LSP Control, while Juniper implements the Ordered Control. Both modes are interoperable, the difference is that Ordered Control prevent blackholing during the LDP convergence process, at cost of slowing down the convergence itself

- unsolicited downstream versus downstream on demand Downstream on demand label distribution is where an LSR must explicitly request that a label be sent from its downstream router for a particular FEC. Unsolicited label distribution is where a label is sent from the downstream router without the original router requesting it.

LDP Configuration

mpls ldp

Enable or disable LDP daemon

router-id A.B.C.D

The following command located under MPLS router node configures the MPLS router-id of the local device.

ordered-control

Configure LDP Ordered Label Distribution Control.

address-family [ipv4 | ipv6]

Configure LDP for IPv4 or IPv6 address-family. Located under MPLS route node, this subnode permits configuring the LDP neighbors.

interface IFACE

Located under MPLS address-family node, use this command to enable or disable LDP discovery per interface. IFACE stands for the interface name where LDP is enabled. By default it is disabled. Once this command executed, the address-family interface node is configured.

discovery transport-address A.B.C.D | A:B::C:D

Located under mpls address-family interface node, use this command to set the IPv4 or IPv6 transport-address used by the LDP protocol to talk on this interface.

ttl-security disable

Located under the LDP address-family node, use this command to disable the GTSM procedures described in RFC 6720 (for the IPv4 address-family) and RFC 7552 (for the IPv6 address-family).

Since GTSM is mandatory for LDPv6, the only effect of disabling GTSM for the IPv6 address-family is that *ldpd* will not discard packets with a hop limit below 255. This may be necessary to interoperate with older implementations. Outgoing packets will still be sent using a hop limit of 255 for maximum compatibility.

If GTSM is enabled, multi-hop neighbors should have either GTSM disabled individually or configured with an appropriate ttl-security hops distance.

neighbor A.B.C.D password PASSWORD

The following command located under MPLS router node configures the router of a LDP device. This device, if found, will have to comply with the configured password. PASSWORD is a clear text password wit its digest sent through the network.

neighbor A.B.C.D holdtime HOLDTIME

The following command located under MPLS router node configures the holdtime value in seconds of the LDP neighbor ID. Configuring it triggers a keepalive mechanism. That value can be configured between 15 and 65535 seconds. After this time of non response, the LDP established session will be considered as set to down. By default, no holdtime is configured for the LDP devices.

neighbor A.B.C.D ttl-security disable

Located under the MPLS LDP node, use this command to override the global configuration and enable/disable GTSM for the specified neighbor.

neighbor A.B.C.D ttl-security hops (1-254)

Located under the MPLS LDP node, use this command to set the maximum number of hops the specified neighbor may be away. When GTSM is enabled for this neighbor, incoming packets are required to have a TTL/hop limit of 256 minus this value, ensuring they have not passed through more than the expected number of hops. The default value is 1.

discovery hello holdtime HOLDTIME

discovery hello interval INTERVAL

INTERVAL value ranges from 1 to 65535 seconds. Default value is 5 seconds. This is the value between each hello timer message sent. HOLDTIME value ranges from 1 to 65535 seconds. Default value is 15 seconds. That value is added as a TLV in the LDP messages.

dual-stack transport-connection prefer ipv4

When *ldpd* is configured for dual-stack operation, the transport connection preference is IPv6 by default (as specified by **RFC 7552**). On such circumstances, *ldpd* will refuse to establish TCP connections over IPv4. You can use above command to change the transport connection preference to IPv4. In this case, it will be possible to distribute label mappings for IPv6 FECs over TCPv4 connections.

Show LDP Information

These commands dump various parts of *ldpd*.

show mpls ldp neighbor [A.B.C.D]

This command dumps the various neighbors discovered. Below example shows that local machine has an operation neighbor with ID set to 1.1.1.1.

```
west-vm# show mpls ldp neighbor
AF   ID              State       Remote Address    Uptime
ipv4 1.1.1.1         OPERATIONAL 1.1.1.1         00:01:37
west-vm#
```

show mpls ldp neighbor [A.B.C.D] capabilities

show mpls ldp neighbor [A.B.C.D] detail

Above commands dump other neighbor information.

show mpls ldp discovery [detail]

show mpls ldp ipv4 discovery [detail]

show mpls ldp ipv6 discovery [detail]

Above commands dump discovery information.

show mpls ldp ipv4 interface

show mpls ldp ipv6 interface

Above command dumps the IPv4 or IPv6 interface per where LDP is enabled. Below output illustrates what is dumped for IPv4.

```
west-vm# show mpls ldp ipv4 interface
AF   Interface   State  Uptime   Hello Timers  ac
ipv4 eth1       ACTIVE 00:08:35 5/15          0
ipv4 eth3       ACTIVE 00:08:35 5/15          1
```

show mpls ldp ipv4|ipv6 binding

Above command dumps the binding obtained through MPLS exchanges with LDP.

```
west-vm# show mpls ldp ipv4 binding
AF   Destination          Nexthop         Local Label Remote Label  In Use
ipv4 1.1.1.1/32           1.1.1.1         16          imp-null         yes
ipv4 2.2.2.2/32           1.1.1.1         imp-null    16                no
ipv4 10.0.2.0/24          1.1.1.1         imp-null    imp-null          no
ipv4 10.115.0.0/24        1.1.1.1         imp-null    17                no
ipv4 10.135.0.0/24        1.1.1.1         imp-null    imp-null          no
ipv4 10.200.0.0/24        1.1.1.1         17          imp-null         yes
west-vm#
```

LDP debugging commands

debug mpls ldp KIND

Enable or disable debugging messages of a given kind. `KIND` can be one of:

- `discovery`
- `errors`
- `event`
- `labels`
- `messages`
- `zebra`

LDP Example Configuration

Below configuration gives a typical MPLS configuration of a device located in a MPLS backbone. LDP is enabled on two interfaces and will attempt to peer with two neighbors with router-id set to either 1.1.1.1 or 3.3.3.3.

```
mpls ldp
 router-id 2.2.2.2
 neighbor 1.1.1.1 password test
 neighbor 3.3.3.3 password test
 !
 address-family ipv4
  discovery transport-address 2.2.2.2
  !
  interface eth1
  !
  interface eth3
  !
 exit-address-family
 !
```

Deploying LDP across a backbone generally is done in a full mesh configuration topology. LDP is typically deployed with an IGP like OSPF, that helps discover the remote IPs. Below example is an OSPF configuration extract that goes with LDP configuration

```
router ospf
 ospf router-id 2.2.2.2
  network 0.0.0.0/0 area 0
 !
```

Below output shows the routing entry on the LER side. The OSPF routing entry (10.200.0.0) is associated with Label entry (17), and shows that MPLS push action that traffic to that destination will be applied.

```
north-vm# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP,
       F - PBR,
       > - selected route, * - FIB route

O>* 1.1.1.1/32 [110/120] via 10.115.0.1, eth2, label 16, 00:00:15
O>* 2.2.2.2/32 [110/20] via 10.115.0.1, eth2, label implicit-null, 00:00:15
O   3.3.3.3/32 [110/10] via 0.0.0.0, loopback1 onlink, 00:01:19
C>* 3.3.3.3/32 is directly connected, loopback1, 00:01:29
O>* 10.0.2.0/24 [110/11] via 10.115.0.1, eth2, label implicit-null, 00:00:15
O   10.100.0.0/24 [110/10] is directly connected, eth1, 00:00:32
C>* 10.100.0.0/24 is directly connected, eth1, 00:00:32
O   10.115.0.0/24 [110/10] is directly connected, eth2, 00:00:25
C>* 10.115.0.0/24 is directly connected, eth2, 00:00:32
O>* 10.135.0.0/24 [110/110] via 10.115.0.1, eth2, label implicit-null, 00:00:15
O>* 10.200.0.0/24 [110/210] via 10.115.0.1, eth2, label 17, 00:00:15
north-vm#
```

Additional example demonstrating use of some miscellaneous config options:

```
interface ge0
!
interface ge1
!
interface loopback0
!
mpls ldp
 dual-stack cisco-interop
 neighbor 10.0.1.5 password opensourcerouting
 neighbor 172.16.0.1 password opensourcerouting
 !
 address-family ipv4
  discovery transport-address 10.0.1.1
  label local advertise explicit-null
  !
  interface ge0
  !
  interface ge1
  !
 !
```

(continues on next page)

(continued from previous page)

```
 address-family ipv6
  discovery transport-address 2001:db8::1
  !
  interface ge1
  !
 !
!
l2vpn ENG type vpls
 !
 member pseudowire mpls-tunnel1
  neighbor lsr-id 1.1.1.1
  pw-id 100
 !
!
```

1.5.4 ISIS

ISIS (Intermediate System to Intermediate System) is a routing protocol which is described in *ISO10589*, **RFC 1195**, **RFC 5308**. ISIS is an IGP (Interior Gateway Protocol). Compared with RIP, ISIS can provide scalable network support and faster convergence times like OSPF. ISIS is widely used in large networks such as ISP (Internet Service Provider) and carrier backbone networks.

ISIS router

To start the ISIS process you have to specify the ISIS router. As of this writing, *isisd* does not support multiple ISIS processes.

router isis WORD [vrf NAME]

Enable or disable the ISIS process by specifying the ISIS domain with 'WORD'. *isisd* does not yet support multiple ISIS processes but you must specify the name of ISIS process. The ISIS process name 'WORD' is then used for interface (see command `ip router isis WORD`).

net XX.XXXX. ... .XXX.XX

Set/Unset network entity title (NET) provided in ISO format.

hostname dynamic

Enable support for dynamic hostname.

area-password [clear | md5] <password>

domain-password [clear | md5] <password>

Configure the authentication password for an area, respectively a domain, as clear text or md5 one.

attached-bit [receive ignore | send]

Set attached bit for inter-area traffic:

- receive If LSP received with attached bit set, create default route to neighbor
- send If L1|L2 router, set attached bit in LSP sent to L1 router

log-adjacency-changes

Log changes in adjacency state.

`log-pdu-drops`
: Log any dropped PDUs.

`metric-style [narrow | transition | wide]`
: Set old-style (ISO 10589) or new-style packet formats:

  - narrow Use old style of TLVs with narrow metric
  - transition Send and accept both styles of TLVs during transition
  - wide Use new style of TLVs to carry wider metric. SoodarOS uses this as a default value

`advertise-high-metrics`
: Advertise high metric value on all interfaces to gracefully shift traffic off the router. Reference: **RFC 3277**

  For narrow metrics, the high metric value is 63; for wide metrics, 16777215; for transition metrics, 62.

`set-overload-bit`
: Set overload bit to avoid any transit traffic.

`set-overload-bit on-startup (0-86400)`
: Set overload bit on startup for the specified duration, in seconds. Reference: **RFC 3277**

`purge-originator`
: Enable or disable **RFC 6232** purge originator identification.

`lsp-mtu (128-4352)`
: Configure the maximum size of generated LSPs, in bytes.

`advertise-passive-only`
: Advertise prefixes of passive interfaces only.

ISIS Timer

`lsp-gen-interval [level-1 | level-2] (1-120)`
: Set minimum interval in seconds between regenerating same LSP, globally, for an area (level-1) or a domain (level-2).

`lsp-refresh-interval [level-1 | level-2] (1-65235)`
: Set LSP refresh interval in seconds, globally, for an area (level-1) or a domain (level-2).

`max-lsp-lifetime [level-1 | level-2] (360-65535)`
: Set LSP maximum LSP lifetime in seconds, globally, for an area (level-1) or a domain (level-2).

`spf-interval [level-1 | level-2] (1-120)`
: Set minimum interval between consecutive SPF calculations in seconds.

ISIS region

`is-type [level-1 | level-1-2 | level-2-only]`
: Define the ISIS router behavior:

  - level-1 Act as a station router only
  - level-1-2 Act as both a station router and an area router
  - level-2-only Act as an area router only

ISIS interface

<ip|ipv6> router isis WORD

Activate ISIS adjacency on this interface. Note that the name of ISIS instance must be the same as the one used to configure the ISIS process (see command `router isis WORD`). To enable IPv4, issue `ip router isis WORD`; to enable IPv6, issue `ipv6 router isis WORD`.

isis circuit-type [level-1 | level-1-2 | level-2]

Configure circuit type for interface:

- level-1 Level-1 only adjacencies are formed
- level-1-2 Level-1-2 adjacencies are formed
- level-2-only Level-2 only adjacencies are formed

isis csnp-interval (1-600) [level-1 | level-2]

Set CSNP interval in seconds globally, for an area (level-1) or a domain (level-2).

isis hello padding

Add padding to IS-IS hello packets.

isis hello padding during-adjacency-formation

Add padding to IS-IS hello packets during adjacency formation only.

isis hello-interval (1-600) [level-1 | level-2]

Set Hello interval in seconds globally, for an area (level-1) or a domain (level-2).

isis hello-multiplier (2-100) [level-1 | level-2]

Set multiplier for Hello holding time globally, for an area (level-1) or a domain (level-2).

isis metric [(0-255) | (0-16777215)] [level-1 | level-2]

Set default metric value globally, for an area (level-1) or a domain (level-2). Max value depend if metric support narrow or wide value (see command *metric-style [narrow | transition | wide]*).

isis network point-to-point

Set network type to 'Point-to-Point' (broadcast by default).

isis passive

Configure the passive mode for this interface.

isis password [clear | md5] <password>

Configure the authentication password (clear or encoded text) for the interface.

isis priority (0-127) [level-1 | level-2]

Set priority for Designated Router election, globally, for the area (level-1) or the domain (level-2).

isis psnp-interval (1-120) [level-1 | level-2]

Set PSNP interval in seconds globally, for an area (level-1) or a domain (level-2).

isis three-way-handshake

Enable or disable **RFC 5303** Three-Way Handshake for P2P adjacencies. Three-Way Handshake is enabled by default.

Showing ISIS information

show isis [vrf <NAME|all>] summary [json]
Show summary information about ISIS.

show isis hostname
Show information about ISIS node.

show isis [vrf <NAME|all>] interface [detail] [IFNAME] [json]
Show state and configuration of ISIS specified interface, or all interfaces if no interface is given with or without details.

show isis [vrf <NAME|all>] neighbor [detail] [SYSTEMID] [json]
Show state and information of ISIS specified neighbor, or all neighbors if no system id is given with or without details.

show isis [vrf <NAME|all>] database [detail] [LSPID] [json]
Show the ISIS database globally, for a specific LSP id without or with details.

show isis topology [level-1|level-2] [algorithm (128-255)]
Show topology IS-IS paths to Intermediate Systems, globally, in area (level-1) or domain (level-2).

show isis route [level-1|level-2] [prefix-sid|backup] [algorithm (128-255)]
Show the ISIS routing table, as determined by the most recent SPF calculation.

Debugging ISIS

debug isis adj-packets
IS-IS Adjacency related packets.

debug isis checksum-errors
IS-IS LSP checksum errors.

debug isis events
IS-IS Events.

debug isis local-updates
IS-IS local update packets.

debug isis packet-dump
IS-IS packet dump.

debug isis protocol-errors
IS-IS LSP protocol errors.

debug isis route-events
IS-IS Route related events.

debug isis snp-packets
IS-IS CSNP/PSNP packets.

debug isis spf-events

debug isis spf-statistics

debug isis spf-triggers
IS-IS Shortest Path First Events, Timing and Statistic Data and triggering events.

debug isis update-packets

Update related packets.

show debugging isis

Print which ISIS debug level is activate.

ISIS Configuration Examples

A simple example, with MD5 authentication enabled:

```
!
interface eth0
 ip router isis FOO
 isis network point-to-point
 isis circuit-type level-2-only
!
router isis FOO
net 47.0023.0000.0000.0000.0000.0000.1900.0004.00
 metric-style wide
 is-type level-2-only
```

ISIS Vrf Configuration Examples

A simple vrf example:

```
!
interface ge0 vrf RED
 ip vrf forwarding RED
 ip router isis FOO vrf RED
 isis network point-to-point
 isis circuit-type level-2-only
!
router isis FOO vrf RED
 net 47.0023.0000.0000.0000.0000.0000.1900.0004.00
 metric-style wide
 is-type level-2-only
```

1.5.5 OSPFv2

OSPF (Open Shortest Path First) version 2 is a routing protocol which is described in **RFC 2328**. OSPF is an IGP. Compared with RIP, OSPF can provide scalable network support and faster convergence times. OSPF is widely used in large networks such as ISP backbone and enterprise networks.

OSPF Fundamentals

OSPF is, mostly, a link-state routing protocol. In contrast to *distance-vector* protocols, such as RIP or BGP, where routers describe available *paths* (i.e. routes) to each other, in *link-state* protocols routers instead describe the state of their links to their immediate neighbouring routers.

Each router describes their link-state information in a message known as an LSA (Link State Advertisement), which is then propagated through to all other routers in a link-state routing domain, by a process called *flooding*. Each router thus builds up an LSDB (Link State Database) of all the link-state messages. From this collection of LSAs in the LSDB, each router can then calculate the shortest path to any other router, based on some common metric, by using an algorithm such as Edsger Dijkstra's SPF (Shortest Path First) algorithm.

By describing connectivity of a network in this way, in terms of routers and links rather than in terms of the paths through a network, a link-state protocol can use less bandwidth and converge more quickly than other protocols. A link-state protocol need distribute only one link-state message throughout the link-state domain when a link on any single given router changes state, in order for all routers to reconverge on the best paths through the network. In contrast, distance vector protocols can require a progression of different path update messages from a series of different routers in order to converge.

The disadvantage to a link-state protocol is that the process of computing the best paths can be relatively intensive when compared to distance-vector protocols, in which near to no computation need be done other than (potentially) select between multiple routes. This overhead is mostly negligible for modern embedded CPUs, even for networks with thousands of nodes. The primary scaling overhead lies more in coping with the ever greater frequency of LSA updates as the size of a link-state area increases, in managing the LSDB and required flooding.

This section aims to give a distilled, but accurate, description of the more important workings of OSPF which an administrator may need to know to be able best configure and trouble-shoot OSPF.

OSPF Mechanisms

OSPF defines a range of mechanisms, concerned with detecting, describing and propagating state through a network. These mechanisms will nearly all be covered in greater detail further on. They may be broadly classed as:

The Hello Protocol

The OSPF Hello protocol allows OSPF to quickly detect changes in two-way reachability between routers on a link. OSPF can additionally avail of other sources of reachability information, such as link-state information provided by hardware, or through dedicated reachability protocols such as BFD.

OSPF also uses the Hello protocol to propagate certain state between routers sharing a link, for example:

- Hello protocol configured state, such as the dead-interval.
- Router priority, for DR/BDR election.
- DR/BDR election results.
- Any optional capabilities supported by each router.

The Hello protocol is comparatively trivial and will not be explored in more detail.

LSAs

At the heart of OSPF are LSA messages. Despite the name, some LSA s do not, strictly speaking, describe link-state information. Common LSA s describe information such as:

- Routers, in terms of their links.
- Networks, in terms of attached routers.
- Routes, external to a link-state domain:

  External Routes
  : Routes entirely external to OSPF. Routers originating such routes are known as ASBR (Autonomous-System Border Router) routers.

  Summary Routes
  : Routes which summarise routing information relating to OSPF areas external to the OSPF link-state area at hand, originated by ABR (Area Boundary Router) routers.

LSA Flooding

OSPF defines several related mechanisms, used to manage synchronisation of LSDB s between neighbours as neighbours form adjacencies and the propagation, or *flooding* of new or updated LSA s.

Areas

OSPF provides for the protocol to be broken up into multiple smaller and independent link-state areas. Each area must be connected to a common backbone area by an ABR. These ABR routers are responsible for summarising the link-state routing information of an area into *Summary LSAs*, possibly in a condensed (i.e. aggregated) form, and then originating these summaries into all other areas the ABR is connected to.

Note that only summaries and external routes are passed between areas. As these describe *paths*, rather than any router link-states, routing between areas hence is by *distance-vector*, **not** link-state.

OSPF LSAs

The core objects in OSPF are LSA s. Everything else in OSPF revolves around detecting what to describe in LSAs, when to update them, how to flood them throughout a network and how to calculate routes from them.

There are a variety of different LSA s, for purposes such as describing actual link-state information, describing paths (i.e. routes), describing bandwidth usage of links for TE (Traffic Engineering) purposes, and even arbitrary data by way of *Opaque* LSA s.

LSA Header

All LSAs share a common header with the following information:

- Type

  Different types of LSA s describe different things in OSPF. Types include:

  - Router LSA
  - Network LSA
  - Network Summary LSA

  - Router Summary LSA
  - AS-External LSA

  The specifics of the different types of LSA are examined below.

- Advertising Router

  The Router ID of the router originating the LSA.

See also:

ospf router-id A.B.C.D.

- LSA ID

  The ID of the LSA, which is typically derived in some way from the information the LSA describes, e.g. a Router LSA uses the Router ID as the LSA ID, a Network LSA will have the IP address of the DR as its LSA ID.

  The combination of the Type, ID and Advertising Router ID must uniquely identify the LSA. There can however be multiple instances of an LSA with the same Type, LSA ID and Advertising Router ID, see *sequence number*.

- Age

  A number to allow stale LSA s to, eventually, be purged by routers from their LSDB s.

  The value nominally is one of seconds. An age of 3600, i.e. 1 hour, is called the *MaxAge*. MaxAge LSAs are ignored in routing calculations. LSAs must be periodically refreshed by their Advertising Router before reaching MaxAge if they are to remain valid.

  Routers may deliberately flood LSAs with the age artificially set to 3600 to indicate an LSA is no longer valid. This is called *flushing* of an LSA.

  It is not abnormal to see stale LSAs in the LSDB, this can occur where a router has shutdown without flushing its LSA(s), e.g. where it has become disconnected from the network. Such LSAs do little harm.

- Sequence Number

  A number used to distinguish newer instances of an LSA from older instances.

Link-State LSAs

Of all the various kinds of LSA s, just two types comprise the actual link-state part of OSPF, Router LSA s and Network LSA s. These LSA types are absolutely core to the protocol.

Instances of these LSAs are specific to the link-state area in which they are originated. Routes calculated from these two LSA types are called *intra-area routes*.

- Router LSA

  Each OSPF Router must originate a router LSA to describe itself. In it, the router lists each of its OSPF enabled interfaces, for the given link-state area, in terms of:

  Cost
  : The output cost of that interface, scaled inversely to some commonly known reference value, *auto-cost reference-bandwidth (1-4294967)*.

  Link Type
  : Transit Network

    A link to a multi-access network, on which the router has at least one Full adjacency with another router.

PtP (Point-to-Point)
A link to a single remote router, with a Full adjacency. No DR (Designated Router) is elected on such links; no network LSA is originated for such a link.

Stub
A link with no adjacent neighbours, or a host route.

– Link ID and Data

These values depend on the Link Type:

Link Type	Link ID	Link Data
Transit	Link IP address of the DR	Interface IP address
Point-to-Point	Router ID of the remote router	Local interface IP address, or the IFINDEX (MIB-II interface index) for unnumbered links
Stub	IP address	Subnet Mask

Links on a router may be listed multiple times in the Router LSA, e.g. a PtP interface on which OSPF is enabled must *always* be described by a Stub link in the Router LSA, in addition to being listed as PtP link in the Router LSA if the adjacency with the remote router is Full.

Stub links may also be used as a way to describe links on which OSPF is *not* spoken, known as *passive interfaces*, see `ip ospf passive [A.B.C.D]`.

- Network LSA

On multi-access links (e.g. ethernets, certain kinds of ATM and X.25 configurations), routers elect a DR. The DR is responsible for originating a Network LSA, which helps reduce the information needed to describe multi-access networks with multiple routers attached. The DR also acts as a hub for the flooding of LSA s on that link, thus reducing flooding overheads.

The contents of the Network LSA describes the:

– Subnet Mask

As the LSA ID of a Network LSA must be the IP address of the DR, the Subnet Mask together with the LSA ID gives you the network address.

– Attached Routers

Each router fully-adjacent with the DR is listed in the LSA, by their Router-ID. This allows the corresponding Router LSA s to be easily retrieved from the LSDB.

Summary of Link State LSAs:

LSA Type	LSA ID	LSA Data Describes
Router LSA	Router ID	The OSPF enabled links of the router, within a specific link-state area.
Network LSA	The IP address of the DR for the network	The subnet mask of the network and the Router IDs of all routers on the network

With an LSDB composed of just these two types of LSA, it is possible to construct a directed graph of the connectivity between all routers and networks in a given OSPF link-state area. So, not surprisingly, when OSPF routers build updated routing tables, the first stage of SPF calculation concerns itself only with these two LSA types.

Link-State LSA Examples

The example below shows two LSA s, both originated by the same router (Router ID 192.168.0.49) and with the same LSA ID (192.168.0.49), but of different LSA types.

The first LSA being the router LSA describing 192.168.0.49's links: 2 links to multi-access networks with fully-adjacent neighbours (i.e. Transit links) and 1 being a Stub link (no adjacent neighbours).

The second LSA being a Network LSA, for which 192.168.0.49 is the DR, listing the Router IDs of 4 routers on that network which are fully adjacent with 192.168.0.49.

```
# show ip ospf database router 192.168.0.49

       OSPF Router with ID (192.168.0.53)

                Router Link States (Area 0.0.0.0)

  LS age: 38
  Options: 0x2  : *|-|-|-|-|-|E|*
  LS Flags: 0x6
  Flags: 0x2 : ASBR
  LS Type: router-LSA
  Link State ID: 192.168.0.49
  Advertising Router: 192.168.0.49
  LS Seq Number: 80000f90
  Checksum: 0x518b
  Length: 60
   Number of Links: 3

    Link connected to: a Transit Network
     (Link ID) Designated Router address: 192.168.1.3
     (Link Data) Router Interface address: 192.168.1.3
      Number of TOS metrics: 0
       TOS 0 Metric: 10

    Link connected to: a Transit Network
     (Link ID) Designated Router address: 192.168.0.49
     (Link Data) Router Interface address: 192.168.0.49
      Number of TOS metrics: 0
       TOS 0 Metric: 10

    Link connected to: Stub Network
     (Link ID) Net: 192.168.3.190
     (Link Data) Network Mask: 255.255.255.255
      Number of TOS metrics: 0
       TOS 0 Metric: 39063
# show ip ospf database network 192.168.0.49

       OSPF Router with ID (192.168.0.53)

                Net Link States (Area 0.0.0.0)

  LS age: 285
  Options: 0x2  : *|-|-|-|-|-|E|*
```

(continues on next page)

(continued from previous page)

```
  LS Flags: 0x6
  LS Type: network-LSA
  Link State ID: 192.168.0.49 (address of Designated Router)
  Advertising Router: 192.168.0.49
  LS Seq Number: 80000074
  Checksum: 0x0103
  Length: 40
  Network Mask: /29
        Attached Router: 192.168.0.49
        Attached Router: 192.168.0.52
        Attached Router: 192.168.0.53
        Attached Router: 192.168.0.54
```

Note that from one LSA, you can find the other. E.g. Given the Network-LSA you have a list of Router IDs on that network, from which you can then look up, in the local LSDB, the matching Router LSA. From that Router-LSA you may (potentially) find links to other Transit networks and Routers IDs which can be used to lookup the corresponding Router or Network LSA. And in that fashion, one can find all the Routers and Networks reachable from that starting LSA.

Given the Router LSA instead, you have the IP address of the DR of any attached transit links. Network LSAs will have that IP as their LSA ID, so you can then look up that Network LSA and from that find all the attached routers on that link, leading potentially to more links and Network and Router LSAs, etc. etc.

From just the above two LSA s, one can already see the following partial topology:

```
----------------------- Network: ......
            |           Designated Router IP: 192.168.1.3
            |
      IP: 192.168.1.3
       (transit link)
        (cost: 10)
   Router ID: 192.168.0.49(stub)---------- IP: 192.168.3.190/32
        (cost: 10)        (cost: 39063)
       (transit link)
      IP: 192.168.0.49
            |
            |
---------------------------- Network: 192.168.0.48/29
  |        |           |       Designated Router IP: 192.168.0.49
  |        |           |
  |        |     Router ID: 192.168.0.54
  |        |
  |   Router ID: 192.168.0.53
  |
Router ID: 192.168.0.52
```

Note the Router IDs, though they look like IP addresses and often are IP addresses, are not strictly speaking IP addresses, nor need they be reachable addresses (though, OSPF will calculate routes to Router IDs).

External LSAs

External, or "Type 5", LSA s describe routing information which is entirely external to OSPF, and is "injected" into OSPF. Such routing information may have come from another routing protocol, such as RIP or BGP, they may represent static routes or they may represent a default route.

An OSPF router which originates External LSA s is known as an ASBR. Unlike the link-state LSA s, and most other LSA s, which are flooded only within the area in which they originate, External LSA s are flooded through-out the OSPF network to all areas capable of carrying External LSA s (*Areas*).

Routes internal to OSPF (intra-area or inter-area) are always preferred over external routes.

The External LSA describes the following:

IP Network number
: The IP Network number of the route is described by the LSA ID field.

IP Network Mask
: The body of the External LSA describes the IP Network Mask of the route. This, together with the LSA ID, describes the prefix of the IP route concerned.

Metric
: The cost of the External Route. This cost may be an OSPF cost (also known as a "Type 1" metric), i.e. equivalent to the normal OSPF costs, or an externally derived cost ("Type 2" metric) which is not comparable to OSPF costs and always considered larger than any OSPF cost. Where there are both Type 1 and 2 External routes for a route, the Type 1 is always preferred.

Forwarding Address
: The address of the router to forward packets to for the route. This may be, and usually is, left as 0 to specify that the ASBR originating the External LSA should be used. There must be an internal OSPF route to the forwarding address, for the forwarding address to be usable.

Tag
: An arbitrary 4-bytes of data, not interpreted by OSPF, which may carry whatever information about the route which OSPF speakers desire.

AS External LSA Example

To illustrate, below is an example of an External LSA in the LSDB of an OSPF router. It describes a route to the IP prefix of 192.168.165.0/24, originated by the ASBR with Router-ID 192.168.0.49. The metric of 20 is external to OSPF. The forwarding address is 0, so the route should forward to the originating ASBR if selected.

```
# show ip ospf database external 192.168.165.0
  LS age: 995
  Options: 0x2  : *|-|-|-|-|-|E|*
  LS Flags: 0x9
  LS Type: AS-external-LSA
  Link State ID: 192.168.165.0 (External Network Number)
  Advertising Router: 192.168.0.49
  LS Seq Number: 800001d8
  Checksum: 0xea27
  Length: 36
  Network Mask: /24
        Metric Type: 2 (Larger than any link state path)
        TOS: 0
        Metric: 20
```

(continues on next page)

(continued from previous page)

```
       Forward Address: 0.0.0.0
       External Route Tag: 0
```

We can add this to our partial topology from above, which now looks like::

```
--------------------- Network: ......
         |            Designated Router IP: 192.168.1.3
         |
   IP: 192.168.1.3      /---- External route: 192.168.165.0/24
    (transit link)     /                Cost: 20 (External metric)
     (cost: 10)       /
Router ID: 192.168.0.49(stub)---------- IP: 192.168.3.190/32
     (cost: 10)        (cost: 39063)
    (transit link)
   IP: 192.168.0.49
         |
         |
------------------------------ Network: 192.168.0.48/29
  |        |           |       Designated Router IP: 192.168.0.49
  |        |           |
  |        |     Router ID: 192.168.0.54
  |        |
  |   Router ID: 192.168.0.53
  |
Router ID: 192.168.0.52
```

Summary LSAs

Summary LSAs are created by ABR s to summarise the destinations available within one area to other areas. These LSAs may describe IP networks, potentially in aggregated form, or ASBR routers.

Routers

To start OSPF process you have to specify the OSPF router.

router ospf [{(1-65535)|vrf NAME}]

Enable or disable the OSPF process.

Multiple instances don't support *vrf NAME*.

ospf router-id A.B.C.D

This sets the router-ID of the OSPF process. The router-ID may be an IP address of the router, but need not be - it can be any arbitrary 32bit number. However it MUST be unique within the entire OSPF domain to the OSPF speaker - bad things will happen if multiple OSPF speakers are configured with the same router-ID! If one is not specified then *ospfd* will obtain a router-ID automatically from *zebra*.

ospf abr-type TYPE

type can be cisco|ibm|shortcut|standard. The "Cisco" and "IBM" types are equivalent.

The OSPF standard for ABR behaviour does not allow an ABR to consider routes through non-backbone areas when its links to the backbone are down, even when there are other ABRs in attached non-backbone areas which still can reach the backbone - this restriction exists primarily to ensure routing-loops are avoided.

With the "Cisco" or "IBM" ABR type, the default in this release of FRR, this restriction is lifted, allowing an ABR to consider summaries learned from other ABRs through non-backbone areas, and hence route via non-backbone areas as a last resort when, and only when, backbone links are down.

Note that areas with fully-adjacent virtual-links are considered to be "transit capable" and can always be used to route backbone traffic, and hence are unaffected by this setting (*area A.B.C.D virtual-link A.B.C.D*).

More information regarding the behaviour controlled by this command can be found in **RFC 3509**, and *draft-ietf-ospf-shortcut-abr-02.txt*.

Quote: "Though the definition of the ABR in the OSPF specification does not require a router with multiple attached areas to have a backbone connection, it is actually necessary to provide successful routing to the inter-area and external destinations. If this requirement is not met, all traffic destined for the areas not connected to such an ABR or out of the OSPF domain, is dropped. This document describes alternative ABR behaviors implemented in Cisco and IBM routers."

ospf rfc1583compatibility

RFC 2328, the successor to **RFC 1583**, suggests according to section G.2 (changes) in section 16.4 a change to the path preference algorithm that prevents possible routing loops that were possible in the old version of OSPFv2. More specifically it demands that inter-area paths and intra-area backbone path are now of equal preference but still both preferred to external paths.

This command should NOT be set normally.

log-adjacency-changes [detail]

Configures ospfd to log changes in adjacency. With the optional detail argument, all changes in adjacency status are shown. Without detail, only changes to full or regressions are shown.

passive-interface default

Make all interfaces that belong to this router passive by default. For the description of passive interface look at *ip ospf passive [A.B.C.D]*. Per-interface configuration takes precedence over the default value.

timers throttle spf (0-600000) (0-600000) (0-600000)

This command sets the initial *delay*, the *initial-holdtime* and the *maximum-holdtime* between when SPF is calculated and the event which triggered the calculation. The times are specified in milliseconds and must be in the range of 0 to 600000 milliseconds.

The *delay* specifies the minimum amount of time to delay SPF calculation (hence it affects how long SPF calculation is delayed after an event which occurs outside of the holdtime of any previous SPF calculation, and also serves as a minimum holdtime).

Consecutive SPF calculations will always be separated by at least 'hold-time' milliseconds. The hold-time is adaptive and initially is set to the *initial-holdtime* configured with the above command. Events which occur within the holdtime of the previous SPF calculation will cause the holdtime to be increased by *initial-holdtime*, bounded by the *maximum-holdtime* configured with this command. If the adaptive hold-time elapses without any SPF-triggering event occurring then the current holdtime is reset to the *initial-holdtime*. The current holdtime can be viewed with `show ip ospf`, where it is expressed as a multiplier of the *initial-holdtime*.

```
router ospf
timers throttle spf 200 400 10000
```

In this example, the *delay* is set to 200ms, the initial holdtime is set to 400ms and the *maximum holdtime* to 10s. Hence there will always be at least 200ms between an event which requires SPF calculation and the actual SPF calculation. Further consecutive SPF calculations will always be separated by between 400ms to 10s, the hold-time increasing by 400ms each time an SPF-triggering event occurs within the hold-time of the previous SPF calculation.

This command supersedes the *timers spf* command in previous FRR releases.

max-metric router-lsa [on-startup (5-86400)|on-shutdown (5-100)]

max-metric router-lsa administrative

This enables **RFC 3137** support, where the OSPF process describes its transit links in its router-LSA as having infinite distance so that other routers will avoid calculating transit paths through the router while still being able to reach networks through the router.

This support may be enabled administratively (and indefinitely) or conditionally. Conditional enabling of max-metric router-lsas can be for a period of seconds after startup and/or for a period of seconds prior to shutdown.

Enabling this for a period after startup allows OSPF to converge fully first without affecting any existing routes used by other routers, while still allowing any connected stub links and/or redistributed routes to be reachable. Enabling this for a period of time in advance of shutdown allows the router to gracefully excuse itself from the OSPF domain.

Enabling this feature administratively allows for administrative intervention for whatever reason, for an indefinite period of time. Note that if the configuration is written to file, this administrative form of the stub-router command will also be written to file. If *ospfd* is restarted later, the command will then take effect until manually deconfigured.

Configured state of this feature as well as current status, such as the number of second remaining till on-startup or on-shutdown ends, can be viewed with the `show ip ospf` command.

auto-cost reference-bandwidth (1-4294967)

This sets the reference bandwidth for cost calculations, where this bandwidth is considered equivalent to an OSPF cost of 1, specified in Mbits/s. The default is 100Mbit/s (i.e. a link of bandwidth 100Mbit/s or higher will have a cost of 1. Cost of lower bandwidth links will be scaled with reference to this cost).

This configuration setting MUST be consistent across all routers within the OSPF domain.

network A.B.C.D/M area A.B.C.D

network A.B.C.D/M area (0-4294967295)

This command specifies the OSPF enabled interface(s). If the interface has an address from range 192.168.1.0/24 then the command below enables ospf on this interface so router can provide network information to the other ospf routers via this interface.

```
router ospf
network 192.168.1.0/24 area 0.0.0.0
```

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

Note that the behavior when there is a peer address defined on an interface changed after release 0.99.7. Currently, if a peer prefix has been configured, then we test whether the prefix in the network command contains the destination prefix. Otherwise, we test whether the network command prefix contains the local address prefix of the interface.

It is also possible to enable OSPF on a per interface/subnet basis using the interface command (*`ip ospf area AREA [ADDR]`*). However, mixing both network commands (`network`) and interface commands (`ip ospf`) on the same router is not supported.

proactive-arp

This command enables or disables sending ARP requests to update neighbor table entries. It speeds up convergence for /32 networks on a P2P connection.

This feature is enabled by default.

clear ip ospf [(1-65535)] process

This command can be used to clear the ospf process data structures. This will clear the ospf neighborship as well and it will get re-established. This will clear the LSDB too. This will be helpful when there is a change in router-id and if user wants the router-id change to take effect.

clear ip ospf [(1-65535)] neighbor

This command can be used to clear the ospf neighbor data structures. This will clear the ospf neighborship and it will get re-established. This command can be used when the neighbor state get stuck at some state and this can be used to recover it from that state.

maximum-paths (1-64)

Use this command to control the maximum number of equal cost paths to reach a specific destination. The upper limit may differ if you change the value of MULTIPATH_NUM during compilation. The default is MULTIPATH_NUM (64).

write-multiplier (1-100)

Use this command to tune the amount of work done in the packet read and write threads before relinquishing control. The parameter is the number of packets to process before returning. The defult value of this parameter is 20.

socket buffer <send | recv | all> (1-4000000000)

This command controls the ospf instance's socket buffer sizes. The 'no' form resets one or both values to the default.

no socket-per-interface

Ordinarily, ospfd uses a socket per interface for sending packets. This command disables those per-interface sockets, and causes ospfd to use a single socket per ospf instance for sending and receiving packets.

Areas

area A.B.C.D range A.B.C.D/M [advertise [cost (0-16777215)]]

area (0-4294967295) range A.B.C.D/M [advertise [cost (0-16777215)]]

Summarize intra area paths from specified area into one Type-3 summary-LSA announced to other areas. This command can be used only in ABR and ONLY router-LSAs (Type-1) and network-LSAs (Type-2) (i.e. LSAs with scope area) can be summarized. Type-5 AS-external-LSAs can't be summarized - their scope is AS.

```
router ospf
 network 192.168.1.0/24 area 0.0.0.0
 network 10.0.0.0/8 area 0.0.0.10
 area 0.0.0.10 range 10.0.0.0/8
```

With configuration above one Type-3 Summary-LSA with routing info 10.0.0.0/8 is announced into backbone area if area 0.0.0.10 contains at least one intra-area network (i.e. described with router or network LSA) from this range.

area A.B.C.D range A.B.C.D/M not-advertise

area (0-4294967295) range A.B.C.D/M not-advertise

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

area A.B.C.D range A.B.C.D/M {substitute A.B.C.D/M|cost (0-16777215)}

area (0-4294967295) range A.B.C.D/M {substitute A.B.C.D/M|cost (0-16777215)}

Substitute summarized prefix with another prefix.

```
router ospf
 network 192.168.1.0/24 area 0.0.0.0
 network 10.0.0.0/8 area 0.0.0.10
 area 0.0.0.10 range 10.0.0.0/8 substitute 11.0.0.0/8
```

One Type-3 summary-LSA with routing info 11.0.0.0/8 is announced into backbone area if area 0.0.0.10 contains at least one intra-area network (i.e. described with router-LSA or network-LSA) from range 10.0.0.0/8.

By default, the metric of the summary route is calculated as the highest metric among the summarized routes. The *cost* option, however, can be used to set an explicit metric.

This command makes sense in ABR only.

area A.B.C.D virtual-link A.B.C.D

area (0-4294967295) virtual-link A.B.C.D

area A.B.C.D shortcut

area (0-4294967295) shortcut

Configure the area as Shortcut capable. See **RFC 3509**. This requires that the 'abr-type' be set to 'shortcut'.

area A.B.C.D stub

area (0-4294967295) stub

Configure the area to be a stub area. That is, an area where no router originates routes external to OSPF and hence an area where all external routes are via the ABR(s). Hence, ABRs for such an area do not need to pass AS-External LSAs (type-5s) or ASBR-Summary LSAs (type-4) into the area. They need only pass Network-Summary (type-3) LSAs into such an area, along with a default-route summary.

area A.B.C.D stub no-summary

area (0-4294967295) stub no-summary

Prevents an *ospfd* ABR from injecting inter-area summaries into the specified stub area.

area A.B.C.D nssa

area (0-4294967295) nssa

Configure the area to be a NSSA (Not-So-Stubby Area). This is an area that allows OSPF to import external routes into a stub area via a new LSA type (type 7). An NSSA autonomous system boundary router (ASBR) will generate this type of LSA. The area border router (ABR) translates the LSA type 7 into LSA type 5, which is propagated into the OSPF domain. NSSA areas are defined in RFC 3101.

area A.B.C.D nssa suppress-fa

area (0-4294967295) nssa suppress-fa

Configure the router to set the forwarding address to 0.0.0.0 in all LSA type 5 translated from LSA type 7. The router needs to be elected the translator of the area for this command to take effect. This feature causes routers that are configured not to advertise forwarding addresses into the backbone to direct forwarded traffic to the NSSA ABR translator.

area A.B.C.D nssa default-information-originate [metric-type (1-2)] [metric (0-16777214)]

**area (0-4294967295) nssa default-information-originate \
[metric-type (1-2)] [metric (0-16777214)]**

NSSA ABRs and ASBRs can be configured with the *default-information-originate* option to originate a Type-7 default route into the NSSA area. In the case of NSSA ASBRs, the origination of the default route is conditioned to the existence of a default route in the RIB that wasn't learned via the OSPF protocol.

area A.B.C.D nssa range A.B.C.D/M [<not-advertise|cost (0-16777215)>]

area (0-4294967295) nssa range A.B.C.D/M [<not-advertise|cost (0-16777215)>]

Summarize a group of external subnets into a single Type-7 LSA, which is then translated to a Type-5 LSA and avertised to the backbone. This command can only be used at the area boundary (NSSA ABR router).

By default, the metric of the summary route is calculated as the highest metric among the summarized routes. The *cost* option, however, can be used to set an explicit metric.

The *not-advertise* option, when present, prevents the summary route from being advertised, effectively filtering the summarized routes.

area A.B.C.D default-cost (0-16777215)

Set the cost of default-summary LSAs announced to stubby areas.

area A.B.C.D export-list NAME

area (0-4294967295) export-list NAME

Filter Type-3 summary-LSAs announced to other areas originated from intra- area paths from specified area.

```
router ospf
 network 192.168.1.0/24 area 0.0.0.0
 network 10.0.0.0/8 area 0.0.0.10
 area 0.0.0.10 export-list foo
!
access-list foo permit 10.10.0.0/16
access-list foo deny any
```

With example above any intra-area paths from area 0.0.0.10 and from range 10.10.0.0/16 (for example 10.10.1.0/24 and 10.10.2.128/30) are announced into other areas as Type-3 summary-LSA's, but any others (for example 10.11.0.0/16 or 10.128.30.16/30) aren't.

This command is only relevant if the router is an ABR for the specified area.

area A.B.C.D import-list NAME

area (0-4294967295) import-list NAME

Same as export-list, but it applies to paths announced into specified area as Type-3 summary-LSAs.

area A.B.C.D filter-list prefix NAME in

area A.B.C.D filter-list prefix NAME out

area (0-4294967295) filter-list prefix NAME in

area (0-4294967295) filter-list prefix NAME out

Filtering Type-3 summary-LSAs to/from area using prefix lists. This command makes sense in ABR only.

area A.B.C.D authentication

area (0-4294967295) authentication

Specify that simple password authentication should be used for the given area.

area A.B.C.D authentication message-digest

area (0-4294967295) authentication message-digest

Specify that OSPF packets must be authenticated with MD5 HMACs within the given area. Keying material must also be configured on a per-interface basis (`ip ospf message-digest-key`).

MD5 authentication may also be configured on a per-interface basis (*ip ospf authentication message-digest*). Such per-interface settings will override any per-area authentication setting.

Interfaces

ip ospf area AREA [ADDR]

Enable OSPF on the interface, optionally restricted to just the IP address given by *ADDR*, putting it in the *AREA* area. If you have a lot of interfaces, and/or a lot of subnets, then enabling OSPF via this command instead of (*network A.B.C.D/M area A.B.C.D*) may result in a slight performance improvement.

Notice that, mixing both network commands (`network`) and interface commands (`ip ospf`) on the same router is not supported. If (`ip ospf`) is present, (`network`) commands will fail.

ip ospf authentication-key AUTH_KEY

Set OSPF authentication key to a simple password. After setting *AUTH_KEY*, all OSPF packets are authenticated. *AUTH_KEY* has length up to 8 chars.

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

ip ospf authentication message-digest

Specify that MD5 HMAC authentication must be used on this interface. MD5 keying material must also be configured. Overrides any authentication enabled on a per-area basis (*area A.B.C.D authentication message-digest*)

Note that OSPF MD5 authentication requires that time never go backwards (correct time is NOT important, only that it never goes backwards), even across resets, if ospfd is to be able to promptly reestablish adjacencies with its neighbours after restarts/reboots. The host should have system time be set at boot from an external or non-volatile source (e.g. battery backed clock, NTP, etc.) or else the system clock should be periodically saved to non-volatile storage and restored at boot if MD5 authentication is to be expected to work reliably.

ip ospf message-digest-key KEYID md5 KEY

Set OSPF authentication key to a cryptographic password. The cryptographic algorithm is MD5.

KEYID identifies secret key used to create the message digest. This ID is part of the protocol and must be consistent across routers on a link.

KEY is the actual message digest key, of up to 16 chars (larger strings will be truncated), and is associated with the given KEYID.

ip ospf authentication key-chain KEYCHAIN

Specify that HMAC cryptographic authentication must be used on this interface using a key chain. Overrides any authentication enabled on a per-area basis (*area A.B.C.D authentication message-digest*)

- `KEYCHAIN`: Specifies the name of the key chain that contains the authentication

key(s) and cryptographic algorithms to be used for OSPF authentication. The key chain is a logical container that holds one or more authentication keys, allowing for key rotation and management.

Note that OSPF HMAC cryptographic authentication requires that time never go backwards (correct time is NOT important, only that it never goes backwards), even across resets, if ospfd is to be able to promptly reestablish adjacencies with its neighbours after restarts/reboots. The host should have system time be set at boot from an external or non-volatile source (e.g. battery backed clock, NTP, etc.) or else the system clock should be

periodically saved to non-volatile storage and restored at boot if HMAC cryptographic authentication is to be expected to work reliably.

Example:

```
r1(config)#key chain temp
r1(config-keychain)#key 13
r1(config-keychain-key)#key-string ospf
r1(config-keychain-key)#cryptographic-algorithm hmac-sha-256
r1(config)#int eth0
r1(config-if)#ip ospf authentication key-chain temp
r1(config-if)#ip ospf area 0
```

ip ospf cost (1-65535)

Set link cost for the specified interface. The cost value is set to router-LSA's metric field and used for SPF calculation.

ip ospf dead-interval (1-65535)

ip ospf dead-interval minimal hello-multiplier (2-20)

Set number of seconds for RouterDeadInterval timer value used for Wait Timer and Inactivity Timer. This value must be the same for all routers attached to a common network. The default value is 40 seconds.

If 'minimal' is specified instead, then the dead-interval is set to 1 second and one must specify a hello-multiplier. The hello-multiplier specifies how many Hellos to send per second, from 2 (every 500ms) to 20 (every 50ms). Thus one can have 1s convergence time for OSPF. If this form is specified, then the hello-interval advertised in Hello packets is set to 0 and the hello-interval on received Hello packets is not checked, thus the hello-multiplier need NOT be the same across multiple routers on a common link.

ip ospf hello-interval (1-65535)

Set number of seconds for HelloInterval timer value. Setting this value, Hello packet will be sent every timer value seconds on the specified interface. This value must be the same for all routers attached to a common network. The default value is 10 seconds.

This command has no effect if *ip ospf dead-interval minimal hello-multiplier (2-20)* is also specified for the interface.

ip ospf graceful-restart hello-delay (1-1800)

Set the length of time during which Grace-LSAs are sent at 1-second intervals while coming back up after an unplanned outage. During this time, no hello packets are sent.

A higher hello delay will increase the chance that all neighbors are notified about the ongoing graceful restart before receiving a hello packet (which is crucial for the graceful restart to succeed). The hello delay shouldn't be set too high, however, otherwise the adjacencies might time out. As a best practice, it's recommended to set the hello delay and hello interval with the same values. The default value is 10 seconds.

**ip ospf network (broadcast|non-broadcast| point-to-multipoint [delay-reflood]| \
point-to-point)**

When configuring a point-to-point network on an interface and the interface has a /32 address associated with then OSPF will treat the interface as being *unnumbered*.

When the network is configured as point-to-multipoint and *delay-reflood* is specified, LSAs received on the interface from neighbors on the interface will not be flooded back out on the interface immediately. Rather, they will be added to the neighbor's link state retransmission list and only sent to the neighbor if the neighbor doesn't acknowledge the LSA prior to the link state retransmission timer expiring.

Set explicitly network type for specified interface.

ip ospf priority (0-255)

Set RouterPriority integer value. The router with the highest priority will be more eligible to become Designated Router. Setting the value to 0, makes the router ineligible to become Designated Router. The default value is 1.

ip ospf retransmit-interval (1-65535)

Set number of seconds for RxmtInterval timer value. This value is used when retransmitting Database Description and Link State Request packets. The default value is 5 seconds.

ip ospf transmit-delay (1-65535) [A.B.C.D]

Set number of seconds for InfTransDelay value. LSAs' age should be incremented by this value when transmitting. The default value is 1 second.

ip ospf passive [A.B.C.D]

Do not speak OSPF on the interface, but do advertise the interface as a stub link in the router-LSA for this router. This allows one to advertise addresses on such connected interfaces without having to originate AS-External/Type-5 LSAs (which have global flooding scope) - as would occur if connected addresses were redistributed into OSPF (*Redistribution*). This is the only way to advertise non-OSPF links into stub areas.

ip ospf area (A.B.C.D|(0-4294967295))

Enable ospf on an interface and set associated area.

OSPF route-map

Usage of *ospfd*'s route-map support.

set metric [+|-](0-4294967295)

Set a metric for matched route when sending announcement. Use plus (+) sign to add a metric value to an existing metric. Use minus (-) sign to substract a metric value from an existing metric.

Redistribution

**redistribute <bgp | connected | isis | kernel | ospf | rip | **
static | table> [metric-type (1-2)] [metric (0-16777214)] [route-map WORD]

Redistribute routes of the specified protocol or kind into OSPF, with the metric type and metric set if specified, filtering the routes using the given route-map if specified. Redistributed routes may also be filtered with distribute-lists, see *ospf distribute-list configuration*.

Redistributed routes are distributed as into OSPF as Type-5 External LSAs into links to areas that accept external routes, Type-7 External LSAs for NSSA areas and are not redistributed at all into Stub areas, where external routes are not permitted.

Note that for connected routes, one may instead use the *ip ospf passive [A.B.C.D]* configuration.

default-information originate

default-information originate metric (0-16777214)

default-information originate metric (0-16777214) metric-type (1|2)

default-information originate metric (0-16777214) metric-type (1|2) route-map WORD

default-information originate always

default-information originate always metric (0-16777214)

default-information originate always metric (0-16777214) metric-type (1|2)

default-information originate always metric (0-16777214) metric-type (1|2) route-map WORD

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

distribute-list NAME out <kernel|connected|static | rip|isis|bgp|table>

Apply the access-list filter, NAME, to redistributed routes of the given type before allowing the routes to be redistributed into OSPF (*ospf redistribution*).

default-metric (0-16777214)

distance (1-255)

distance ospf (intra-area | inter-area|external) (1-255)

Graceful Restart

graceful-restart [grace-period (1-1800)]

Configure Graceful Restart (RFC 3623) restarting support. When enabled, the default grace period is 120 seconds.

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

When Graceful Restart is enabled and the ospfd daemon crashes or is killed abruptely (e.g. SIGKILL), it will attempt an unplanned Graceful Restart once it restarts.

graceful-restart helper enable [A.B.C.D]

Configure Graceful Restart (RFC 3623) helper support. By default, helper support is disabled for all neighbours. This config enables/disables helper support on this router for all neighbours. To enable/disable helper support for a specific neighbour, the router-id (A.B.C.D) has to be specified.

graceful-restart helper strict-lsa-checking

If 'strict-lsa-checking' is configured then the helper will abort the Graceful Restart when a LSA change occurs which affects the restarting router. By default 'strict-lsa-checking' is enabled"

graceful-restart helper supported-grace-time

Supports as HELPER for configured grace period.

graceful-restart helper planned-only

It helps to support as HELPER only for planned restarts. By default, it supports both planned and unplanned outages.

graceful-restart prepare ip ospf

Initiate a graceful restart for all OSPF instances configured with the "graceful-restart" command. The ospfd daemon should be restarted during the instance-specific grace period, otherwise the graceful restart will fail.

This is an EXEC-level command.

Showing Information

show ip ospf [vrf <NAME|all>] [json]

Show information on a variety of general OSPF and area state and configuration information.

show ip ospf interface [INTERFACE] [json]

Show state and configuration of OSPF the specified interface, or all interfaces if no interface is given.

show ip ospf neighbor [json]

show ip ospf [vrf <NAME|all>] neighbor INTERFACE [json]

show ip ospf [vrf <NAME|all>] neighbor A.B.C.D [detail] [json]

show ip ospf [vrf <NAME|all>] neighbor INTERFACE detail [json]

Display lsa information of LSDB. Json o/p of this command covers base route information i.e all LSAs except opaque lsa info.

show ip ospf [vrf <NAME|all>] database [self-originate] [json]

Show the OSPF database summary.

show ip ospf [vrf <NAME|all>] database max-age [json]

Show all MaxAge LSAs present in the OSPF link-state database.

show ip ospf [vrf <NAME|all>] database detail \ [LINK-STATE-ID] [adv-router A.B.C.D] [json]

show ip ospf [vrf <NAME|all>] database detail \ [LINK-STATE-ID] [self-originate] [json]

**show ip ospf [vrf <NAME|all>] database **
(asbr-summary|external|network| router |summary| nssa-external|opaque-link
|opaque-area |opaque-as) [LINK-STATE-ID] [adv-router A.B.C.D] [json]

**show ip ospf [vrf <NAME|all>] database **
(asbr-summary|external|network| router|summary| nssa-external|opaque-link
|opaque-area|opaque-as) \ [LINK-STATE-ID] [self-originate] [json]

Show detailed information about the OSPF link-state database.

show ip ospf route [json]

Show the OSPF routing table, as determined by the most recent SPF calculation.

show ip ospf [vrf <NAME|all>] border-routers [json]

Show the list of ABR and ASBR border routers summary learnt via OSPFv2 Type-3 (Summary LSA) and Type-4 (Summary ASBR LSA). User can get that information as JSON format when `json` keyword at the end of cli is presented.

show ip ospf graceful-restart helper [detail] [json]

Displays the Grcaeful Restart Helper details including helper config changes.

Debugging OSPF

debug ospf [(1-65535)] bfd

Enable or disable debugging for BFD events. This will show BFD integration library messages and OSPF BFD integration messages that are mostly state transitions and validation problems.

debug ospf [(1-65535)] default-information

Show debug information of default information

debug ospf [(1-65535)] packet (hello|dd|ls-request | ls-update|ls-ack|all) \ (send|recv) [detail]

Dump Packet for debugging

debug ospf [(1-65535)] ism [status|events|timers]

Show debug information of Interface State Machine

debug ospf [(1-65535)] nsm [status|events|timers]

Show debug information of Network State Machine

debug ospf [(1-65535)] event

Show debug information of OSPF event

debug ospf [(1-65535)] nssa

Show debug information about Not So Stub Area

debug ospf [(1-65535)] ldp-sync

Show debug information about LDP-Sync

debug ospf [(1-65535)] lsa [aggregate|flooding |generate|install|refresh]

Show debug detail of Link State messages

debug ospf [(1-65535)] zebra [interface|redistribute]

Show debug information of ZEBRA API

debug ospf [(1-65535)] graceful-restart

Enable/disable debug information for OSPF Graceful Restart Helper

show debugging ospf

OSPF Configuration Examples

A simple example, with MD5 authentication enabled:

```
!
interface ge0
 ip ospf authentication message-digest
 ip ospf message-digest-key 1 md5 ABCDEFGHIJK
!
router ospf
 network 192.168.0.0/16 area 0.0.0.1
 area 0.0.0.1 authentication message-digest
```

An ABR router, with MD5 authentication and performing summarisation of networks between the areas:

```
!
log syslog
!
interface ge0
 ip ospf authentication message-digest
 ip ospf message-digest-key 1 md5 ABCDEFGHIJK
!
interface ge1
 ip ospf passive
!
interface ge2
 ip ospf authentication message-digest
 ip ospf message-digest-key 2 md5 XYZ12345
!
router ospf
 ospf router-id 192.168.0.1
 redistribute connected
 network 192.168.0.0/24 area 0.0.0.0
 network 10.0.0.0/16 area 0.0.0.0
 network 192.168.1.0/24 area 0.0.0.1
 area 0.0.0.0 authentication message-digest
 area 0.0.0.0 range 10.0.0.0/16
 area 0.0.0.0 range 192.168.0.0/24
 area 0.0.0.1 authentication message-digest
 area 0.0.0.1 range 10.2.0.0/16
!
```

1.5.6 OSPFv3

ospf6d is a daemon support OSPF version 3 for IPv6 network. OSPF for IPv6 is described in **RFC 2740**.

OSPF6 router

router ospf6 [vrf NAME]

ospf6 router-id A.B.C.D

Set router's Router-ID.

timers throttle spf (0-600000) (0-600000) (0-600000)

This command sets the initial *delay*, the *initial-holdtime* and the *maximum-holdtime* between when SPF is calculated and the event which triggered the calculation. The times are specified in milliseconds and must be in the range of 0 to 600000 milliseconds.

The *delay* specifies the minimum amount of time to delay SPF calculation (hence it affects how long SPF calculation is delayed after an event which occurs outside of the holdtime of any previous SPF calculation, and also serves as a minimum holdtime).

Consecutive SPF calculations will always be separated by at least 'hold-time' milliseconds. The hold-time is adaptive and initially is set to the *initial-holdtime* configured with the above command. Events which occur within the holdtime of the previous SPF calculation will cause the holdtime to be increased by *initial-holdtime*, bounded by the *maximum-holdtime* configured with this command. If the adaptive hold-time elapses without any SPF-triggering event occurring then the current holdtime is reset to the *initial-holdtime*.

```
router ospf6
 timers throttle spf 200 400 10000
```

In this example, the *delay* is set to 200ms, the initial holdtime is set to 400ms and the *maximum holdtime* to 10s. Hence there will always be at least 200ms between an event which requires SPF calculation and the actual SPF calculation. Further consecutive SPF calculations will always be separated by between 400ms to 10s, the hold-time increasing by 400ms each time an SPF-triggering event occurs within the hold-time of the previous SPF calculation.

auto-cost reference-bandwidth COST

This sets the reference bandwidth for cost calculations, where this bandwidth is considered equivalent to an OSPF cost of 1, specified in Mbits/s. The default is 100Mbit/s (i.e. a link of bandwidth 100Mbit/s or higher will have a cost of 1. Cost of lower bandwidth links will be scaled with reference to this cost).

This configuration setting MUST be consistent across all routers within the OSPF domain.

maximum-paths (1-64)

Use this command to control the maximum number of parallel routes that OSPFv3 can support. The default is 64.

write-multiplier (1-100)

Use this command to tune the amount of work done in the packet read and write threads before relinquishing control. The parameter is the number of packets to process before returning. The default value of this parameter is 20.

clear ipv6 ospf6 process [vrf NAME]

This command clears up the database and routing tables and resets the neighborship by restarting the interface state machine. This will be helpful when there is a change in router-id and if user wants the router-id change to take effect, user can use this cli instead of restarting the ospf6d daemon.

clear ipv6 ospf6 [vrf NAME] interface [IFNAME]

This command restarts the interface state machine for all interfaces in the VRF or only for the specific interface if `IFNAME` is specified.

ASBR Summarisation Support in OSPFv3

External routes in OSPFv3 are carried by type 5/7 LSA (external LSAs). External LSAs are generated by ASBR (Autonomous System Boundary Router). Large topology database requires a large amount of router memory, which slows down all processes, including SPF calculations. It is necessary to reduce the size of the OSPFv3 topology database, especially in a large network. Summarising routes keeps the routing tables smaller and easier to troubleshoot.

External route summarization must be configured on ASBR. Stub area do not allow ASBR because they don't allow type 5 LSAs.

An ASBR will inject a summary route into the OSPFv3 domain.

Summary route will only be advertised if you have at least one subnet that falls within the summary range.

Users will be allowed an option in the CLI to not advertise range of ipv6 prefixes as well.

The configuration of ASBR Summarisation is supported using the CLI command

summary-address X:X::X:X/
M [tag (1-4294967295)] [{metric (0-16777215) | metric-type (1-2)}]

This command will advertise a single External LSA on behalf of all the prefixes falling under this range configured by the CLI. The user is allowed to configure tag, metric and metric-type as well. By default, tag is not configured,

default metric as 20 and metric-type as type-2 gets advertised. A summary route is created when one or more specific routes are learned and removed when no more specific route exist. The summary route is also installed in the local system with Null0 as next-hop to avoid leaking traffic.

no summary-address X:X::X:X/
M [tag (1-4294967295)] [{metric (0-16777215) | metric-type (1-2)}]

This command can be used to remove the summarisation configuration. This will flush the single External LSA if it was originated and advertise the External LSAs for all the existing individual prefixes.

summary-address X:X::X:X/M no-advertise

This command can be used when user do not want to advertise a certain range of prefixes using the no-advertise option. This command when configured will flush all the existing external LSAs falling under this range.

no summary-address X:X::X:X/M no-advertise

This command can be used to remove the previous configuration. When configured, tt will resume originating external LSAs for all the prefixes falling under the configured range.

aggregation timer (5-1800)

The summarisation command takes effect after the aggregation timer expires. By default the value of this timer is 5 seconds. User can modify the time after which the external LSAs should get originated using this command.

no aggregation timer (5-1800)

This command removes the timer configuration. It reverts back to default 5 second timer.

show ipv6 ospf6 summary-address [detail] [json]

This command can be used to see all the summary-address related information. When detail option is used, it shows all the prefixes falling under each summary-configuration apart from other information.

OSPF6 area

area A.B.C.D range X:X::X:X/M [<advertise|not-advertise|cost (0-16777215)>]

area (0-4294967295) range X:X::X:X/M [<advertise|not-advertise|cost (0-16777215)>]

Summarize a group of internal subnets into a single Inter-Area-Prefix LSA. This command can only be used at the area boundary (ABR router).

By default, the metric of the summary route is calculated as the highest metric among the summarized routes. The *cost* option, however, can be used to set an explicit metric.

The *not-advertise* option, when present, prevents the summary route from being advertised, effectively filtering the summarized routes.

**area A.B.C.D nssa [no-summary] [default-information-originate **
[metric-type (1-2)] [metric (0-16777214)]]

**area (0-4294967295) nssa [no-summary] [default-information-originate **
[metric-type (1-2)] [metric (0-16777214)]]

Configure the area to be a NSSA (Not-So-Stubby Area).

The following functionalities are implemented as per RFC 3101:

1. Advertising Type-7 LSA into NSSA area when external route is redistributed into OSPFv3.
2. Processing Type-7 LSA received from neighbor and installing route in the route table.
3. Support for NSSA ABR functionality which is generating Type-5 LSA when backbone area is configured. Currently translation of Type-7 LSA to Type-5 LSA is enabled by default.
4. Support for NSSA Translator functionality when there are multiple NSSA ABR in an area.

An NSSA ABR can be configured with the *no-summary* option to prevent the advertisement of summaries into the area. In that case, a single Type-3 LSA containing a default route is originated into the NSSA.

NSSA ABRs and ASBRs can be configured with *default-information-originate* option to originate a Type-7 default route into the NSSA area. In the case of NSSA ASBRs, the origination of the default route is conditioned to the existence of a default route in the RIB that wasn't learned via the OSPF protocol.

area A.B.C.D nssa range X:X::X:X/M [<not-advertise|cost (0-16777215)>]

area (0-4294967295) nssa range X:X::X:X/M [<not-advertise|cost (0-16777215)>]

Summarize a group of external subnets into a single Type-7 LSA, which is then translated to a Type-5 LSA and avertised to the backbone. This command can only be used at the area boundary (NSSA ABR router).

By default, the metric of the summary route is calculated as the highest metric among the summarized routes. The *cost* option, however, can be used to set an explicit metric.

The *not-advertise* option, when present, prevents the summary route from being advertised, effectively filtering the summarized routes.

area A.B.C.D export-list NAME

area (0-4294967295) export-list NAME

Filter Type-3 summary-LSAs announced to other areas originated from intra- area paths from specified area.

```
router ospf6
 area 0.0.0.10 export-list foo
!
ipv6 access-list foo permit 2001:db8:1000::/64
ipv6 access-list foo deny any
```

With example above any intra-area paths from area 0.0.0.10 and from range 2001:db8::/32 (for example 2001:db8:1::/64 and 2001:db8:2::/64) are announced into other areas as Type-3 summary-LSA's, but any others (for example 2001:200::/48) aren't.

This command is only relevant if the router is an ABR for the specified area.

area A.B.C.D import-list NAME

area (0-4294967295) import-list NAME

Same as export-list, but it applies to paths announced into specified area as Type-3 summary-LSAs.

area A.B.C.D filter-list prefix NAME in

area A.B.C.D filter-list prefix NAME out

area (0-4294967295) filter-list prefix NAME in

area (0-4294967295) filter-list prefix NAME out

Filtering Type-3 summary-LSAs to/from area using prefix lists. This command makes sense in ABR only.

OSPF6 interface

ipv6 ospf6 area <A.B.C.D|(0-4294967295)>

Enable OSPFv3 on the interface and add it to the specified area.

ipv6 ospf6 cost COST

Sets interface's output cost. Default value depends on the interface bandwidth and on the auto-cost reference bandwidth.

ipv6 ospf6 hello-interval HELLOINTERVAL

Sets interface's Hello Interval. Default 10

ipv6 ospf6 dead-interval DEADINTERVAL

Sets interface's Router Dead Interval. Default value is 40.

ipv6 ospf6 graceful-restart hello-delay HELLODELAYINTERVAL

Set the length of time during which Grace-LSAs are sent at 1-second intervals while coming back up after an unplanned outage. During this time, no hello packets are sent.

A higher hello delay will increase the chance that all neighbors are notified about the ongoing graceful restart before receiving a hello packet (which is crucial for the graceful restart to succeed). The hello delay shouldn't be set too high, however, otherwise the adjacencies might time out. As a best practice, it's recommended to set the hello delay and hello interval with the same values. The default value is 10 seconds.

ipv6 ospf6 retransmit-interval RETRANSMITINTERVAL

Sets interface's Rxmt Interval. Default value is 5.

ipv6 ospf6 priority PRIORITY

Sets interface's Router Priority. Default value is 1.

ipv6 ospf6 transmit-delay TRANSMITDELAY

Sets interface's Inf-Trans-Delay. Default value is 1.

ipv6 ospf6 network (broadcast|point-to-point)

Set explicitly network type for specified interface.

OSPF6 route-map

Usage of *ospfd6*'s route-map support.

set metric [+|-](0-4294967295)

Set a metric for matched route when sending announcement. Use plus (+) sign to add a metric value to an existing metric. Use minus (-) sign to substract a metric value from an existing metric.

Redistribute routes to OSPF6

**redistribute <bgp | connected | isis | kernel | ripng | static | table> **
[metric-type (1-2)] [metric (0-16777214)] [route-map WORD]

Redistribute routes of the specified protocol or kind into OSPFv3, with the metric type and metric set if specified, filtering the routes using the given route-map if specified.

default-information originate [{always|metric (0-16777214)|metric-type (1-2)
|route-map WORD}]

The command injects default route in the connected areas. The always argument injects the default route regardless of it being present in the router. Metric values and route-map can also be specified optionally.

Graceful Restart

graceful-restart [grace-period (1-1800)]

Configure Graceful Restart (RFC 5187) restarting support. When enabled, the default grace period is 120 seconds.

To perform a graceful shutdown, the "graceful-restart prepare ipv6 ospf" EXEC-level command needs to be issued before restarting the ospf6d daemon.

When Graceful Restart is enabled and the ospf6d daemon crashes or is killed abruptely (e.g. SIGKILL), it will attempt an unplanned Graceful Restart once it restarts.

graceful-restart helper enable [A.B.C.D]

Configure Graceful Restart (RFC 5187) helper support. By default, helper support is disabled for all neighbours. This config enables/disables helper support on this router for all neighbours. To enable/disable helper support for a specific neighbour, the router-id (A.B.C.D) has to be specified.

graceful-restart helper strict-lsa-checking

If 'strict-lsa-checking' is configured then the helper will abort the Graceful Restart when a LSA change occurs which affects the restarting router. By default 'strict-lsa-checking' is enabled"

graceful-restart helper supported-grace-time (10-1800)

Supports as HELPER for configured grace period.

graceful-restart helper planned-only

It helps to support as HELPER only for planned restarts. By default, it supports both planned and unplanned outages.

graceful-restart prepare ipv6 ospf

Initiate a graceful restart for all OSPFv3 instances configured with the "graceful-restart" command. The ospf6d daemon should be restarted during the instance-specific grace period, otherwise the graceful restart will fail.

This is an EXEC-level command.

Showing OSPF6 information

show ipv6 ospf6 [vrf <NAME|all>] [json]

Show information on a variety of general OSPFv3 and area state and configuration information. JSON output can be obtained by appending 'json' to the end of command.

show ipv6 ospf6 [vrf <NAME|all>] database [<detail|dump|internal>] [json]

This command shows LSAs present in the LSDB. There are three view options. These options helps in viewing all the parameters of the LSAs. JSON output can be obtained by appending 'json' to the end of command. JSON option is not applicable with 'dump' option.

**show ipv6 ospf6 [vrf <NAME|all>] database <router | network | inter-prefix| \
inter-router | as-external | group-membership | type-7 | link | intra-prefix> [json]**

These options filters out the LSA based on its type. The three views options works here as well. JSON output can be obtained by appending 'json' to the end of command.

show ipv6 ospf6 [vrf <NAME|all>] database adv-router A.B.C.D linkstate-id A.B.C.D [json]

The LSAs additinally can also be filtered with the linkstate-id and advertising-router fields. We can use the LSA type filter and views with this command as well and visa-versa. JSON output can be obtained by appending 'json' to the end of command.

show ipv6 ospf6 [vrf <NAME|all>] database self-originated [json]

This command is used to filter the LSAs which are originated by the present router. All the other filters are applicable here as well.

show ipv6 ospf6 [vrf <NAME|all>] interface [json]

To see OSPF interface configuration like costs. JSON output can be obtained by appending "json" in the end.

show ipv6 ospf6 [vrf <NAME|all>] neighbor [json]

Shows state and chosen (Backup) DR of neighbor. JSON output can be obtained by appending 'json' at the end.

show ipv6 ospf6 [vrf <NAME|all>] interface traffic [json]

Shows counts of different packets that have been received and transmitted by the interfaces. JSON output can be obtained by appending "json" at the end.

show ipv6 route ospf6

This command shows internal routing table.

show ipv6 ospf6 zebra [json]

Shows state about what is being redistributed between zebra and OSPF6. JSON output can be obtained by appending "json" at the end.

show ipv6 ospf6 [vrf <NAME|all>] redistribute [json]

Shows the routes which are redistributed by the router. JSON output can be obtained by appending 'json' at the end.

show ipv6 ospf6 [vrf <NAME|all>] route [<intra-area| inter-area |external-1|external-2| \ X:X::X:X|X:X::X:X/M|detail|summary>] [json]

This command displays the ospfv3 routing table as determined by the most recent SPF calculations. Options are provided to view the different types of routes. Other than the standard view there are two other options, detail and summary. JSON output can be obtained by appending 'json' to the end of command.

show ipv6 ospf6 [vrf <NAME|all>] route X:X::X:X/M match [detail] [json]

The additional match option will match the given address to the destination of the routes, and return the result accordingly.

show ipv6 ospf6 [vrf <NAME|all>] interface [IFNAME] prefix [detail|<X:X::X:X|X:X::X:X/ M> [<match|detail>]] [json]

This command shows the prefixes present in the interface routing table. Interface name can also be given. JSON output can be obtained by appending 'json' to the end of command.

show ipv6 ospf6 [vrf <NAME|all>] spf tree [json]

This commands shows the spf tree from the recent spf calculation with the calling router as the root. If json is appended in the end, we can get the tree in JSON format. Each area that the router belongs to has it's own JSON object, with each router having "cost", "isLeafNode" and "children" as arguments.

show ipv6 ospf6 graceful-restart helper [detail] [json]

This command shows the graceful-restart helper details including helper configuration parameters.

OSPFv3 Debugging

The following debug commands are supported:

debug ospf6 abr

Toggle OSPFv3 ABR debugging messages.

debug ospf6 asbr

Toggle OSPFv3 ASBR debugging messages.

debug ospf6 border-routers {router-id [A.B.C.D] | area-id [A.B.C.D]}

Toggle OSPFv3 border router debugging messages. This can be specified for a router with specific Router-ID/Area-ID.

debug ospf6 flooding

Toggle OSPFv3 flooding debugging messages.

debug ospf6 interface

Toggle OSPFv3 interface related debugging messages.

debug ospf6 lsa

Toggle OSPFv3 Link State Advertisements debugging messages.

debug ospf6 lsa aggregation

Toggle OSPFv3 Link State Advertisements summarization debugging messages.

debug ospf6 message

Toggle OSPFv3 message exchange debugging messages.

debug ospf6 neighbor

Toggle OSPFv3 neighbor interaction debugging messages.

debug ospf6 nssa

Toggle OSPFv3 Not So Stubby Area (NSSA) debugging messages.

debug ospf6 route

Toggle OSPFv3 routes debugging messages.

debug ospf6 spf

Toggle OSPFv3 Shortest Path calculation debugging messages.

debug ospf6 zebra

Toggle OSPFv3 zebra interaction debugging messages.

debug ospf6 graceful-restart

Toggle OSPFv3 graceful-restart helper debugging messages.

OSPF6 Configuration Examples

Example of ospf6d configured on one interface and area:

```
interface ge0
 ipv6 ospf6 area 0.0.0.0
 ipv6 ospf6 instance-id 0
!
router ospf6
```

(continues on next page)

(continued from previous page)

```
 ospf6 router-id 212.17.55.53
 area 0.0.0.0 range 2001:770:105:2::/64
!
```

Larger example with policy and various options set:

```
debug ospf6 neighbor state
!
interface ge0
 ipv6 ospf6 area 0.0.0.0
 ipv6 ospf6 cost 1
 ipv6 ospf6 hello-interval 10
 ipv6 ospf6 dead-interval 40
 ipv6 ospf6 retransmit-interval 5
 ipv6 ospf6 priority 0
 ipv6 ospf6 transmit-delay 1
 ipv6 ospf6 instance-id 0
!
interface loopback0
 ipv6 ospf6 cost 1
 ipv6 ospf6 hello-interval 10
 ipv6 ospf6 dead-interval 40
 ipv6 ospf6 retransmit-interval 5
 ipv6 ospf6 priority 1
 ipv6 ospf6 transmit-delay 1
 ipv6 ospf6 instance-id 0
!
router ospf6
 router-id 255.1.1.1
 redistribute static route-map static-ospf6
!
!
ipv6 prefix-list test-prefix seq 1000 deny any
!
route-map static-ospf6 permit 10
 match ipv6 address prefix-list test-prefix
 set metric-type type-2
 set metric 2000
!
```

1.5.7 RIP

RIP – Routing Information Protocol is widely deployed interior gateway protocol. RIP was developed in the 1970s at Xerox Labs as part of the XNS routing protocol. RIP is a *distance-vector* protocol and is based on the *Bellman-Ford* algorithms. As a distance-vector protocol, RIP router send updates to its neighbors periodically, thus allowing the convergence to a known topology. In each update, the distance to any given network will be broadcast to its neighboring router.

ripd supports RIP version 2 as described in RFC2453 and RIP version 1 as described in RFC1058.

RIP netmask

The netmask features of *ripd* support both version 1 and version 2 of RIP. Version 1 of RIP originally contained no netmask information. In RIP version 1, network classes were originally used to determine the size of the netmask. Class A networks use 8 bits of mask, Class B networks use 16 bits of masks, while Class C networks use 24 bits of mask. Today, the most widely used method of a network mask is assigned to the packet on the basis of the interface that received the packet. Version 2 of RIP supports a variable length subnet mask (VLSM). By extending the subnet mask, the mask can be divided and reused. Each subnet can be used for different purposes such as large to middle size LANs and WAN links. FRR *ripd* does not support the non-sequential netmasks that are included in RIP Version 2.

In a case of similar information with the same prefix and metric, the old information will be suppressed. Ripd does not currently support equal cost multipath routing.

RIP Configuration

router rip [vrf NAME]

The *router rip* command is necessary to enable RIP. To disable RIP, use the *no router rip* command. RIP must be enabled before carrying out any of the RIP commands.

network NETWORK

Set the RIP enable interface by NETWORK. The interfaces which have addresses matching with NETWORK are enabled.

This group of commands either enables or disables RIP interfaces between certain numbers of a specified network address. For example, if the network for 10.0.0.0/24 is RIP enabled, this would result in all the addresses from 10.0.0.0 to 10.0.0.255 being enabled for RIP. The *no network* command will disable RIP for the specified network.

network IFNAME

Set a RIP enabled interface by IFNAME. Both the sending and receiving of RIP packets will be enabled on the port specified in the *network ifname* command. The *no network ifname* command will disable RIP on the specified interface.

neighbor A.B.C.D

Specify a RIP neighbor to send updates to. This is required when a neighbor is connected via a network that does not support multicast, or when it is desired to statically define a neighbor. RIP updates will be sent via unicast to each neighbour. Neighbour updates are in addition to any multicast updates sent when an interface is not in passive mode (see the *passive-interface* command). RIP will continue to process updates received from both the neighbor and any received via multicast. The *no neighbor a.b.c.d* command will disable the RIP neighbor.

Below is very simple RIP configuration. Interface *eth0* and interface which address match to *10.0.0.0/8* are RIP enabled.

```
!
router rip
 network 10.0.0.0/8
 network eth0
!
```

passive-interface (IFNAME|default)

This command sets the specified interface to passive mode. On passive mode interface, all receiving packets are processed as normal and ripd does not send either multicast or unicast RIP packets except to RIP neighbors specified with *neighbor* command. The interface may be specified as *default* to make ripd default to passive on all interfaces.

The default is to be passive on all interfaces.

ip split-horizon [poisoned-reverse]

Control split-horizon on the interface. Default is *ip split-horizon*. If you don't perform split-horizon on the interface, please specify *no ip split-horizon*.

If *poisoned-reverse* is also set, the router sends the poisoned routes with highest metric back to the sending router.

allow-ecmp [1-MULTIPATH_NUM]

Control how many ECMP paths RIP can inject for the same prefix. If specified without a number, a maximum is taken (64).

RIP Version Control

RIP can be configured to send either Version 1 or Version 2 packets. The default is to send RIPv2 while accepting both RIPv1 and RIPv2 (and replying with packets of the appropriate version for REQUESTS / triggered updates). The version to receive and send can be specified globally, and further overridden on a per-interface basis if needs be for send and receive separately (see below).

It is important to note that RIPv1 cannot be authenticated. Further, if RIPv1 is enabled then RIP will reply to REQUEST packets, sending the state of its RIP routing table to any remote routers that ask on demand. For a more detailed discussion on the security implications of RIPv1 see *RIP Authentication*.

version VERSION

Set RIP version to accept for reads and send. VERSION can be either `1` or `2`.

Disabling RIPv1 by specifying version 2 is STRONGLY encouraged, *RIP Authentication*. This may become the default in a future release.

Default: Send Version 2, and accept either version.

ip rip send version VERSION

VERSION can be `1`, `2`, or `1 2`.

This interface command overrides the global rip version setting, and selects which version of RIP to send packets with, for this interface specifically. Choice of RIP Version 1, RIP Version 2, or both versions. In the latter case, where `1 2` is specified, packets will be both broadcast and multicast.

Default: Send packets according to the global version (version 2)

ip rip receive version VERSION

VERSION can be `1`, `2`, or `1 2`.

This interface command overrides the global rip version setting, and selects which versions of RIP packets will be accepted on this interface. Choice of RIP Version 1, RIP Version 2, or both.

Default: Accept packets according to the global setting (both 1 and 2).

How to Announce RIP route

**redistribute < bgp | connected | isis | kernel | ospf | sharp | static | table> **
[metric (0-16)] [route-map WORD]

Redistribute routes from other sources into RIP.

If you want to specify RIP only static routes:

default-information originate

route A.B.C.D/M

This command is specific to FRR. The *route* command makes a static route only inside RIP. This command should be used only by advanced users who are particularly knowledgeable about the RIP protocol. In most cases, we recommend creating a static route in FRR and redistributing it in RIP using *redistribute static*.

Filtering RIP Routes

RIP routes can be filtered by a distribute-list.

distribute-list [prefix] LIST <in|out> IFNAME

You can apply access lists to the interface with a *distribute-list* command. If prefix is specified LIST is a prefix-list. If prefix is not specified then LIST is the access list name. *in* specifies packets being received, and *out* specifies outgoing packets. Finally if an interface is specified it will be applied against a specific interface.

The *distribute-list* command can be used to filter the RIP path. *distribute-list* can apply access-lists to a chosen interface. First, one should specify the access-list. Next, the name of the access-list is used in the distribute-list command. For example, in the following configuration `eth0` will permit only the paths that match the route 10.0.0.0/8

```
!
router rip
 distribute-list private in eth0
!
access-list private permit 10 10.0.0.0/8
access-list private deny any
!
```

distribute-list can be applied to both incoming and outgoing data.

RIP Metric Manipulation

RIP metric is a value for distance for the network. Usually *ripd* increment the metric when the network information is received. Redistributed routes' metric is set to 1.

default-metric (1-16)

This command modifies the default metric value for redistributed routes. The default value is 1. This command does not affect connected route even if it is redistributed by *redistribute connected*. To modify connected route's metric value, please use `redistribute connected metric` or *route-map*. *offset-list* also affects connected routes.

offset-list ACCESS-LIST (in|out)

offset-list ACCESS-LIST (in|out) IFNAME

RIP distance

Distance value is used in zebra daemon. Default RIP distance is 120.

distance (1-255)

Set default RIP distance to specified value.

distance (1-255) A.B.C.D/M

Set default RIP distance to specified value when the route's source IP address matches the specified prefix.

distance (1-255) A.B.C.D/M ACCESS-LIST

Set default RIP distance to specified value when the route's source IP address matches the specified prefix and the specified access-list.

RIP route-map

Usage of *ripd*'s route-map support.

Optional argument route-map MAP_NAME can be added to each *redistribute* statement.

```
redistribute static [route-map MAP_NAME]
redistribute connected [route-map MAP_NAME]
.....
```

Cisco applies route-map _before_ routes will exported to rip route table. In current FRR's test implementation, *ripd* applies route-map after routes are listed in the route table and before routes will be announced to an interface (something like output filter). I think it is not so clear, but it is draft and it may be changed at future.

Route-map statement (*Route Maps*) is needed to use route-map functionality.

match interface WORD

This command match to incoming interface. Notation of this match is different from Cisco. Cisco uses a list of interfaces - NAME1 NAME2 ... NAMEN. Ripd allows only one name (maybe will change in the future). Next - Cisco means interface which includes next-hop of routes (it is somewhat similar to "ip next-hop" statement). Ripd means interface where this route will be sent. This difference is because "next-hop" of same routes which sends to different interfaces must be different. Maybe it'd be better to made new matches - say "match interface-out NAME" or something like that.

match ip address WORD

match ip address prefix-list WORD

Match if route destination is permitted by access-list.

match ip next-hop WORD

match ip next-hop prefix-list WORD

Match if route next-hop (meaning next-hop listed in the rip route-table as displayed by "show ip rip") is permitted by access-list.

match metric (0-4294967295)

This command match to the metric value of RIP updates. For other protocol compatibility metric range is shown as (0-4294967295). But for RIP protocol only the value range (0-16) make sense.

set ip next-hop A.B.C.D

This command set next hop value in RIPv2 protocol. This command does not affect RIPv1 because there is no next hop field in the packet.

set metric (0-4294967295)

Set a metric for matched route when sending announcement. The metric value range is very large for compatibility with other protocols. For RIP, valid metric values are from 1 to 16.

RIP Authentication

RIPv2 allows packets to be authenticated via either an insecure plain text password, included with the packet, or via a more secure MD5 based HMAC (keyed-Hashing for Message AuthentiCation), RIPv1 can not be authenticated at all, thus when authentication is configured *ripd* will discard routing updates received via RIPv1 packets.

However, unless RIPv1 reception is disabled entirely, *RIP Version Control*, RIPv1 REQUEST packets which are received, which query the router for routing information, will still be honoured by *ripd*, and *ripd* WILL reply to such packets. This allows *ripd* to honour such REQUESTs (which sometimes is used by old equipment and very simple devices to bootstrap their default route), while still providing security for route updates which are received.

In short: Enabling authentication prevents routes being updated by unauthenticated remote routers, but still can allow routes (I.e. the entire RIP routing table) to be queried remotely, potentially by anyone on the internet, via RIPv1.

To prevent such unauthenticated querying of routes disable RIPv1, *RIP Version Control*.

ip rip authentication mode md5

Set the interface with RIPv2 MD5 authentication.

ip rip authentication mode text

Set the interface with RIPv2 simple password authentication.

ip rip authentication string STRING

RIP version 2 has simple text authentication. This command sets authentication string. The string must be shorter than 16 characters.

ip rip authentication key-chain KEY-CHAIN

Specify Keyed MD5 chain.

```
!
key chain test
 key 1
  key-string test
!
interface eth1
 ip rip authentication mode md5
 ip rip authentication key-chain test
!
```

RIP Timers

timers basic UPDATE TIMEOUT GARBAGE

RIP protocol has several timers. User can configure those timers' values by *timers basic* command.

The default settings for the timers are as follows:

- The update timer is 30 seconds. Every update timer seconds, the RIP process is awakened to send an unsolicited Response message containing the complete routing table to all neighboring RIP routers.
- The timeout timer is 180 seconds. Upon expiration of the timeout, the route is no longer valid; however, it is retained in the routing table for a short time so that neighbors can be notified that the route has been dropped.

- The garbage collect timer is 120 seconds. Upon expiration of the garbage-collection timer, the route is finally removed from the routing table.

The `timers basic` command allows the the default values of the timers listed above to be changed.

Show RIP Information

To display RIP routes.

show ip rip [vrf NAME]
Show RIP routes.

The command displays all RIP routes. For routes that are received through RIP, this command will display the time the packet was sent and the tag information. This command will also display this information for routes redistributed into RIP.

show ip rip [vrf NAME] status
The command displays current RIP status. It includes RIP timer, filtering, version, RIP enabled interface and RIP peer information.

```
ripd> **show ip rip status**
Routing Protocol is "rip"
  Sending updates every 30 seconds with +/-50%, next due in 35 seconds
  Timeout after 180 seconds, garbage collect after 120 seconds
  Outgoing update filter list for all interface is not set
  Incoming update filter list for all interface is not set
  Default redistribution metric is 1
  Redistributing: kernel connected
  Default version control: send version 2, receive version 2
    Interface  Send  Recv
  Routing for Networks:
    eth0
    eth1
    1.1.1.1
    203.181.89.241
  Routing Information Sources:
    Gateway    BadPackets BadRoutes  Distance Last Update
```

RIP Debug Commands

Debug for RIP protocol.

debug rip events
Shows RIP events. Sending and receiving packets, timers, and changes in interfaces are events shown with *ripd*.

debug rip packet
Shows display detailed information about the RIP packets. The origin and port number of the packet as well as a packet dump is shown.

debug rip zebra
This command will show the communication between *ripd* and *zebra*. The main information will include addition and deletion of paths to the kernel and the sending and receiving of interface information.

show debugging rip
Shows all information currently set for ripd debug.

Sample configuration

```
debug rip events
debug rip packet

router rip
 network 11.0.0.0/8
 network eth0
 route 10.0.0.0/8
 distribute-list private-only in eth0

access-list private-only
 permit 10.0.0.0/8 any
 deny any any
```

1.5.8 RIPng

ripngd supports the RIPng protocol as described in **RFC 2080**. It's an IPv6 reincarnation of the RIP protocol.

ripngd Configuration

Currently ripngd supports the following commands:

router ripng [vrf NAME]
: Enable RIPng.

network NETWORK
: Set RIPng enabled interface by NETWORK.

network IFNAME
: Set RIPng enabled interface by IFNAME.

route NETWORK
: Set RIPng static routing announcement of NETWORK.

allow-ecmp [1-MULTIPATH_NUM]
: Control how many ECMP paths RIPng can inject for the same prefix. If specified without a number, a maximum is taken (64).

ripngd Terminal Mode Commands

show ipv6 ripng [vrf NAME] status

show debugging ripng

debug ripng events

debug ripng packet

debug ripng zebra

ripngd Filtering Commands

RIPng routes can be filtered by a distribute-list.

distribute-list [prefix] LIST <in|out> IFNAME

You can apply access lists to the interface with a *distribute-list* command. If prefix is specified LIST is a prefix-list. If prefix is not specified then LIST is the access list name. *in* specifies packets being received, and *out* specifies outgoing packets. Finally if an interface is specified it will be applied against a specific interface.

The `distribute-list` command can be used to filter the RIPNG path. `distribute-list` can apply access-lists to a chosen interface. First, one should specify the access-list. Next, the name of the access-list is used in the distribute-list command. For example, in the following configuration `eth0` will permit only the paths that match the route 10.0.0.0/8

```
!
router ripng
 distribute-list private in ge0
!
access-list private
 permit 10.0.0.0/8 any
 deny any any
!
```

distribute-list can be applied to both incoming and outgoing data.

Sample configuration

```
debug ripng events
debug ripng packet

router ripng
 network ge0
 route 3ffe:506::0/32
 distribute-list local-only out sit1

ipv6 access-list local-only
 permit 3ffe:506::0/32 any
 deny any any
```

1.5.9 STATIC

STATIC is a daemon that handles the installation and deletion of static routes.

Static Route Commands

Static routing is a very fundamental feature of routing technology. It defines a static prefix and gateway, with several possible forms.

```
ip route NETWORK GATEWAY [DISTANCE] [table TABLENO] \ [nexthop-vrf VRFNAME] [vrf VRFNAME]

ip route NETWORK IFNAME [DISTANCE] [table TABLENO] \ [nexthop-vrf VRFNAME] [vrf VRFNAME]

ip route NETWORK GATEWAY IFNAME [DISTANCE] [onlink] \
[table TABLENO] [nexthop-vrf VRFNAME] [vrf VRFNAME]

ip route NETWORK (Null0| blackhole|reject) [DISTANCE] \
[table TABLENO] [nexthop-vrf VRFNAME] [vrf VRFNAME]

ipv6 route NETWORK [from SRCPREFIX] GATEWAY [DISTANCE] \
[table TABLENO] [nexthop-vrf VRFNAME] [vrf VRFNAME]

ipv6 route NETWORK [from SRCPREFIX] IFNAME [DISTANCE] \
[table TABLENO] [nexthop-vrf VRFNAME] [vrf VRFNAME]

ipv6 route NETWORK [from SRCPREFIX] GATEWAY IFNAME \
[DISTANCE] [onlink] [table TABLENO] \ [nexthop-vrf VRFNAME] [vrf VRFNAME]

ipv6 route NETWORK [from SRCPREFIX] (Null0| blackhole|reject) \
[DISTANCE] [table TABLENO] [nexthop-vrf VRFNAME] \ [vrf VRFNAME]
```

NETWORK is destination prefix with a valid v4 or v6 network based upon initial form of the command.

GATEWAY is the IP address to use as next-hop for the prefix. Currently, it must match the v4 or v6 route type specified at the start of the command.

IFNAME is the name of the interface to use as next-hop. If only IFNAME is specified (without GATEWAY), a connected route will be created.

When both IFNAME and GATEWAY are specified together, it binds the route to the specified interface. In this case, it is also possible to specify `onlink` to force the kernel to consider the next-hop as "on link" on the given interface.

Alternatively, the gateway can be specified as `Null0` or `blackhole` to create a blackhole route that drops all traffic. It can also be specified as `reject` to create an unreachable route that rejects traffic with ICMP "Destination Unreachable" messages.

TABLENO is an optional parameter for namespaces that allows you to create the route in a specified table associated with the vrf namespace. `table` will be rejected if you are not using namespace based vrfs.

`vrf` VRFNAME allows you to create the route in a specified vrf.

`nexthop-vrf` VRFNAME allows you to create a leaked route with a nexthop in the specified VRFNAME. `nexthop-vrf` cannot be currently used with namespace based vrfs.

The IPv6 variant allows the installation of a static source-specific route with the SRCPREFIX sub command. These routes are currently supported on Linux operating systems only, and perform AND matching on packet's destination and source addresses in the kernel's forwarding path. Note that destination longest-prefix match is "more important" than source LPM, e.g. `2001:db8:1::/64 from 2001:db8::/48` will win over `2001:db8::/48 from 2001:db8:1::/64` if both match.

Multiple nexthop static route

To create multiple nexthops to the same NETWORK (also known as a multipath route), just reenter the same network statement with different nexthop information.

```
ip route 10.0.0.1/32 10.0.0.2
ip route 10.0.0.1/32 10.0.0.3
ip route 10.0.0.1/32 eth0
```

If there is no route to 10.0.0.2 and 10.0.0.3, and interface eth0 is reachable, then the last route is installed into the kernel.

If zebra has been compiled with multipath support, and both 10.0.0.2 and 10.0.0.3 are reachable, zebra will install a multipath route via both nexthops, if the platform supports this.

```
router> show ip route
S>  10.0.0.1/32 [1/0] via 10.0.0.2 inactive
    via 10.0.0.3 inactive
  *       is directly connected, eth0
```

```
ip route 10.0.0.0/8 10.0.0.2
ip route 10.0.0.0/8 10.0.0.3
ip route 10.0.0.0/8 null0 255
```

This will install a multipath route via the specified next-hops if they are reachable, as well as a high-distance blackhole route, which can be useful to prevent traffic destined for a prefix to match less-specific routes (e.g. default) should the specified gateways not be reachable. E.g.:

```
router> show ip route 10.0.0.0/8
Routing entry for 10.0.0.0/8
  Known via "static", distance 1, metric 0
    10.0.0.2 inactive
    10.0.0.3 inactive

Routing entry for 10.0.0.0/8
  Known via "static", distance 255, metric 0
    directly connected, Null0
```

Also, if the user wants to configure a static route for a specific VRF, then a specific VRF configuration mode is available. After entering into that mode with *vrf VRF* the user can enter the same route command as before, but this time, the route command will apply to the VRF.

```
# case with VRF
configure
vrf r1-cust1
 ip route 10.0.0.0/24 10.0.0.2
exit-vrf
```

1.5.10 PIM

PIM – Protocol Independent Multicast

pimd supports pim-sm as well as igmp v2 and v3. pim is vrf aware and can work within the context of vrf's in order to do S,G mrouting.

ip pim rp A.B.C.D A.B.C.D/M

In order to use pim, it is necessary to configure a RP for join messages to be sent to. Currently the only methodology to do this is via static rp commands. All routers in the pim network must agree on these values. The first ip address is the RP's address and the second value is the matching prefix of group ranges covered. This command is vrf aware, to configure for a vrf, enter the vrf submode.

ip pim rp keep-alive-timer (1-65535)

Modify the time out value for a S,G flow from 1-65535 seconds at RP. The normal keepalive period for the KAT(S,G) defaults to 210 seconds. However, at the RP, the keepalive period must be at least the Register_Suppression_Time, or the RP may time out the (S,G) state before the next Null-Register arrives. Thus, the KAT(S,G) is set to max(Keepalive_Period, RP_Keepalive_Period) when a Register-Stop is sent. If choosing a value below 31 seconds be aware that some hardware platforms cannot see data flowing in better than 30 second chunks. This command is vrf aware, to configure for a vrf, enter the vrf submode.

ip pim register-accept-list PLIST

When pim receives a register packet the source of the packet will be compared to the prefix-list specified, PLIST, and if a permit is received normal processing continues. If a deny is returned for the source address of the register packet a register stop message is sent to the source.

ip pim spt-switchover infinity-and-beyond [prefix-list PLIST]

On the last hop router if it is desired to not switch over to the SPT tree configure this command. Optional parameter prefix-list can be use to control which groups to switch or not switch. If a group is PERMIT as per the PLIST, then the SPT switchover does not happen for it and if it is DENY, then the SPT switchover happens. This command is vrf aware, to configure for a vrf, enter the vrf submode.

ip pim ecmp

If pim has the a choice of ECMP nexthops for a particular RPF, pim will cause S,G flows to be spread out amongst the nexthops. If this command is not specified then the first nexthop found will be used. This command is vrf aware, to configure for a vrf, enter the vrf submode.

ip pim ecmp rebalance

If pim is using ECMP and an interface goes down, cause pim to rebalance all S,G flows across the remaining nexthops. If this command is not configured pim only modifies those S,G flows that were using the interface that went down. This command is vrf aware, to configure for a vrf, enter the vrf submode.

ip pim join-prune-interval (1-65535)

Modify the join/prune interval that pim uses to the new value. Time is specified in seconds. This command is vrf aware, to configure for a vrf, enter the vrf submode. The default time is 60 seconds. If you enter a value smaller than 60 seconds be aware that this can and will affect convergence at scale.

ip pim keep-alive-timer (1-65535)

Modify the time out value for a S,G flow from 1-65535 seconds. If choosing a value below 31 seconds be aware that some hardware platforms cannot see data flowing in better than 30 second chunks. This command is vrf aware, to configure for a vrf, enter the vrf submode.

ip pim packets (1-255)

When processing packets from a neighbor process the number of packets incoming at one time before moving on to the next task. The default value is 3 packets. This command is only useful at scale when you can possibly have a large number of pim control packets flowing. This command is vrf aware, to configure for a vrf, enter the vrf submode.

`ip pim register-suppress-time (1-65535)`

Modify the time that pim will register suppress a FHR will send register notifications to the kernel. This command is vrf aware, to configure for a vrf, enter the vrf submode.

`ip pim send-v6-secondary`

When sending pim hello packets tell pim to send any v6 secondary addresses on the interface. This information is used to allow pim to use v6 nexthops in it's decision for RPF lookup. This command is vrf aware, to configure for a vrf, enter the vrf submode.

`ip pim ssm prefix-list WORD`

Specify a range of group addresses via a prefix-list that forces pim to never do SM over. This command is vrf aware, to configure for a vrf, enter the vrf submode.

`ip multicast rpf-lookup-mode WORD`

Modify how PIM does RPF lookups in the zebra routing table. You can use these choices:

longer-prefix
Lookup the RPF in both tables using the longer prefix as a match

lower-distance
Lookup the RPF in both tables using the lower distance as a match

mrib-only
Lookup in the Multicast RIB only

mrib-then-urib
Lookup in the Multicast RIB then the Unicast Rib, returning first found. This is the default value for lookup if this command is not entered

urib-only
Lookup in the Unicast Rib only.

`ip igmp generate-query-once [version (2-3)]`

Generate IGMP query (v2/v3) on user requirement. This will not depend on the existing IGMP general query timer.If no version is provided in the cli, the default will be the igmp version enabled on that interface.

`ip igmp watermark-warn (1-65535)`

Configure watermark warning generation for an igmp group limit. Generates warning once the configured group limit is reached while adding new groups. 'no' form of the command disables the warning generation. This command is vrf aware. To configure per vrf, enter vrf submode.

PIM Interface Configuration

PIM interface commands allow you to configure an interface as either a Receiver or a interface that you would like to form pim neighbors on. If the interface is in a vrf, enter the interface command with the vrf keyword at the end.

`ip pim active-active`

Turn on pim active-active configuration for a Vxlan interface. This command will not do anything if you do not have the underlying ability of a mlag implementation.

`ip pim bsm`

Tell pim that we would like to use this interface to process bootstrap messages. This is enabled by default. 'no' form of this command is used to restrict bsm messages on this interface.

`ip pim unicast-bsm`

Tell pim that we would like to allow interface to process unicast bootstrap messages. This is enabled by default. 'no' form of this command is used to restrict processing of unicast bsm messages on this interface.

ip pim drpriority (1-4294967295)

Set the DR Priority for the interface. This command is useful to allow the user to influence what node becomes the DR for a lan segment.

ip pim hello (1-65535) (1-65535)

Set the pim hello and hold interval for a interface.

ip pim

Tell pim that we would like to use this interface to form pim neighbors over. Please note that this command does not enable the reception of IGMP reports on the interface. Refer to the next *ip igmp* command for IGMP management.

ip pim use-source A.B.C.D

If you have multiple addresses configured on a particular interface and would like pim to use a specific source address associated with that interface.

ip pim passive

Disable sending and receiving pim control packets on the interface.

ip igmp

Tell pim to receive IGMP reports and Query on this interface. The default version is v3. This command is useful on a LHR.

ip igmp join A.B.C.D [A.B.C.D]

Join multicast group or source-group on an interface.

ip igmp query-interval (1-65535)

Set the IGMP query interval that PIM will use.

ip igmp query-max-response-time (1-65535)

Set the IGMP query response timeout value. If an report is not returned in the specified time we will assume the S,G or *,G has timed out.

ip igmp version (2-3)

Set the IGMP version used on this interface. The default value is 3.

ip multicast boundary oil WORD

Set a pim multicast boundary, based upon the WORD prefix-list. If a pim join or IGMP report is received on this interface and the Group is denied by the prefix-list, PIM will ignore the join or report.

ip igmp last-member-query-count (1-255)

Set the IGMP last member query count. The default value is 2. 'no' form of this command is used to to configure back to the default value.

ip igmp last-member-query-interval (1-65535)

Set the IGMP last member query interval in deciseconds. The default value is 10 deciseconds. 'no' form of this command is used to to configure back to the default value.

ip mroute INTERFACE A.B.C.D [A.B.C.D]

Set a static multicast route for a traffic coming on the current interface to be forwarded on the given interface if the traffic matches the group address and optionally the source address.

See also:

PIM BFD Configuration

PIM Multicast RIB

In order to influence Multicast RPF lookup, it is possible to insert into zebra routes for the Multicast RIB. These routes are only used for RPF lookup and will not be used by zebra for insertion into the kernel *or* for normal rib processing. As such it is possible to create weird states with these commands. Use with caution. Most of the time this will not be necessary.

ip mroute A.B.C.D/M A.B.C.D (1-255)

Insert into the Multicast Rib Route A.B.C.D/M with specified nexthop. The distance can be specified as well if desired.

ip mroute A.B.C.D/M INTERFACE (1-255)

Insert into the Multicast Rib Route A.B.C.D/M using the specified INTERFACE. The distance can be specified as well if desired.

Multicast Source Discovery Protocol (MSDP) Configuration

MSDP can be setup in different ways:

- MSDP meshed-group: where all peers are connected with each other creating a fully meshed network. SAs (source active) messages are not forwarded in this mode because the origin is able to send SAs to all members.

  This setup is commonly used with anycast.

- MSDP peering: when there is one or more peers that are not fully meshed. SAs may be forwarded depending on the result of filtering and RPF checks.

  This setup is commonly consistent with BGP peerings (for RPF checks).

- MSDP default peer: there is only one peer and all SAs will be forwarded there.

Note: MSDP default peer and SA filtering is not implemented.

Commands available for MSDP:

ip msdp timers (1-65535) (1-65535) [(1-65535)]

Configure global MSDP timers.

First value is the keep-alive interval. This configures the interval in seconds between keep-alive messages. The default value is 60 seconds. It should be less than the remote hold time.

Second value is the hold-time. This configures the interval in seconds before closing a non responding connection. The default value is 75. This value should be greater than the remote keep alive time.

Third value is the connection retry interval and it is optional. This configures the interval between connection attempts. The default value is 30 seconds.

ip msdp mesh-group WORD member A.B.C.D

Create or update a mesh group to include the specified MSDP peer.

ip msdp mesh-group WORD source A.B.C.D

Create or update a mesh group to set the source address used to connect to peers.

ip msdp peer A.B.C.D source A.B.C.D

Create a regular MSDP session with peer using the specified source address.

Show PIM Information

All PIM show commands are vrf aware and typically allow you to insert a specified vrf command if information is desired about a specific vrf. If no vrf is specified then the default vrf is assumed. Finally the special keyword 'all' allows you to look at all vrfs for the command. Naming a vrf 'all' will cause great confusion.

show ip igmp interface

Display IGMP interface information.

show ip igmp [vrf NAME] join [json]

Display IGMP static join information for a specific vrf.

show ip igmp [vrf VRFNAME] groups [INTERFACE [GROUP]] [detail] [json]

Display IGMP static join information for all the vrfs present.

show ip igmp vrf all groups [GROUP] [detail] [json]

Display IGMP groups information.

show ip igmp groups retransmissions

Display IGMP group retransmission information.

show ip igmp [vrf NAME] sources [json]

Display IGMP sources information.

show ip igmp sources retransmissions

Display IGMP source retransmission information.

show ip igmp statistics

Display IGMP statistics information.

show ip multicast

Display various information about the interfaces used in this pim instance.

show ip mroute [vrf NAME] [A.B.C.D [A.B.C.D]] [fill] [json]

Display information about installed into the kernel S,G mroutes. If one address is specified we assume it is the Group we are interested in displaying data on. If the second address is specified then it is Source Group. The keyword *fill* says to fill in all assumed data for test/data gathering purposes.

show ip mroute [vrf NAME] count [json]

Display information about installed into the kernel S,G mroutes and in addition display data about packet flow for the mroutes for a specific vrf.

show ip mroute vrf all count [json]

Display information about installed into the kernel S,G mroutes and in addition display data about packet flow for the mroutes for all vrfs.

show ip mroute [vrf NAME] summary [json]

Display total number of S,G mroutes and number of S,G mroutes installed into the kernel for a specific vrf.

show ip mroute vrf all summary [json]

Display total number of S,G mroutes and number of S,G mroutes installed into the kernel for all vrfs.

show ip msdp mesh-group

Display the configured mesh-groups, the local address associated with each mesh-group, the peer members included in each mesh-group, and their status.

show ip msdp peer

Display information about the MSDP peers. That includes the peer address, the local address used to establish the connection to the peer, the connection status, and the number of active sources.

show ip pim assert

Display information about asserts in the PIM system for S,G mroutes. This command does not show S,G Channel states that in a NOINFO state.

show ip pim assert-internal

Display internal assert state for S,G mroutes

show ip pim assert-metric

Display metric information about assert state for S,G mroutes

show ip pim assert-winner-metric

Display winner metric for assert state for S,G mroutes

show ip pim group-type

Display SSM group ranges.

show ip pim interface

Display information about interfaces PIM is using.

show ip pim mlag [vrf NAME|all] interface [detail|WORD] [json]

Display mlag interface information.

show ip pim join

Display information about PIM joins received. If one address is specified then we assume it is the Group we are interested in displaying data on. If the second address is specified then it is Source Group.

show ip pim local-membership

Display information about PIM interface local-membership.

show ip pim mlag summary [json]

Display mlag information state that PIM is keeping track of.

show ip pim neighbor

Display information about PIM neighbors.

show ip pim nexthop

Display information about pim nexthops that are being used.

show ip pim nexthop-lookup

Display information about a S,G pair and how the RPF would be chosen. This is especially useful if there are ECMP's available from the RPF lookup.

show ip pim [vrf NAME] rp-info [A.B.C.D/M] [json]

Display information about RP's that are configured on this router.

You can filter the output by specifying an arbitrary group range.

```
soodar# show ip pim rp-info
RP address       group/prefix-list   OIF               I am RP    Source   Group-
↪Type
192.168.10.123   225.0.0.0/24        ge2               yes        Static   ASM
192.168.10.123   239.0.0.0/8         ge2               yes        Static   ASM
192.168.10.123   239.4.0.0/24        ge2               yes        Static   SSM
```

(continues on next page)

(continued from previous page)

```
soodar# show ip pim rp-info 239.4.0.0/25
RP address       group/prefix-list   OIF               I am RP    Source   Group-
↪Type
192.168.10.123   239.0.0.0/8         ge2               yes        Static   ASM
192.168.10.123   239.4.0.0/24        ge2               yes        Static   SSM
```

show ip pim rpf

Display information about currently being used S,G's and their RPF lookup information. Additionally display some statistics about what has been happening on the router.

show ip pim secondary

Display information about an interface and all the secondary addresses associated with it.

show ip pim state

Display information about known S,G's and incoming interface as well as the OIL and how they were chosen.

show ip pim [vrf NAME] upstream [A.B.C.D [A.B.C.D]] [json]

Display upstream information about a S,G mroute. Allow the user to specify sub Source and Groups that we are only interested in.

show ip pim upstream-join-desired

Display upstream information for S,G's and if we desire to join the multicast tree

show ip pim upstream-rpf

Display upstream information for S,G's and the RPF data associated with them.

show ip pim [vrf NAME] mlag upstream [A.B.C.D [A.B.C.D]] [json]

Display upstream entries that are synced across MLAG switches. Allow the user to specify sub Source and Groups address filters.

show ip pim mlag summary

Display PIM MLAG (multi-chassis link aggregation) session status and control message statistics.

show ip pim bsr

Display current bsr, its uptime and last received bsm age.

show ip pim bsrp-info

Display group-to-rp mappings received from E-BSR.

show ip pim bsm-database

Display all fragments of stored bootstrap message in user readable format.

mtrace A.B.C.D [A.B.C.D]

Display multicast traceroute towards source, optionally for particular group.

show ip multicast count [vrf NAME] [json]

Display multicast data packets count per interface for a vrf.

show ip multicast count vrf all [json]

Display multicast data packets count per interface for all vrf.

PIM Debug Commands

The debugging subsystem for PIM behaves in accordance with how FRR handles debugging. You can specify debugging at the enable CLI mode as well as the configure CLI mode. If you specify debug commands in the configuration cli mode, the debug commands can be persistent across restarts of the FRR pimd if the config was written out.

debug igmp

This turns on debugging for IGMP protocol activity.

debug mtrace

This turns on debugging for mtrace protocol activity.

debug mroute

This turns on debugging for PIM interaction with kernel MFC cache.

debug pim events

This turns on debugging for PIM system events. Especially timers.

debug pim nht

This turns on debugging for PIM nexthop tracking. It will display information about RPF lookups and information about when a nexthop changes.

debug pim nht detail

This turns on debugging for PIM nexthop in detail. This is not enabled by default.

debug pim packet-dump

This turns on an extraordinary amount of data. Each pim packet sent and received is dumped for debugging purposes. This should be considered a developer only command.

debug pim packets

This turns on information about packet generation for sending and about packet handling from a received packet.

debug pim trace

This traces pim code and how it is running.

debug pim bsm

This turns on debugging for BSR message processing.

debug pim zebra

This gathers data about events from zebra that come up through the ZAPI.

PIM Clear Commands

Clear commands reset various variables.

clear ip interfaces

Reset interfaces.

clear ip igmp interfaces

Reset IGMP interfaces.

clear ip mroute

Reset multicast routes.

clear ip mroute [vrf NAME] count

When this command is issued, reset the counts of data shown for packet count, byte count and wrong interface to 0 and start count up from this spot.

clear ip pim interfaces

Reset PIM interfaces.

clear ip pim oil

Rescan PIM OIL (output interface list).

clear ip pim [vrf NAME] bsr-data

This command will clear the BSM scope data struct. This command also removes the next hop tracking for the bsr and resets the upstreams for the dynamically learnt RPs.

1.5.11 Policy Based Routing

PBR (Policy-Based Routing) is a powerful feature available in modern routers that allows network administrators to exert granular control over the routing decisions made within their network. Unlike traditional routing, which relies solely on destination IP addresses to determine the path packets take, PBR enables routing decisions based on policies defined by network administrators.

Policy-Based Routing operates on the principle of defining policies that match specific criteria within packets, such as source IP addresses, application types, or QoS markings. When a packet matches a defined policy, it is directed to follow a custom routing path, bypassing the regular routing table.

PBR can be employed in various scenarios, including:

a. Load balancing traffic across multiple ISP connections.

b. Implementing security measures by directing traffic through a firewall.

c. Redirecting traffic to optimize network performance.

PBR Configuration

To configure a PBR, we need to define the policies. *Route Maps* are used to define the policy-based routing criteria. You can create a route-map with the route-map command followed by a name. Let's call it "PBR-Map" in this example:

```
Router(config)# route-map PBR-Map permit 10
```

- `PBR-Map`: This is the name of the route-map. You can choose any name you prefer.
- `permit 10`: This is a sequence number that determines the order of evaluation. Lower sequence numbers are evaluated first.

Now in our created route-map we define the match criteria. Specify the matching criteria for the policy using the match command within the route-map. This criteria defines what kind of traffic the policy will match.

Note: Only `match ip address ACL_NAME` command could be used for a PBR in the route-map.

```
Router(config-route-map)# match ip address allowed_sources
```

- `ip address allowed_sources`: This specifies that the route-map should match packets that match an access control list (ACL) named "allowed_sources."

The next step, is defining an action. To specify the action to be taken when the criteria are met, use the set command within the route-map.

Note: Only `set ip next-hop A.B.C.D` command could be used for a PBR in the route-map.

```
Router(config-route-map)# set ip next-hop 192.168.1.1
```

- `ip next-hop 192.168.1.1`: This sets the next-hop IP address for the matched packets.

Note: If not specified, the next-hop is looked up from the default VRF. To change this behaviour, the user can use the `ip next-hop vrf` command.

Finally, you should apply the route-map to the relevant interface, subinterface, or VLAN where you want to implement PBR:

`ip policy route-map PBR-Map`

- `PBR-Map`: Specifies the name of the route-map that contains the policy criteria and actions to be applied to the traffic.

Note: Policies are applied to ingress traffic, so route-map should be set on internal interface(s).

Note: Packets that are generated by the device are not policy-routed.

1.5.12 Example

In this example, we'll configure PBR to route HTTP traffic through a specific gateway while all other traffic uses the regular routing table.

Assumptions:

- The router has two interfaces: `ge0` for the internal network and `ge1` for the external network.
- The gateway for HTTP traffic is *192.168.1.254*, and all other traffic follows the default routing(*192.168.1.1*).

We define an ACL that permits TCP traffic with a source IP address of any internal client and a destination port 80 (HTTP):

```
Router(config)# ip access-list http_traffic
Router(config-nacl)# permit tcp any any eq http
```

A route-map is created with sequence number 10. It matches traffic using ACL *http_traffic* and sets the next-hop IP address to *192.168.1.254*:

```
Router(config)# route-map redirect_http permit 10
Router(config-route-map)# match ip address http_traffic
Router(config-route-map)# set ip next-hop 192.168.1.254
```

The route-map is applied to the internal interface (*ge0*). This means that HTTP traffic will follow the route-map and be forwarded to 192.168.1.254, while all other traffic will use the default route:

```
Router(config)# interface ge0
Router(config-if)# ip policy route-map redirect_http
Router(config)# ip route 0.0.0.0/0 192.168.1.1
```

1.6 IP Routing Manager

1.6.1 IP Routing Manager

The IRM (IP Routing Manager) (also known as Zebra), is a daemon responsible for managing the routing table and communicating with the dataplane to install and withdraw routes and other routing/interface options.

The IP Routing Manager provides a unified interface for different routing protocols and allows for dynamic route updates based on protocol-specific events. It supports a variety of routing protocols such as OSPF, BGP, RIP, and IS-IS, as well as other features like route filtering, and route redistribution.

Interface Commands

Standard Commands

`interface IFNAME`

The interface command is used to enter interface configuration mode for a specific network interface.

- `IFNAME`: Specifies the name of the interface to be configured.

Example:

```
soodar(config)# interface ge0
soodar(config-if)#
```

This command allows the user to configure the settings for the ge0 interface, such as IP address and interface features like Quality of Service (QoS), MPLS, and so on. The specific commands available in interface configuration mode depend on the type of interface being configured.

`shutdown`

The command is used to disable a specific interface on a network device. When an interface is shut down, it stops transmitting and receiving traffic. To re-enable the interface, use the `no shutdown` command in interface configuration mode.

Note: It is important to note that shutting down an interface will cause any routes that depend on that interface to be removed from the routing table. Additionally, any neighbor relationships established on that interface will also be lost. Therefore, it is recommended to use the shutdown command with caution and only when necessary.

`ip address ADDRESS/PREFIX`

The command is used to assign an IP address and subnet mask to an interface in a device.

- `ADDRESS/PREFIX`: specifies the IP address and subnet mask for the interface in CIDR notation.

Here's an example of how to assign an IP address to an interface:

```
soodar(config)# interface ge0
soodar(config-if)# ip address 192.168.1.1/24
```

This would assign the IP address 192.168.1.1 with a prefix length of 24 to the ge0 interface.

ipv6 address ADDRESS/PREFIX [eui-64]

The command is used to assign an IPv6 address and subnet mask to an interface in a device.

- `ADDRESS/PREFIX`: specifies the IPv6 address and subnet mask for the interface in CIDR notation.
- `eui-64`: (Optional) This keyword enables the automatic configuration of the interface ID using the EUI-64 method. This method uses the MAC address of the device to construct the interface ID portion of the IPv6 address.

ip address LOCAL-ADDR peer PEER-ADDR/PREFIX

The command is used in a point-to-point link configuration to specify the IP addresses for the local and remote ends of the link. (The concept of PtP addressing does not exist for IPv6). The command is used on the local device to assign an IP address to the local interface of the link, and to specify the IP address of the remote end of the link.

- `LOCAL-ADDR`: the IP address to assign to the local interface of the link.
- `peer PEER-ADDR/PREFIX`: specifies the IP address of the remote end of the link. PEER-ADDR is the IP address of the remote end and PREFIX specifies the length of the prefix in bits.

Note: local address has no subnet mask since the local side in PtP addressing is

always a single (/32) address.

Note: peer address and prefix can be an arbitrary subnet behind the other end of the link (or even on the link in Point-to-Multipoint setups), though generally /32s are used.

Note: This command is commonly used in point-to-point links such as tunnels.

Example:

```
soodar(config)# interface tunnel10
soodar(config-if)#tunnel source 200.1.2.1
soodar(config-if)#tunnel destination 200.1.2.2
soodar(config-if)#ip address 10.1.1.1 peer 10.1.1.2/32
```

For example, these commands would create a GRE tunnel and assign the IP address 10.1.1.1/32 to the local interface and specify that the remote end of the link has the IP address 10.1.1.2/32.

mac-address X:X:X:X:X:X

The command is used to configure the Media Access Control (MAC) address for that interface. The `no` form of this command reset the interface MAC address to the original one.

- `X:X:X:X:X:X`: Represents the MAC address you want to assign to the interface. A MAC address consists of six pairs of hexadecimal digits separated by colons (e.g., 00:1A:2B:3C:4D:5E).

Note: This MAC address can't be multicast or zero address.

Note: Manually changing a MAC address should be done with caution, as MAC addresses are typically unique

and play a role in device identification on a network. Changing a MAC address can have network-related consequences.

description DESCRIPTION ...

The command is used to add a description to an interface. This command allows the network administrator to add a human-readable description to the interface configuration, which can help identify the interface's purpose or its location.

- DESCRIPTION: is the description text, which can be up to 240 characters long

For example, the following command sets a description for interface ge0:

```
soodar(config)# interface ge0
soodar(config-if)# description Main office LAN
```

This will set the description for ge0 to "Main office LAN". The description will appear in the output of commands like "show interface" and can be used to help identify the interface's function.

multicast

Enable or disables multicast flag for the interface.

bandwidth (1-10000000)

The bandwidth command is used to manually set the bandwidth value for an interface. The bandwidth value is a measure of the speed of the interface and is used by routing protocols to calculate the best path for traffic. By default, the bandwidth of an interface is calculated based on its physical characteristics, but the bandwidth command can be used to manually set the value.

- (1-10000000): is the bandwidth value to be set in kilobits per second (kbps). The valid range is 1 to 10,000,000 kbps.

Note: Setting the bandwidth manually may affect the routing decisions made by the router and should be used with caution. It is recommended to leave the bandwidth value to be calculated automatically unless there is a specific reason to override it.

Note: This command does not affect the actual device configuration.

link-detect

The command enables the automatic detection of the physical link state changes on an interface. When this command is enabled, the device will monitor the status of the physical link and take appropriate action when changes are detected. This command can be useful for monitoring and managing the connectivity between devices and for quickly identifying and troubleshooting link failures.

Note: the "link-detect" command is automatically enabled by default on all interfaces. It can be disabled using the "no link-detect" command.

URPF

URPF (Unicast Reverse-Path Forwarding) is a security mechanism used to validate the source address of incoming traffic to a network. It helps to prevent attacks where an attacker spoofs the source IP address of their traffic to make it appear to come from a trusted source. URPF filters out traffic that is not from a valid source IP address, thereby reducing the possibility of network attacks.

URPF works by comparing the source IP address of incoming traffic to the routing table of the network. If the source IP address is not found in the routing table, the traffic is discarded. URPF can be configured in two modes: `strict mode` and `loose mode`. In strict mode, only traffic with a source IP address that is reachable through the receiving interface is allowed. In loose mode, traffic with a source IP address that can be reached through any interface is allowed.

`ip verify unicast source reachable-via [rx | any]`

The ip verify unicast command is used to enable Unicast Reverse Path Forwarding (uRPF) on an interface.

- `rx`: This option specifies that the source IP address should be reachable through the received interface(Strict mode).
- `any`: This option specifies that the source IP address should be reachable(Loose mode).

Administrative Distance

Administrative distance allows IRM to make decisions about what routes should be installed in the rib based on the originating protocol. The lowest Admin Distance is the route selected. This is purely a subjective decision about ordering and care has been taken to choose the same distances that other routing suites have chosen.

Protocol	Distance
System	0
Connect	0
Static	1
Wireguard	1
EBGP	20
EIGRP	90
OSPF	110
ISIS	115
RIP	120
IBGP	200

An admin distance of 255 indicates to IRM that the route should not be installed into the Data Plane. Additionally routes with an admin distance of 255 will not be redistributed.

Virtual Routing and Forwarding

See also:

VRF

`show ip route vrf VRF`

The command is used to display the IP routing table for a specific VRF (Virtual Routing and Forwarding) instance. This command is useful when troubleshooting network connectivity issues or verifying the routing table on a particular VRF.

- `VRF`: Specifies the name of the VRF for which the routing table is to be displayed.

When this command is executed, the output will display all the routes that are installed in the routing table of the specified VRF, along with the next-hop address, the metric, the administrative distance, and the route type.

show <ip|ipv6> route summary [vrf VRF] [prefix]

The command displays a summary of the routing table entries.

- `vrf VRF`: This parameter is optional and specifies the VRF table to display the summary for.
- `prefix`: This parameter is also optional and limits the summary to the specified IP prefix.

ECMP

Soodar supports ECMP as part of normal operations and is generally compiled with a limit of 64 way ECMP. Individual protocols each have their way of dictating ECMP policy and their respective documentation should be read.

ECMP can be inspected in IRM by doing a *show ip route X* command.

```
soodar# show ip route 4.4.4.4/32
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP,
       F - PBR, f - OpenFabric,
       > - selected route, * - FIB route, q - queued, r - rejected, b - backup
       t - trapped, o - offload failure

S>* 4.4.4.4/32 [150/0] via 192.168.161.1, ge0, weight 1, 00:00:02
  *                    via 192.168.161.2, ge0, weight 1, 00:00:02
  *                    via 192.168.161.3, ge0, weight 1, 00:00:02
  *                    via 192.168.161.4, ge0, weight 1, 00:00:02
  *                    via 192.168.161.5, ge0, weight 1, 00:00:02
  *                    via 192.168.161.6, ge0, weight 1, 00:00:02
  *                    via 192.168.161.7, ge0, weight 1, 00:00:02
  *                    via 192.168.161.8, ge0, weight 1, 00:00:02
  *                    via 192.168.161.9, ge0, weight 1, 00:00:02
  *                    via 192.168.161.10, ge0, weight 1, 00:00:02
  *                    via 192.168.161.11, ge0, weight 1, 00:00:02
  *                    via 192.168.161.12, ge0, weight 1, 00:00:02
  *                    via 192.168.161.13, ge0, weight 1, 00:00:02
  *                    via 192.168.161.14, ge0, weight 1, 00:00:02
  *                    via 192.168.161.15, ge0, weight 1, 00:00:02
  *                    via 192.168.161.16, ge0, weight 1, 00:00:02
```

In this example, we have a 16-way ECMP for the 4.4.4.4/32 route. The * character tells us that the route is installed in the Data Plane or FIB.

MPLS Commands

You can configure static MPLS entries in IRM. Handling MPLS consists of popping, swapping or pushing labels to IP packets.

MPLS Acronyms

LSR (Labeled Switch Router)
Networking devices handling labels used to forward traffic between and through them.

LER (Labeled Edge Router)
A Labeled edge router is located at the edge of an MPLS network, generally between an IP network and an MPLS network.

MPLS Push Action

The push action is generally used for LER devices, which want to encapsulate all traffic for a wished destination into an MPLS label. This action is stored in routing entry, and can be configured like a route:

`ip route NETWORK/PREFIX GATEWAY|INTERFACE label LABEL`

This command is used to add a MPLS label to an IP route.

- `NETWORK/PREFIX`: The destination network for this route in IPv4 CIDR notation.
- `GATEWAY`: The IP address of the next-hop router for this route.
- `INTERFACE`: The interface name for this route. This is used when the next-hop router is directly connected to the local router.
- `LABEL`: is the MPLS label to use to reach the prefix abovementioned.

You can check that the static entry is stored in the IRM RIB database, by looking at the presence of the entry.

```
soodar(config)# ip route 1.1.1.1/32 10.0.1.1 label 777
soodar# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
T - Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP,
F - PBR,
> - selected route, * - FIB route

S>* 1.1.1.1/32 [1/0] via 10.0.1.1, ge1, label 777, 00:39:42
```

MPLS Swap and Pop Action

The swap action is generally used for LSR devices, which swap a packet with a label, with an other label. The Pop action is used on LER devices, at the termination of the MPLS traffic; this is used to remove the MPLS header.

You can check that the MPLS actions are stored in the IRM MPLS table, by looking at the presence of the entry.

`show mpls table`

The command is used to display the MPLS forwarding table or label-switching table (LSP). The MPLS forwarding table contains information about the incoming and outgoing labels for a given destination prefix or network.

```
soodar(confi)# mpls lsp 18 10.125.0.2 implicit-null
soodar(confi)# mpls lsp 19 10.125.0.2 20
soodar(confi)# mpls lsp 21 10.125.0.2 explicit-null
soodar# show mpls table
Inbound                              Outbound
Label     Type          Nexthop      Label
--------  -------  ---------------  --------
18     Static       10.125.0.2  implicit-null
19     Static       10.125.0.2  20
21     Static       10.125.0.2  IPv4 Explicit Null
```

IRM Route Filtering

IRM supports *prefix-list* s and *Route Maps* s to match routes received from other Soodar components. The permit/deny facilities provided by these commands can be used to filter which routes IRM will install in the Data plane.

ip protocol PROTOCOL route-map ROUTEMAP

Apply a route-map filter to routes for the specified protocol. PROTOCOL can be:

- any,
- bgp,
- connected,
- eigrp,
- isis,
- ospf,
- ospf6,
- rip,
- static,
- ripng

If you choose any as the option that will cause all protocols that are sending routes to IRM. You can specify a *ip protocol PROTOCOL route-map ROUTEMAP* on a per vrf basis, by entering this command under vrf mode for the vrf you want to apply the route-map against.

set src ADDRESS

Within a route-map, set the preferred source address for matching routes when installing in the data plane.

The following creates a prefix-list that matches all addresses, a route-map that sets the preferred source address, and applies the route-map to all *rip* routes.

```
soodar(config)# ip prefix-list ANY permit 0.0.0.0/0 le 32
soodar(config)# route-map RM1 permit 10
soodar(config-route-map)# match ip address prefix-list ANY
soodar(config-route-map)# set src 10.0.0.1
soodar(config)# ip protocol rip route-map RM1
```

IPv6 example for OSPFv3.

```
soodar(config)# ipv6 prefix-list ANY seq 10 permit any
soodar(config)# route-map RM6 permit 10
soodar(config-route-map)# match ipv6 address prefix-list ANY
soodar(config-route-map)# set src 2001:db8:425:1000::3
soodar(config)# ipv6 protocol ospf6 route-map RM6
```

zebra route-map delay-timer (0-600)

Set the delay before any route-maps are processed in IRM. The default time for this is 5 seconds.

IRM Terminal Mode Commands

show ip route

Display current routes which IRM holds in its database.

```
Router# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
 B - BGP * - FIB route.

S* 0.0.0.0/0         203.181.89.1
C* 127.0.0.0/8       loopback0
C* 203.181.89.240/28       ge0
```

show ipv6 route

show [ip|ipv6] route [PREFIX] [nexthop-group]

Display detailed information about a route. If [nexthop-group] is included, it will display the nexthop group ID the route is using as well.

show interface [NAME] [{vrf VRF|brief}] [json]

show interface [NAME] [{vrf all|brief}] [json]

show interface [NAME] [{vrf VRF|brief}] [nexthop-group]

show interface [NAME] [{vrf all|brief}] [nexthop-group]

Display interface information. If no extra information is added, it will dump information on all interfaces. If [NAME] is specified, it will display detailed information about that single interface. If [nexthop-group] is specified, it will display nexthop groups pointing out that interface.

If the `json` option is specified, output is displayed in JSON format.

show ip prefix-list [NAME]

show route-map [NAME]

show zebra

Display various statistics related to the installation and deletion of routes, neighbor updates, and LSP's into the kernel.

show zebra client [summary]

Display statistics about clients that are connected to IRM. This is useful for debugging and seeing how much data is being passed between IRM and it's clients. If the summary form of the command is choosen a table is displayed with shortened information.

show zebra router table summary

Display summarized data about tables created, their afi/safi/tableid and how many routes each table contains. Please note this is the total number of route nodes in the table. Which will be higher than the actual number of routes that are held.

Router-id

Many routing protocols require a router-id to be configured. To have a consistent router-id across all daemons, the following commands are available to configure and display the router-id:

[ip] router-id A.B.C.D

Allow entering of the router-id. This command also works under the vrf subnode, to allow router-id's per vrf.

[ip] router-id A.B.C.D vrf NAME

Configure the router-id of this router from the configure NODE. A show run of this command will display the router-id command under the vrf sub node. This command is deprecated and will be removed at some point in time in the future.

show [ip] router-id [vrf NAME]

Display the user configured router-id.

For protocols requiring an IPv6 router-id, the following commands are available:

ipv6 router-id X:X::X:X

Configure the IPv6 router-id of this router. Like its IPv4 counterpart, this command works under the vrf subnode, to allow router-id's per vrf.

show ipv6 router-id [vrf NAME]

Display the user configured IPv6 router-id.

Debugging

debug zebra mpls [detailed]

MPLS-related events and information.

debug zebra events

IRM events

debug zebra pseudowires

Pseudowire events.

debug zebra packet [<recv|send>] [detail]

ZAPI message and packet details

debug zebra kernel

Kernel / OS events.

debug zebra kernel msgdump [<recv|send>]

Raw OS (netlink) message details.

debug zebra rib [detailed]

RIB events.

debug zebra dplane [detailed]

Dataplane / FIB events.

1.7 NAT

1.7.1 NAT

NAT (Network Address Translation) stands for Network Address Translation. It is a technique used in computer networking to allow devices in a private network to access resources on a public network, such as the Internet. NAT modifies the source IP addresses of outgoing packets and the destination IP addresses of incoming packets, allowing them to be properly routed between private and public networks.

NAT works by mapping private IP addresses to public IP addresses. When a device in the private network sends a packet to a device on the public network, the NAT device replaces the source IP address of the packet with its own public IP address. When a packet is sent from a device on the public network to a device in the private network, the NAT device replaces the destination IP address with the private IP address of the destination device.

NAT can be used to allow multiple devices in a private network to share a single public IP address. This is known as NAT overload or Port Address Translation (PAT). In this scenario, the NAT device keeps track of the source ports of outgoing packets and the destination ports of incoming packets, allowing multiple devices to use the same public IP address at the same time.

NAT can also be used to provide security by hiding the private IP addresses of devices in a network from the public network. This is known as NAT hiding or NAT firewalling. In this scenario, the NAT device only allows incoming packets that are in response to outgoing packets from devices on the private network. This prevents unsolicited incoming traffic from reaching devices in the private network.

SoodarOS uses PNAT variation, which changes Port and Address.

NAT Static Mapping

A static NAT mapping is a one-to-one mapping of a public IP address to a private IP address, allowing an internal host to be accessible from the public network with a public IP address. With a static NAT mapping, any incoming traffic destined for the public IP address is automatically forwarded to the corresponding private IP address.

Static NAT mappings can also be configured to translate specific TCP or UDP ports, allowing multiple internal hosts to share a single public IP address.

Address only NAT

In this mode, only the address is translated to the given address. Depending on the flow direction(whether in2out or out2in), the source or destination of the packet is changed.

When a packet containing a *source* same as *local address* passes through an *input* interface, its *source* is replaced with *global address*. When a packet containing a *destination* similar to *global address* passes through an *output* interface, its *destination* is replaced with *local address*.

ip nat inside source static A.B.C.D A.B.C.D

Add a new static map entry to the NAT static table. The first IP is a local address, and the second is a global address.

Example :

```
soodar(config)# ip nat inside source static 192.168.1.10 85.20.1.1
soodar(config)# interface ge0
soodar(config-if)# ip nat inside
soodar(config)# interface ge1
soodar(config-if)# ip nat outside
```

Define a static map entry that translates every ingress traffic from *ge0* sourced from *192.168.1.10* to *85.20.1.1* (Also known as *Source NAT*). Every packet coming from *ge1*, which is destined to *85.20.1.1*, is also translated to *192.168.1.10*

```
soodar(config)# ip nat inside source static 85.20.1.1 192.168.1.10
soodar(config)# interface ge0
soodar(config-if)# ip nat outside
soodar(config)# interface ge1
soodar(config-if)# ip nat inside
```

Define a static map entry that translates every ingress traffic from *ge0* destined to *192.168.1.10* to *85.20.1.1* (Also known as *Destination NAT*). Every packet coming from *ge1*, which is sourced from *85.20.1.1*, is also translated to *192.168.1.10*

Protocol NAT

Sometimes we need to be more specific about our NAT and translate a specified protocol on a defined port. So when defining an entry, we introduce the protocol and desired ports. All other aspects of this entry(including behavior) are simple *Address only NAT*.

ip nat inside source static <tcp|udp> A.B.C.D (1-65535) A.B.C.D (1-65535)

Add a new static map entry to the NAT static table. The first IP is a local address, and the number following is its port. The second IP is a global address, and the number following is its port.

Example :

```
soodar(config)# ip nat inside source static tcp 192.168.1.10 444 85.20.1.1 666
soodar(config)# interface ge0
soodar(config-if)# ip nat inside
soodar(config)# interface ge1
soodar(config-if)# ip nat outside
```

Define a static map entry that translates every ingress traffic from *ge0* sourced from *192.168.1.10:444* to *85.20.1.1:666* (Also known as *Source NAT*). Every packet coming from *ge1*, which is destined to *85.20.1.1:666*, is also translated to *192.168.1.10:444*

```
soodar(config)# ip nat inside source static tcp 85.20.1.1 666 192.168.1.10 444
soodar(config)# interface ge0
soodar(config-if)# ip nat outside
soodar(config)# interface ge1
soodar(config-if)# ip nat inside
```

Define a static map entry that translates every ingress traffic from *ge0* destined to *192.168.1.10:444* to *85.20.1.1:666* (Also known as *Destination NAT*). Every packet coming from *ge1*, which is sourced from *85.20.1.1:666*, is also translated to *192.168.1.10:444*

Dynamic NAT

In dynamic NAT, every packet's source outgoing from an *input* interface and destined to an *output* interface is translated to an IP provided by an IP pool.

A new session is created for every *source translation*, and its state is kept. So the packets coming from an *output* interface and having a matched session, its destination is changed concerning the session's information.

IP pool

An IP NAT (Network Address Translation) pool is a collection of public IP addresses that can map private IP addresses to public IP addresses. When a private IP address needs to communicate with a device outside of its local network, NAT is used to translate the private IP address to a public IP address, allowing communication to take place.

A NAT pool contains a range of public IP addresses that are used for NAT translations. The router selects an IP address from the pool for each translation. The selection is performed using random selection methods.

IP NAT pools are commonly used in network environments that use private IP addresses internally, such as those using **RFC 1918** address space. They are also used in cases where multiple devices need to share a limited number of public IP addresses, such as in a small office or home network. By using an IP NAT pool, many devices can use a single public IP address to communicate with external networks.

ip nat pool PNAT44 A.B.C.D [A.B.C.D] [type <normal|lb>]

Add an IP(or an IP range if the second IP is provided) to a nat pool named PNAT44 Creates a pool containing an IP(or a range of IPs). There are two types of pools: normal and load-balanced. The normal type is the default and provides a simple pool of IP addresses for NAT translations. The lb (load-balanced) type creates a pool of IP addresses that can be used for load balancing between multiple servers or hosts.

Example :

```
soodar(config)# ip nat pool p1 1.1.1.1
soodar(config)# ip nat pool p2 2.1.1.1 2.1.1.10
```

The first command is to create *p1* nat pool and add IP *1.1.1.1* to it. the second one adds *2.1.1.1* to *2.1.1.10* to *p2* nat pool.

Configuring dynamic NAT

ip nat inside source list ACL4 pool PNAT44 [<match-in-vrf|vrf VRF>]

The ip nat inside source command is used to configure NAT to translate inside source IP addresses to public IP addresses from a NAT pool.

The list ACL4 part of the command specifies the access control list (ACL) to identify the inside source traffic to translate. The ACL should be configured to match the flows that need to be translated.

The PNAT44 specifies the NAT pool to use for the translation. This pool should have been previously created using the ip nat pool command.

The `match-in-vrf` or `vrf VRF` is an optional parameter that allows you to specify a VRF (virtual routing and forwarding) instance to match for the after translation outgoing traffic. If the `match-in-vrf` keyword is specified, the VRF of the inside interface will be used. If the `vrf VRF` keyword is specified, the VRF specified will be used.

Configuring dynamic NAT using interface

ip nat inside source list ACL4 interface IFNAME [<match-in-vrf|vrf VRF>]

The ip nat inside source command is used to configure NAT to translate inside source IP addresses to the first IP addresses of the specified interface.

The list ACL4 part of the command specifies the access control list (ACL) to identify the inside source traffic to translate. The ACL should be configured to match the flows that need to be translated.

The IFNAME specifies the interface to use its IP for the translation.

The `match-in-vrf` or `vrf VRF` is an optional parameter that allows you to specify a VRF instance to match for the after translation outgoing traffic. If the `match-in-vrf` keyword is specified, the VRF of the inside interface will be used. If the `vrf VRF` keyword is specified, the VRF specified will be used.

Load balancing with NAT

NAT load balancing is a technique that is used to distribute incoming traffic across multiple servers in a network. It is achieved by configuring a router to perform NAT with load balancing.

NAT load balancing works by assigning a public IP address to a group of private IP addresses on the network. When incoming traffic arrives at the router, the router examines the destination IP address and port number. It maps it to one of the private IP addresses in the pool using a round-robin algorithm. The router then performs NAT, replacing the destination IP address in the packet header with the selected private IP address, and forwards the packet to the appropriate server.

NAT load balancing can be used for various applications, such as web servers, email servers, and other applications that require high availability and scalability. It also allows traffic from external networks to access specific devices on your network that are hidden behind a NAT router.

ip nat inside destination <tcp|udp> A.B.C.D (1-65535) pool PNAT44

The command is used to configure load balancing and destination translation for Network Address Translation (NAT). This command allows you to specify a protocol (TCP or UDP), a public IP address, a destination port number, and a pool of private servers' IP addresses to which traffic should be forwarded.

The `tcp|udp` part of the command specifies the incoming traffic is using the TCP or UDP protocol.

The `A.B.C.D` part of the command is the public IP address to which traffic is destined. It is the public IP address of the device on your network that you want to receive the traffic.

The (1-65535) part of the command is the destination port number on which traffic will be received.

The `PNAT44` is the name of the NAT pool that you want to use for the destination translation. This pool of private IP addresses will be used to forward traffic to servers.

Example:

Let's say we have two web servers in our private network, with IP addresses 192.168.1.10 and 192.168.1.11, and we want to load-balance incoming HTTP traffic to them using the public IP address 203.0.113.10. We can configure NAT with load-balancing on using the following commands:

First, we'll create a NAT pool with the IP addresses of our web servers:

```
soodar(config)# ip nat pool web-servers 192.168.1.10 192.168.1.11 type lb
```

Then, we'll configure NAT to translate the destination IP address of incoming HTTP traffic to the IP address of one of our web servers:

```
soodar(config)# ip nat inside destination tcp 203.0.113.10 80 pool web-servers
```

Put interface behind NAT

ip nat inside

Define an interface as a NAT inside interface.

ip nat outside

Define an interface as a NAT outside interface.

NAT Forwarding

When a packet arrives at an inside or an outside interface, Dataplane looks for a translation to use or create. If none is found, the packet is dropped. Admin can change this behavior and forward them like normal packets on non-NAT-enabled interfaces instead of dropping them.

ip nat forwarding

Enable NAT forwarding feature on device.

Clearing Translations

clear ip nat translation inside A.B.C.D [outside A.B.C.D]

This command is used to clear Network Address Translation (NAT) translations for a specific inside IP address and, optionally, a specific outside IP address.

Example:

```
soodar# clear ip nat translation inside 192.168.1.100
```

This command will clear all NAT translations for the inside IP address 192.168.1.100. If there are any active NAT translations for this address, they will be removed, allowing new translations to be established.

```
soodar# clear ip nat translation inside 192.168.1.100 outside 203.0.113.10
```

This command will clear all NAT translations for traffic originating from inside IP address 192.168.1.100 and being translated to outside IP address 203.0.113.10.

clear ip nat translation *

This command is used to clear all Network Address Translation (NAT) translations on the router.

clear ip nat translation tcp inside A.B.C.D [(1-65535) outside A.B.C.D (1-65535)]

The clear ip nat translation command is used to clear Network Address Translation (NAT) translations. The tcp keyword indicates that only TCP translations will be cleared.

Example:

If we want to clear TCP NAT translations for an inside IP address of 192.168.1.10 on port 80, communicating with an outside IP address of 203.0.113.10 on any port, we would use the following command:

```
soodar# clear ip nat translation tcp inside 192.168.1.10 80 outside 203.0.113.10
```

clear ip nat translation udp inside A.B.C.D [(1-65535) outside A.B.C.D (1-65535)]

The clear ip nat translation command is used to clear Network Address Translation (NAT) translations. The tcp keyword indicates that only UDP translations will be cleared.

clear ip nat translation icmp inside A.B.C.D [(1-65535) outside A.B.C.D (1-65535)]

The clear ip nat translation command is used to clear Network Address Translation (NAT) translations. The tcp keyword indicates that only ICMP translations will be cleared.

Debugging

Debugging logs can be set in case of need.

debug nat44 event

log data plane installation processes and results

show ip nat statistics

show statistics about translations and current NAT configuration

```
soodar# show ip nat statistics
Total active translations: 4 (1 static, 3 dynamic)
Outside interfaces:
  ge1
Inside interfaces:
  ge0
NAT Forwarding: Disabled
```

show ip nat translations

Show current active translations

```
soodar# show ip nat translations
 Pro     Inside Local     Inside Global   Outside Local   Outside Global
 ------------------------------------------------------------------------
 ---          1.1.1.10         200.2.3.3             ---              ---
 ICMP      1.1.1.10:48      200.2.3.3:48     2.1.1.10:48      2.1.1.10:48
 TCP    1.1.1.10:46122   200.2.3.3:46122   2.1.1.10:5201    2.1.1.10:5201
 TCP    1.1.1.10:46120   200.2.3.3:46120   2.1.1.10:5201    2.1.1.10:5201
 ICMP      1.1.1.10:45   200.2.3.3:63327     2.1.1.10:45      2.1.1.10:45

Total number of translations: 4
```

Example configuration

```
soodar(config)# int ge0
soodar(config-if)#  ip nat outside
soodar(config)# int ge2
soodar(config-if)#  ip nat inside
soodar(config)# ip nat pool nat1 200.1.2.1
```

1.8 Qos

1.8.1 QoS

QoS, or Quality of Service, is a networking concept that refers to the ability of a network to provide different levels of service to different types of traffic. It is a way to prioritize certain types of traffic over others based on their importance or the requirements of the applications using the network.

QoS allows administrators to manage network resources effectively by controlling the amount of bandwidth allocated to specific applications or devices, and by ensuring that critical applications receive the necessary bandwidth to function properly.

QoS can be implemented in different ways, such as by setting priorities for different types of traffic, limiting the bandwidth available for certain applications, or using queuing and scheduling algorithms to manage network traffic. The goal is to optimize network performance, reduce latency and packet loss, and ensure that critical applications and services operate smoothly.

QoS is particularly important in networks that carry real-time traffic such as voice or video, which require low latency and minimal packet loss to function properly. Without QoS, other types of traffic such as file transfers or software updates can cause congestion and negatively impact the performance of real-time applications.

Class Map

In QoS (Quality of Service), a class map is used to define a traffic class. It is a configuration construct used to classify traffic on the basis of various parameters like IP address, protocol, or port number. Class maps help to identify traffic that needs to be treated in a particular way. For example, traffic from a specific application can be marked as a high priority while traffic from other applications can be marked as a low priority.

Once traffic is classified using class maps, policies can be applied to each class to ensure that the traffic is handled according to its requirements.

Define Class Map

`class-map match-all CNAME`

The command is used to create a class map. The `match-all` keyword indicates that all of the specified match criteria must be met in order for traffic to be classified into the corresponding class.

- `CNAME`: is the name of the class map, which is used to reference the class in other QoS configuration commands

`class-map match-any CNAME`

The command is used in QoS (Quality of Service) configuration on devices to create a class map that matches any one of the specified match criteria. It is used to group multiple match criteria with an *OR* logic.

- `CNAME`: is the name you give to the class map.

`no class-map CNAME`

Removes a class map

Example :

```
soodar(config)# class-map match-all cmap1
soodar(config-cmap)#
```

Define matching criteria

you can use the `match` command to define the match criteria for a class map.

`match any`

Every packet is accpeted.

`match access-list ACL`

The command is used in the configuration of a class map to match packets based on an Access Control List (ACL). When a packet matches the criteria specified in the ACL, it is classified under the specified class map.

- `ACL`: Specifies the name of the Access Control List that contains the matching criteria.

Example:

```
soodar(config)# ip access-list 101
soodar(config-nacl)# permit tcp any any eq 80
soodar(config-nacl)# permit tcp any any eq 443
soodar(config)# class-map match-any WEB_TRAFFIC
soodar(config-cmap)# match access-list 101
```

In the above example, the class map *WEB_TRAFFIC* is configured to match packets based on the Access Control List 101, which permits TCP traffic to ports 80 and 443.

`match source-address A.B.C.D/M`

The command is used to match packets based on their source IP address or IP address range.

- `A.B.C.D/M`: is the source IP address or IP address range in CIDR notation.

This command is typically used in QoS (Quality of Service) configurations to match specific traffic flows based on their source IP address.

Example:

```
soodar(config)# class-map match-all LOCAL
soodar(config-cmap)# match source-address 192.168.1.0/24
```

This will create a class-map called *LOCAL* that matches all traffic with a source IP address in the range of 192.168.1.0 to 192.168.1.255

`match destination-address A.B.C.D/M`

The command is used to define the match criteria for a class-map in devices based on the destination IP address.

- `A.B.C.D/M`: is the destination IP address or IP address range in CIDR notation.

Example:

```
soodar(config)# class-map VOIP
soodar(config-cmap)# match destination-address 10.1.1.0/24

In this example, a class-map named *VOIP* is created to match traffic with a␣
↪destination IP address in the subnet 10.1.1.0/24.
```

`match source-address X:X::X:X/M`

This command specifies a match criterion for IPv6 source addresses within the class map.

- `X:X::X:X/M`: specifies the IPv6 address prefix and mask to match against.

match destination-address X:X::X:X/M

This command specifies a match criterion for IPv6 destination addresses within the class map.

- X:X::X:X/M: specifies the IPv6 address prefix and mask to match against.

match dscp (0-63)

The command is used to match packets based on the Differentiated Services Code Point (DSCP) value in their IP header.

- (0-63): is the decimal value of the DSCP field in the IP header. The DSCP value is a 6-bit value, which means it can range from 0 to 63.

match protocol <(0-255)|PROTOCOLNAME>

The command is used in a class-map to match packets based on the Layer 4 protocol.

- (0-255): Specifies the protocol number, which is an integer between 0 and 255.
- PROTOCOLS: Specifies the name of the protocol.

Note: Note that the match protocol command matches only the Layer 4 protocol and does not look at the content of the packets. It is typically used in combination with other match criteria, such as source and destination IP address, to create a more specific match.

Policy Map

In QoS (Quality of Service), a policy-map is used to apply specific QoS features to specific traffic classes in a class-map. It defines the specific actions that should be taken on the classified traffic.

A policy-map consists of one or more class-maps, and each class-map specifies a particular type of traffic. The policy-map is then applied to an interface to enforce the QoS policies.

Within a policy-map, you can configure various QoS features, such as traffic shaping, bandwidth allocation, queuing, and marking. These features can be tailored to meet specific network requirements and can be used to ensure that critical traffic is prioritized over less important traffic, leading to a more efficient use of network resources.

Note: Currently, only the traffic policing feature is supported.

Define Policy Map

policy-map NAME

The command is used to define a policy map that contains one or more class maps and a set of actions to be taken on the matching traffic. The command takes the user to the policy-map configuration mode, where the actions that should be taken on the matching traffic can be configured.

- NAME: is the name of the policy map that the user wants to create.

Policies brief table

Policy	Guaranteed bandwidth	Maximum bandwidth	Exceed action	Priority
Policing	0	User defined	Drop or mark	No
Shaping	0	User defined	Queue	No
Priority	User defined	User defined	Drop	Yes
Bandwidth	User defined	Infinite	Queue	No

Define a new policy

Traffic policing

Traffic policing is a mechanism used in Quality of Service (QoS) to regulate and control the amount of network traffic that is allowed to pass through a network interface or a specific port. It can be used to ensure that certain types of traffic or specific users do not consume too much bandwidth and negatively impact other users or applications.

Traffic policing works by examining the incoming traffic, comparing it to a configured traffic rate, and either allowing or dropping packets based on the configured rate. If the incoming traffic rate exceeds the allowed rate, the traffic is either dropped or marked with a lower priority. This helps to prevent network congestion and ensures that high-priority traffic is given priority over lower-priority traffic.

Policing can be implemented in different ways, depending on the network topology and requirements. SoodarOS uses token bucket algorithm.

Token bucket algorithm

A token bucket is a traffic management mechanism used to control the rate of data transmission. It is based on the idea of having a token bucket that holds a finite number of tokens. Each token represents a unit of data that can be transmitted, usually measured in bytes. The token bucket is refilled at a certain rate, which is called the committed information rate (CIR). If a device wants to transmit data, it must first obtain a token from the bucket. If there are no tokens available, the device cannot transmit data until more tokens become available.

The implementation of a token bucket involves two key parameters: the bucket size and the token refill rate. The bucket size determines the maximum amount of data that can be transmitted in a given time interval, while the token refill rate determines the rate at which new tokens are added to the bucket.

Single-rate policing and dual-rate policing are two different implementations of the token bucket. Single-rate policing uses a single token bucket to control both the CIR and the excess information rate (EIR), while dual-rate policing uses separate token buckets to control the CIR and EIR.

In single-rate policing, the token bucket is filled at the CIR. Any excess data beyond the CIR is considered to be part of the EIR. The size of the token bucket is set to the committed burst (CB), which is the maximum amount of data that can be transmitted at the CIR. If there are no tokens available in the bucket, excess data is dropped.

In dual-rate policing, there are two token buckets: one for the CIR and one for the EIR. The size of the CIR bucket is set to the committed burst (CB), while the size of the EIR bucket is set to the excess burst (EB). The refill rate for the CIR bucket is set to the CIR, while the refill rate for the EIR bucket is set to the excess information rate (EIR).

The CIR and EIR values are used to define the bandwidth allocation for the traffic being policed. The CB and EB values are used to limit the amount of data that can be transmitted during a given time interval. The token bucket mechanism ensures that the rate of data transmission never exceeds the CIR or EIR, and that the amount of data transmitted does not exceed the CB or EB.

For example, a token bucket policing policy might be defined as follows:

- CIR = 1 Mbps
- EIR = 512 Kbps
- CB = 500 KB
- EB = 250 KB

In this policy, the token bucket would be refilled at a rate of 1 Mbps, and would have a maximum capacity of 500 KB. If the bucket ever contained more than 500 KB of tokens, excess tokens would be dropped. The CIR would be enforced by limiting the rate at which tokens are refilled, while the EIR would be enforced by allowing the token bucket to accumulate additional tokens beyond the maximum capacity. The CB and EB would limit the amount of data that could be sent during a burst, and the amount of excess data that could be sent during a burst, respectively.

`class CNAME`

The class command is used inside a policy map to create a class for which specific actions will be defined. Once the class has been defined, you can specify the actions to be taken for traffic matching the class criteria using QoS command police.

- `CNAME`: Creates a class within the policy map and specifies the name of the class.

`police CB [CIR [EIR]] conform-action ACTION exceed-action ACTION [violate-action ACTION]`

The police command is used within a policy-map in to define the policing action to be taken for a specific class.

- `CB`: The size of the committed burst in bits.
- `CIR`: Committed Information Rate in bits per second (bps). This is the rate at which the device commits to forwarding traffic.
- `EIR`: The Excess Information Rate in bits per second (bps). This is the rate at which the device allows traffic to exceed the committed rate for a specified amount of time.
- `conform-action ACTION`: Specifies the action to be taken for traffic that conforms to the rate limit. The available actions are:
  - `transmit`: Allow the traffic to be transmitted normally.
  - `set-dscp-transmit n`: Set the packet's DSCP value to *n* and then transmit the packet.
- `exceed-action ACTION`: Specifies the action to be taken for traffic that exceeds the rate limit but falls within the normal burst size. The available actions are the same as for `conform-action`.
- `violate-action ACTION`: Specifies the action to be taken for traffic that exceeds the rate limit and the normal burst size. The available actions are the same as for `conform-action`.

The "police" command can be used for both single-rate and dual-rate policing. In single-rate policing, only the CIR and CB values are used, while in dual-rate policing, both the CIR and EIR values are used.

In the context of the "police" command, the token bucket algorithm is used to enforce the configured CIR, EIR, and CB values. The bucket size is set to CB, and tokens are added to the bucket at a rate of CIR. If the bucket is full, any additional tokens are discarded. When traffic arrives, the bucket is checked to see if it contains enough tokens to accommodate the traffic. If there are enough tokens, the traffic is transmitted, and tokens are subtracted from the bucket. If there are not enough tokens, the traffic is dropped.

In dual-rate policing, two token buckets are used - one for the CIR and another for the EIR. The CIR bucket is filled at a rate of CIR, while the EIR bucket is filled at a rate of EIR. Traffic is transmitted if there are enough tokens in either the CIR or EIR bucket.

Example :

```
soodar(config)# policy-map pmap1
soodar(config-pmap)# class cmap1
soodar(config-pmap-c)# police 1000000 8000 16000 conform-action transmit␣
↪exceed-action set-dscp-transmit 26 violate-action drop
```

This command sets the committed burst size to 8000 bytes, the CIR to 1000000 bits per second, and the EIR to 16000 bits per second. Conforming traffic is transmitted, exceeding traffic has its DSCP value set to CS3 and is transmitted, and violating traffic is dropped.

Traffic shaping

Traffic shaping is a QoS (Quality of Service) mechanism used to control the rate of data transmission on a network. It is used to smooth out bursts of traffic and ensure that the network does not become congested. Traffic shaping works by delaying packets that exceed a certain rate, allowing them to be transmitted at a slower rate.

shape average RATE

The shape command is used within a policy-map to define the traffic shaping action to be taken for a specific class.

- RATE: The average rate at which traffic should be shaped, in bits per second (bps). The rate could be a value between 8000 and 1000000000(8kbps to 1Gbps). It could be entered in human readable format like 1M, 1G etc.

Example :

```
soodar(config)# class-map cmap1
soodar(config-cmap)# match protocol http
soodar(config)# policy-map pmap1
soodar(config-pmap)# class cmap1
soodar(config-pmap-c)# shape average 10M
```

This command shapes the traffic to an average rate of 10 Mbps for *http* protocol.

Priority queuing

Priority queuing is a QoS (Quality of Service) mechanism used to guarantee the bandwidth and prioritize the transmission of certain types of traffic over others. It is used to ensure that critical traffic, such as voice or video, is given priority over less important traffic, such as file transfers or software updates. When the rate of outgoing prioritized traffic exceeds the allocated bandwidth, the excess traffic is dropped.

priority BPS

The priority command is used within a policy-map to define the priority queuing action to be taken for a specific class.

- BPS: The bandwidth in bits per second (bps) that should be allocated to the priority queue. It could be entered in human readable format like 1M, 1G etc.

priority percent PERCENT

The priority percent command is used within a policy-map to define the priority queuing action to be taken for a specific class.

- PERCENT: The percentage of the total bandwidth that should be allocated to the priority queue.

Example :

```
soodar(config)# class-map cmap1
soodar(config-cmap)# match protocol icmp
soodar(config)# policy-map pmap1
soodar(config-pmap)# class cmap1
soodar(config-pmap-c)# priority 1M
```

This command allocates 1 Mbps of bandwidth to the priority queue for *icmp* protocol. When the rate of outgoing prioritized traffic exceeds 1Mbps, the excess traffic is dropped.

Bandwidth allocation

Bandwidth allocation is a QoS (Quality of Service) mechanism used to allocate a specific amount of bandwidth to a particular class of traffic. It is used to ensure that traffic receives the necessary bandwidth during the congestion to function properly.

SoodarOS uses Weighted Fair Queuing (WFQ) algorithm to allocate bandwidth for different classes of traffic. When there are no congestion, the traffic rate is not limited.

bandwidth BPS

The bandwidth command is used within a policy-map to define the bandwidth allocation action to be taken for a specific class.

- BPS: The bandwidth in bits per second (bps) that should be allocated to the class. It could be entered in human readable format like 1M, 1G etc.

bandwidth percent PERCENT

The bandwidth percent command is used within a policy-map to define the bandwidth allocation action to be taken for a specific class.

- PERCENT: The percentage of the total bandwidth that should be allocated to the class.

Example :

```
soodar(config)# class-map cmap1
soodar(config-cmap)# match protocol udp
soodar(config)# policy-map pmap1
soodar(config-pmap)# class cmap1
soodar(config-pmap-c)# bandwidth 1M
```

This command allocates 1 Mbps of bandwidth to the class *udp* protocol during congestion.

Apply to interface

service-policy PMAP <input|output> [track (1-1000)]

This command applies a QoS (Quality of Service) policy map to an incoming or outgoing router interface. The PMAP parameter specifies the name of the policy map to be applied. The input or output keyword specifies the direction of the interface to which the policy is applied.

The command also supports an optional track parameter with a value between 1 and 1000 to associate a track object with the policy map. A track object can be used to track the status of an interface or a specific IP route, and if the tracked object fails, the policy map can be removed.

Note: For ingress traffic, the only acceptable policy is policing. For egress traffic, the acceptable policies are policing, shaping, priority, and bandwidth.

Example :

```
n1(config-if)# service-policy pmap1 in
```

Show commands

show policy-map [NAME]

Example :

```
n1(config)# do sh policy-map pmap1

Policy Map pmap1
   Class cmap
      Police CIR 102400 (bps) CB 25600 (byte) EB 35840 (byte)
      Conform Action : Transmit
      Exceed Action : Drop
```

Logging

Debugging logs can be set in case of need.

debug qos event

log data plane installation processes and results

1.9 SLA

1.9.1 IP SLA

IP SLA (Service-Level Agreement) is a feature used to measure network performance and verify network service levels. It allows network administrators to simulate network traffic and measure the performance of devices. IP SLA can be used to monitor a wide range of network parameters such as packet loss, latency, jitter, and availability. It can also be used to trigger logging events, such as failover, when performance thresholds are exceeded. IP SLA can help network administrators identify and troubleshoot network issues and ensure that network service levels are met.

Defining an IP SLA operation

In this part, we will see how to define an IP SLA operation that measures network performance and verifies network service levels.

Operation types

Currently there are two operation types:

- **icmp-echo**: Sends ICMP echo requests to measure round-trip time between devices.
- **icmp-jitter**: Measures packet loss, and jitter for ICMP traffic.

ICMP Echo

This IP SLA operation sends ICMP echo requests to a specified destination IP address to measure the round-trip time between devices. The ICMP echo requests simulate network traffic and help to identify network performance issues.

ICMP Echo parameters

- **destination**: Specifies the destination IP address for the ICMP echo request.
- **source**: Specifies the source IP address for the ICMP echo request.
- **frequency**: Specifies the interval at which the ICMP echo requests are sent.
- **timeout**: Specifies the time that the IP SLA operation waits for a response to the ICMP echo request.
- **threshold**: Specifies the upper threshold for round-trip time. Used in monitoring and reactions.
- **VRF**: Set the VRF instance for this IP SLA operation
- **payload size**: Specifies ICMP packet payload size.

ICMP Jitter

This IP SLA operation sends a series of ICMP echo requests to a specified destination IP address to measure the network's packet delay variation (jitter). The operation measures the delay between the transmission of each packet and the response received from the destination and calculates the jitter by comparing the delay values.

ICMP Jitter parameters

- **destination**: Specifies the destination IP address for the ICMP echo request.
- **source**: Specifies the source IP address for the ICMP echo request.
- **num**-packets: Specifies the number of packets to be sent for each operation.
- **interval**: Specifies the duration between consecutive echo requests sent to a target device.
- **frequency**: Specifies the interval at which the operation is ran.
- **timeout**: Specifies the time that the IP SLA operation waits for a response to the ICMP echo request.
- **threshold**: Specifies the upper threshold for round-trip time. Used in monitoring and reactions.
- **VRF**: Set the VRF instance for this IP SLA operation
- **percentile**: Specifies the percentage of lower round-trip times that are used in percentile statistics

Commands

This section introduce CLI commands to define an IP SLA.

ip sla (1-2147483647)

Create an IP Service Level Agreement (SLA).

- (1-2147483647): The numeric argument specifies the index number of the IP SLA instance being configured.

ICMP echo

icmp-echo <A.B.C.D|X:X::X:X|HOST> [source-ip <A.B.C.D|X:X::X:X>]

It is used to configure an ICMP Echo operation to be performed by the router. The following argument, which can be either an IPv4 or IPv6 address or a hostname, specifies the destination for the ICMP Echo request.

Optionally, the source-ip parameter can be used to specify the source IP address for the ICMP Echo operation. This is useful in cases where the router has multiple interfaces, and the administrator wants to control which interface is used to generate the ICMP Echo request.

frequency (1-604800)

Set the rate at which the ICMP Echo requests are sent to the destination. The frequency parameter is specified in seconds and can range from 1 to 604800 (one week).

For example, if a frequency of 60 seconds is specified(the default value for frequency), the router will send an ICMP Echo request to the set destination every minute. This parameter can be adjusted to increase or decrease the rate at which the ICMP Echo requests are sent, depending on the network administrator's needs.

Note: default frequency value is 60 seconds.

---

timeout (0-604800000)

Determines the maximum time the IP SLA operation will wait for a response from the destination host. The timeout parameter is specified in milliseconds and can range from 1 to 604800000 (i.e., one week).

When the IP SLA operation sends an ICMP Echo packet to the destination host, it waits for a response from the host within the timeout period. If a response is not received within the specified timeout, the IP SLA operation is considered to have failed. The timeout value can be set to a value that is appropriate for the network being monitored based on the typical response times and performance requirements.

For example, suppose a timeout of 5000 milliseconds (5 seconds, the default value for timeout) is specified. In that case, the IP SLA operation will wait up to 5 seconds for a response from the destination host. If a response is not received within this time, the IP SLA operation is considered to have failed.

> **Warning:** timeout value can not be greater than the frequency.

Note: default time value is 5000 milliseconds.

---

threshold (1-60000)

Set the upper threshold value for calculating network monitoring statistics. The threshold parameter is specified in milliseconds and can range from 1 to 60000.

When the IP SLA operation sends an ICMP Echo packet to the destination host, The IP SLA operation is considered Overthreshold if a response is not received within the specified threshold. The threshold value can be set to a value appropriate for the monitored network based on the typical response times and performance requirements.

For example, suppose a threshold of `5000` milliseconds (5 seconds, the default value for threshold) is specified. In that case, If a response is not received within this time, the IP SLA operation is considered Overthreshold.

Warning: threshold value can not be greater than timeout.

Note: Overthreshold operations are not failed and are reachable and Ok. This is not intended to be used as an event triggering event, but to be used with *IP SLA Reactions*.

Note: default threshold value is 5000 milliseconds.

`vrf VRF`

This optional parameter specifies the VRF instance in which the operation should be performed.

For example, consider a scenario where a service provider provides Internet connectivity to multiple customers, each of whom has its own VRF. In this scenario, the service provider can use the VRF parameter in the IP SLA operation to perform the operation in the correct VRF. This allows the service provider to monitor each customer's network's performance independently.

If the VRF parameter is not specified in the IP SLA operation, the operation will be performed in the default VRF of the router

`request-data-size (0-16384)`

Specify the number of bytes in the payload of the ICMP echo request message. The value can be set to any number between 0 and 16384 bytes.

The purpose of this parameter is to simulate different network traffic conditions and evaluate the network's performance. The IP SLA operation can determine if the network can handle large data payloads without significant loss or delay by specifying a large value.

Note: default request-data-size value is 28 bytes.

Example:

To configure an IP SLA operation that sends ICMP echo requests to a destination IP address of 192.0.2.1, use the following command:

```
soodar(config)# ip sla 1
soodar(config-ip-sla)# icmp-echo 192.0.2.1
soodar(config-ip-sla-echo)# frequency 10
soodar(config-ip-sla-echo)# timeout 1000
soodar(config-ip-sla-echo)# threshold 500
```

This configuration will cause the router to send ICMP echo requests to the IP address 192.0.2.1 every 10 seconds and measure the round-trip time for each request. If the round-trip time exceeds 1000 milliseconds, an event will be triggered to indicate that the operation has failed and the operation state changes to Timed-out. If the round-trip time exceeds 500 milliseconds, the operation state is Overthreshold. Otherwise, the operation state is Ok.

ICMP jitter

`icmp-jitter <A.B.C.D|X:X::X:X|HOST> [{source-ip <A.B.C.D|X:X::X:X> | interval (4-60000) | num-packets (1-60000)}`

It is used to configure an ICMP Jitter operation to be performed by the router. The following argument, which can be either an IPv4 or IPv6 address or a hostname, specifies the destination for the ICMP Echo request.

Optionally, the `source-ip` parameter can be used to specify the source IP address for the ICMP Jitter operation. This is useful in cases where the router has multiple interfaces, and the administrator wants to control which interface is used to generate the ICMP Echo request.

The `interval` option specifies the interval (in milliseconds) between packets sent by the IP SLA operation. The default value is `20 milliseconds`.

The `num-packets` option specifies the number of packets to be sent by the IP SLA operation. The default value is `10 packets`.

Note: Default value for interval is 20 milliseconds.

Note: Default value for num-packets is 10 packets.

`frequency (1-604800)`

Set the interval in seconds between the initiation of consecutive operations. The `frequency` value is specified in seconds, and the valid range is from 1 to 604800 seconds (one week).

For example, if a frequency of `60` seconds is specified(the default value for frequency), the router will initiate an operation every minute. This parameter can be adjusted to increase or decrease the rate at which the ICMP Echo requests are sent, depending on the network administrator's needs.

Note: default frequency value is 60 seconds.

`timeout (0-604800000)`

Determines how long the system will wait for a response from the target device before considering the IP ICMP Echo request operation failed. The `timeout` parameter is specified in milliseconds and can range from 1 to 604800000 (i.e., one week).

For example, suppose a timeout of `5000` milliseconds (5 seconds, the default value for timeout) is specified. In that case, the IP SLA operation will wait up to 5 seconds for a response from the destination host for each ICMP Echo request. If a response is not received within this time, the IP ICMP Echo request is considered to have failed.

Warning: (timeout + number of packets * interval) value can not be greater than the frequency.

Note: default time value is 5000 milliseconds.

Note: When all ICMP Echo requests are failed, the whole operation status is Timedout. Otherwise, the operation status is Ok.

threshold (1-60000)

Set the upper threshold value for calculating network monitoring statistics. The `threshold` parameter is specified in milliseconds and can range from 1 to 60000.

When the IP SLA operation is done and the average jitter is computed, The IP SLA operation is considered Overthreshold if the average jitter value is above the threshold. The threshold value can be set to a value appropriate for the monitored network based on the typical response times and performance requirements.

> **Warning:** threshold value can not be greater than the timeout.

Note: Overthreshold operations are not failed and are reachable and Ok. This is not intended to be used as a trigger for an event but to be used with *IP SLA Reactions*.

Note: default threshold value is 5000 milliseconds.

vrf VRF

This optional parameter specifies the VRF instance in which the operation should be performed.

For example, consider a scenario where a service provider provides Internet connectivity to multiple customers, each with its own VRF. In this scenario, the service provider can use the VRF parameter in the IP SLA operation to operate in the correct VRF. This allows the service provider to monitor each customer's network's performance independently.

If the VRF parameter is not specified in the IP SLA operation, the operation will be performed in the default VRF of the router

percentile <jitteravg|rtt> (90-100)

Specifies the percentage of packets that must have a jitter or RTT measurement less than or equal to the maximum measured jitter or RTT. The percentile parameter can take two values `jitteravg` and `rtt`.

The percentile parameter takes a value between 90 and 100. For example, if percentile jitteravg 95 is specified, top 5% of the highest jitter measurements are not considered in statistics and computations.

Example:

One possible real-world scenario for using the IP SLA ICMP Jitter operation with the percentile feature is to monitor the quality of a VoIP (Voice over IP) service. VoIP is a real-time application sensitive to delays, jitter, and packet loss, which can affect call quality.

We can configure the router to generate ICMP traffic to the VoIP server to monitor the VoIP quality using the IP SLA ICMP Jitter operation. The router will send a specified number of ICMP packets at a specific interval to the server and measure the delay, jitter, and packet loss statistics.

We can use the percentile feature to calculate the 95th percentile of the round-trip time (RTT) of the ICMP packets.

```
soodar(config)# ip sla 1
soodar(config-ip-sla)# icmp-jitter 10.0.0.1 source-ip 10.0.0.2 num-packets 100␣
↪interval 20
soodar(config-ip-sla-icmpjitter)# frequency 20
soodar(config-ip-sla-icmpjitter)# timeout 5000
soodar(config-ip-sla-icmpjitter)# threshold 100
soodar(config-ip-sla-icmpjitter)# percentile rtt 95
```

In this example, we configure the router to generate 100 ICMP packets every 20 seconds with intervals of 20 ms from the IP address 10.0.0.2 to the VoIP server at IP address 10.0.0.1.

Scheduling an IP SLA operation

After defining an IP SLA operation, the next step to use it is to schedule it.

IP SLA scheduling allows you to configure and schedule IP SLA operations to run automatically and periodically without manual intervention.

Commands

```
ip sla schedule (1-2147483647)$sla [{life <forever|(0-2147483647)> \
|start-time <now|HH:MM|after HH:MM|pending>|recurring}]
```

It is used to schedule an IP SLA operation. The command configures the operation's starting time, duration, and whether it should recur daily. The input parameters for this command are:

- `(1-2147483647)` : the ID of the IP SLA operation to be scheduled.
- `life`: optional parameter used to specify the duration of the operation. It can be set to either **forever** or a time value in seconds, after which the operation will stop.
- `start-time`: specifies the time at which the operation should start. It can be set to **now** to start the operation immediately after configuration, a specific time in **HH:MM** format, a time in **after HH:MM** format to start after a specific time, or **pending** to wait for an external trigger.
- `recurring`: specifies whether the IP SLA operation should recur. If set to, the operation will run again after 24 hours.

Note: Default start time is **pending** .

Note: Default lifetime is 3600(1 hour).

IP SLA Reactions

An IP SLA reaction is a set of actions a device can take when a particular IP SLA operation meets certain criteria or thresholds. IP SLA reactions can trigger two responses: generating a log message to be used by the monitoring system or starting another IP SLA operation. IP SLA reactions aim to help network administrators promptly identify and respond to network performance issues. By monitoring key metrics and taking automated actions, IP SLA reactions can help ensure critical network services remain available and responsive.

IP SLA Reactions type

A reaction can consider multiple aspects of an operation as its basis for criteria. These are called **reacts** or reaction-type.

The user can define an action based on the following measurements of IP SLA operation:

- **Average jitter**: A specific react type that considers the average jitter value of an **ICMP-jitter** operation.
- **Percentile average jitter**: A specific react type that considers the percentile average jitter value of an **ICMP-jitter** operation.
- **Round-trip time**: A react type considering the RTT value of an **ICMP-echo** operation or the average RTT value of an **ICMP-jitter** operation.
- **Overthreshold**: This react type takes into account the percentage of Overthreshold packets in an **ICMP-jitter** operation.
- **Packet loss**: A specific react type considering the packet loss count in an **ICMP-jitter** operation.
- **Timeout**: This react type considers the state of an IP SLA operation.

IP SLA Reactions action

The action specifies the action that should be taken when an IP SLA operation has reached a certain threshold. The available actions are:

- **None**: Do nothing.
- **Only log**: Log in syslog.
- **Only trigger**: Trigger another SLA operation to run.
- **Log and trigger**: Do both logging and triggering.

Note: Currently, SoodarOS does not have any Event Management, so user-defined actions are not feasible.

IP SLA Reactions threshold

A reaction threshold is a user-defined value that defines a threshold to determine whether the network performance metric is considered satisfactory. If the metric value exceeds the threshold, an action can be triggered as defined in the IP SLA reaction.

The reaction threshold is a numerical value that varies depending on the IP SLA operation. For example, the reaction threshold for the ICMP echo operation might be the maximum round-trip time acceptable for the network. If the IP SLA operation measures a round-trip time greater than this value, the threshold is exceeded, and the action associated with this threshold will be triggered.

Various types of thresholds can be used in IP SLA, including average, immediate, consecutive, and x of y. The specific kind of threshold used depends on the IP SLA operation and the desired behavior.

- **Immediate**: triggers an event immediately when the value for a reaction type (such as RTT) exceeds the upper threshold value or falls below the lower threshold value or when a timeout occurs.
- **Consecutive**: generates an event after a violation takes place a number (*n*) of times consecutively. For example, this type would configure an action after a timeout repeats *3* times or when the RTT falls below the upper threshold value *n* times.

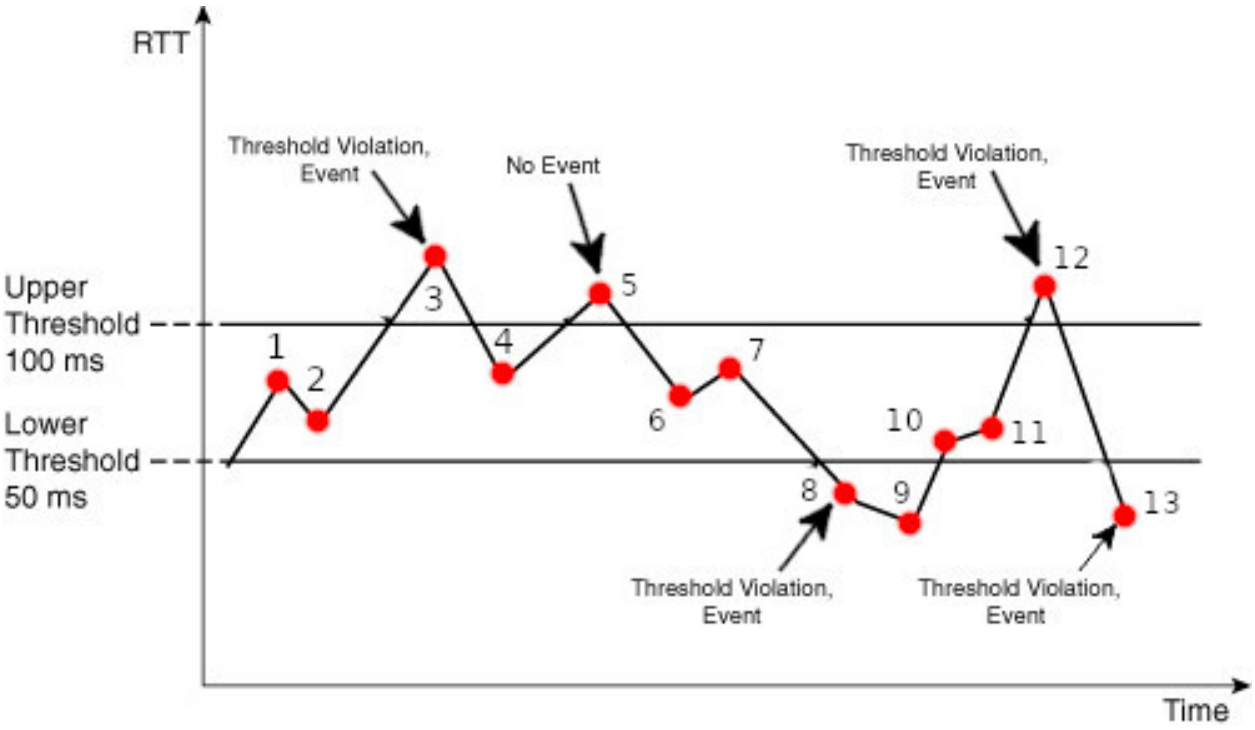
- **X of Y**: triggers an event after a number (*x*) of violations within another number (*y*) of operations. **Example:** generate an event if the average jitter exceeds 30 ms for 10 (x) times during 100 (y) ICMP Jitter operations.
- **Average**: triggers an event when the averaged totals of a value for a number (*n*) of operations exceeds the specified upper threshold value or falls below the lower threshold value.

There are two threshold values: **Upper threshold** and **lower threshold**. The upper threshold is the maximum value the monitored metric (e.g. RTT, jitter, packet loss) can reach before a violation occurs. The lower threshold is the minimum value the watched metric (e.g. RTT, jitter, packet loss) can reach before a violation occurs. Depending on the threshold type and previous states, this violation could create an event to start an action and changes the reaction state to *Raised* or *Fallen*.

When a reaction state is *raised*, it will stay in that state until the lower threshold criteria are met and vice versa.

For example, the following plot is the RTT values throughout an IP SLA ICMP Echo operation running; with an IP SLA Reaction its threshold type is Immediate, and an upper threshold value of 100 and a lower threshold value of 50:

Note: Reactions default state is **fallen**



The above figure shows 13 RTT values, each corresponding with an IP SLA Operation. The 1st and 2nd RTT values lie between the upper and lower threshold(the neutral zone), so no violation occurs and no events(no change in the state). In the 3rd run, the RTT value passes the upper threshold; a violation occurs, and since we have an immediate threshold type, events are generated, and the state changes to *raised*. The 4th run RTT value is in the neutral zone, and the reaction state does not change. The next run(5th) again passes the upper threshold. But since the state is already *raised*, no violation occurs. The next two runs are between the upper and lower threshold. The 8th run is the first run that passes the lower threshold and causes a violation. The immediate threshold type makes an event on the first violation and changes the reaction state to *fallen*. The subsequent run is also below the lower threshold, but nothing happens because the state is already *fallen*. Runs number 9 and 10 are in the neutral zone. The 12th run passes the upper threshold, and the reaction state flips to *raised*. The following run lies below the lower threshold. Therefore, the state again turns to *fallen*, triggering the appropriate events.

IP SLA Reactions trigger

One can define a secondary operation to enable the transition from a pending state to an active state when an IP SLA Reaction event is activated. So when the first IP SLA Reaction event is created, the second IP SLA Operation starts.

Note: When a reaction triggers another SLA, that SLA begins immediately regardless of its schedule. If there is any schedule for that SLA, the lifetime is respected. Otherwise, the default lifetime(3600) is used.

Warning: Triggered operations keep running even after the criteria come under the threshold. If needed to stop, the user should manually unschedule it.

IP SLA Reactions CLI

```
ip sla reaction-configuration (1-2147483647) react \
<jitteravg | jitterAvgPct |rtt |overThreshold | packetLoss | timeout> \
[action-type <none | logOnly | logAndTrigger | triggerOnly> | <average (1-16)| \
immediate | consecutive (1-16) | never | xOfy (1-16) (1-16)>| \
threshold-value (1-60000) (1-60000)]
```

The command is used to configure the reaction behavior of IP SLA operations. These configurations allow you to specify the behavior that should occur when certain SLA thresholds are exceeded.

The parameters for this command are:

- `(1-2147483647)`: The number of the SLA operation for which you want to configure the reaction.
- `react`: The type of SLA metric that will trigger the reaction. This type can be one of the following:
  - `jitterAvg`: The average jitter value.
  - `jitterAvgPct`: The average jitter value computed using percentile.
  - `rtt`: The round-trip time(or average round-trip time in case of an IP SLA ICMP-Jitter).
  - `overThreshold`: The percentage of packets with RTT over a certain threshold.
  - `packetLoss`: The number of lost packets.
  - `timeout`: The status of the operation.
- `action-type`: Specifies the type of action to be taken when the threshold is exceeded. This can be one of the following:
  - `none`: No action will be taken.
  - `logOnly`: An entry will be logged in the device's log.
  - `logAndTrigger`: An entry will be logged in the device's log, and another SLA Operation will start.
  - `triggerOnly`: Another SLA operation will start.
- `threshold-type`: Specifies the threshold type for the action to be taken. This can be one of the following:
  - `average (1-16)`: Specifies the number of operations to average the metric.
  - `immediate`: The action will be taken once a threshold is exceeded.
  - `consecutive (1-16)`: Specifies the number of successive operations that must exceed a threshold before action.

  - `never`: The action will never be taken.
  - `xOfy (1-16) (1-16)`: Specifies that the action should be taken when at least *x* out of *y* packets exceed the threshold. The first argument is the x value, and the second is the y value.
- `threshold-value`: Specifies the upper and lower thresholds for the metric. The first value is the upper threshold, and the next is the lower threshold.

Note: The default action-type is **none**.

Note: The rtt reaction type default threshold values are *5000* for upper and *3000* for lower threshold.

Note: The jitter average reaction type default threshold values are *100* for upper and *100* for lower threshold.

Note: The jitter average reaction percentile type default threshold values are *100* for upper and *100* for lower threshold.

Note: The packet loss reaction type default values are *5* and *5*.

Note: The overthreshold reaction type default values are *40* and *40*.

Note: Threshold values use the same unit as the metric. For example, the RTT reaction type thresholds are in milliseconds.

Note: *timeout* reaction type has no threshold values.

Note: An IP SLA ICMP Echo operation only supports **RTT** and **Timeout** reactions

---

Note: In IP SLA ICMP Jitter operation, the **RTT** reaction equals the average RTT reaction.

`ip sla reaction-trigger (1-2147483647) (1-2147483647)`

This command configures the IP SLA operation to be triggered by configured IP SLA Reactions of an IP SLA.

The first argument is the IP SLA index of the operation that its reactions may trigger another SLA.

The second input is the index of the IP SLA that will be triggered.

Note: Each IP SLA could trigger multiple IP SLAs.

IP SLA Reaction example

This example configures two IP SLA operations to monitor the jitter to a host and the gateway. The first IP SLA operation is configured to monitor the gateway at noon for 1 hour daily. This operation triggers a reaction if the jitter average exceeds 50ms in a single measurement interval. The reaction is configured to be a logOnly action, meaning it will only log the violation.

The second IP SLA operation is configured to trigger a reaction if the jitter average exceeds 20ms in three consecutive measurement intervals. The reaction is configured to be a logAndTrigger action, meaning it will log the violation and also schedule another IP SLA operation to measure the quality of the connection with the gateway. The ip sla reaction-trigger command is used to associate the second IP SLA operation(the host) with the first IP SLA operation(The gateway), so that the trigger from the second operation will activate the first operation.

```
soodar(config)# ip sla 1
soodar(config-ip-sla)# icmp-jitter 192.168.1.1
soodar(config-ip-sla-icmpjitter)# frequency 60
soodar(config)# ip sla schedule 1 start-time 12:00 life 3600 recurring
soodar(config)# ip sla reaction-configuration 1 react jitteravg threshold-type immediate␣
↪threshold-value 50 50 action-type logOnly

soodar(config)# ip sla 2
soodar(config-ip-sla)# icmp-jitter 200.1.2.2
soodar(config-ip-sla-icmpjitter)# frequency 60
soodar(config)# ip sla schedule 2 start-time now life forever
soodar(config)# ip sla reaction-configuration 2 react jitteravg threshold-type␣
↪consecutive 3 threshold-value 20 20 action-type logAndTrigger
soodar(config)# ip sla reaction-trigger 2 1
```

IP SLA Troubleshooting

IP SLA (Service Level Agreement) troubleshooting involves using various commands and tools to diagnose and resolve issues related to IP SLA operations. Some common issues that can be encountered include incorrect configuration, failed operations, and inaccurate measurements.

To troubleshoot IP SLA, one can use the commands to view the status and results of IP SLA operations, and to verify the configuration of IP SLA operations. There are also commands to enable debugging messages for IP SLA events.

In addition to these commands, other troubleshooting tools, such as packet capture and network analysis tools, can be used to identify and resolve IP SLA issues. It's also essential to ensure that the underlying network infrastructure is configured properly and functioning correctly to avoid any problems with IP SLA.

Debugging events

debug sla event

The debug ip sla event command is used to display debug messages related to IP Service Level Agreements (SLAs) events

When the command is enabled, the router generates debug messages showing the IP SLAs events. These messages can be helpful for troubleshooting network issues related to IP SLAs.

debug socket event

This command enables the debugging of socket events on the router. This command is used to troubleshoot issues related to socket operations, such as socket connections and disconnections, socket errors, and the read data.

The socket is used for connection between the SLA service program and the router program.

Displaying IP SLA

show ip sla statistics (1-2147483647) [<details|json>]

This command is used to display the statistics for a specific IP SLA operation. It can be used to view real-time performance metrics such as round-trip time (RTT), packet loss, jitter, and more.

The parameters are defined as follows:

- `(1-2147483647)`: Specifies the number of the IP SLA operation for which statistics are to be displayed.
- `details`: (Optional) Displays additional details about the operation.
- `json`: (Optional) Formats the output as JSON for easy parsing and automation.

Example:

```
soodar# show ip sla statistics 12
IPSLA Operation id: 12
Type of operation: icmp-echo
  Latest RTT: 13 milliseconds
Latest operation start time: Wed Feb 22 09:56:30 2023
Latest successfull operation time: Wed Feb 22 09:56:30 2023
Latest failed operation time: N/A
Latest operation return code: OK
Number of successes: 6
Number of failures: 0
Operation time to live: 00:59:34

soodar# show ip sla statistics 13
IPSLA Operation id: 13
Type of operation: icmp-jitter
  Latest RTT: 4 milliseconds
Latest operation start time: Wed Feb 22 10:23:10 2023
Latest successfull operation time: Wed Feb 22 10:23:11 2023
Latest failed operation time: N/A
Latest failed operation error: Timed out
Latest operation return code: OK
RTT Values:
  Number of RTT:10      RTT Min/Avg/Max: 4/10/13 milliseconds
Jitter time:
  Number of Jitter Samples: 9
  Jitter Min/Avg/Max: 3/3/4 milliseconds
Percentile Jitter time:
  Number of Percentile Jitter Samples (95%): 8
  Percentile Jitter Min/Avg/Max: 3/3/4 milliseconds
Over Threshold:
  Number Of RTT Over Threshold: 0
Out of Sequence: 0
Packet Loss: 0
Number of successes: 41
Number of failures: 0
Operation time to live: 00:56:40
```

(continues on next page)

(continued from previous page)

```
soodar# show ip sla statistics 13 details
IPSLA Operation id: 13
Type of operation: icmp-jitter
  Latest RTT: 8 milliseconds
Latest operation start time: Wed Feb 22 10:23:40 2023
Latest successfull operation time: Wed Feb 22 10:23:41 2023
Latest failed operation time: N/A
Latest failed operation error: Timed out
Latest operation return code: OK
RTT Values:
  Number of RTT:10      RTT Min/Avg/Max: 8/10/13 milliseconds
Jitter time:
  Number of Jitter Samples: 9
  Jitter Min/Avg/Max: 3/3/4 milliseconds
  Positive Jitter Num/Min/Avg/Max: 4/3/3/4 milliseconds
  Negative Jitter Num/Min/Avg/Max: 5/3/3/4 milliseconds
Percentile Jitter time:
  Number of Percentile Jitter Samples (95%): 8
  Percentile Jitter Min/Avg/Max: 3/3/4 milliseconds
Over Threshold:
  Number Of RTT Over Threshold: 0
Out of Sequence: 0
Packet Loss: 0
Number of successes: 47
Number of failures: 0
Operation time to live: 00:56:09
```

show ip sla configuration [(1-2147483647)] [json]

The command is used to display the configuration details of an IP Service Level Agreements (SLA) operation. This command can be used with or without specifying the specific operation ID.

The parameters are defined as follows:

- `(1-2147483647)`: (Optional) Specifies the operation ID of the IP SLA operation for which the configuration details are to be displayed. The command will display the configuration for all IP SLA operations if not specified.
- `json`: (Optional) Formats the output as JSON for easy parsing and automation.

Example:

```
soodar# show ip sla configuration 13
Entry number: 13
 Operation timeout (milliseconds): 5000
 Type of operation to perform: icmp-jitter
 Target address/Source address: 200.1.2.2/0.0.0.0
 Packet Interval (milliseconds)/Number of packets: 20/10
 VRF name:
 Schedule:
  Operation frequency (seconds): 5
  Next Scheduled Start Time: Start Time already passed
  Life (seconds): 3600
```

(continues on next page)

(continued from previous page)

```
  Recurring (Starting Everyday): FALSE
 Threshold (milliseconds):: 5000
 Percentile:
  JitterAvg: 95%


soodar# show ip sla configuration
Entry number: 12
 Operation timeout (milliseconds): 5000
 Type of operation to perform: icmp-echo
 Target address/Source address: 200.1.2.2/0.0.0.0
 Request size (ARR data portion): 28
 VRF name:
 Schedule:
  Operation frequency (seconds): 5
  Next Scheduled Start Time: Start Time already passed
  Life (seconds): 3600
  Recurring (Starting Everyday): FALSE
 Threshold (milliseconds):: 5000
Entry number: 13
 Operation timeout (milliseconds): 5000
 Type of operation to perform: icmp-jitter
 Target address/Source address: 200.1.2.2/0.0.0.0
 Packet Interval (milliseconds)/Number of packets: 20/10
 VRF name:
 Schedule:
  Operation frequency (seconds): 5
  Next Scheduled Start Time: Start Time already passed
  Life (seconds): 3600
  Recurring (Starting Everyday): FALSE
 Threshold (milliseconds):: 5000
 Percentile:
  JitterAvg: 95%
```

show ip sla reaction-configuration [(1-2147483647)] [json]

The command displays a device's current IP SLA reaction configuration. It provides information on how IP SLA operations react to certain events or conditions, such as when a threshold is exceeded.

The parameters are defined as follows:

- `(1-2147483647)`: (Optional) Specifies the operation ID of the IP SLA operation for which the reaction configuration details are to be displayed. If not specified, the command will display the reaction configuration for all IP SLA operations.
- `json`: (Optional) Formats the output as JSON for easy parsing and automation.

Example:

```
soodar# show ip sla reaction-configuration 12
Entry number: 12
  Reaction: rtt
    Threshold type: Immediate
    Rising threshold: 500 milliseconds
    Falling threshold: 100 milliseconds
```

(continues on next page)

(continued from previous page)

```
    Action type: Log only

soodar# show ip sla reaction-configuration
Entry number: 12
  Reaction: rtt
    Threshold type: Immediate
    Rising threshold: 500 milliseconds
    Falling threshold: 100 milliseconds
    Action type: Log only
Entry number: 13
  Reaction: jitterAvgPct
    Threshold type: X of Y
    Threshold X value: 3
    Threshold Y value: 5
    Rising threshold: 120
    Falling threshold: 80
    Action type: Log and trigger
```

show ip sla reaction-trigger [(1-2147483647)] [json]

The command is used to display the current status and configuration of IP SLA reaction triggers.

The parameters are defined as follows:

- `(1-2147483647)`: (Optional) Specifies the operation ID of the IP SLA operation for which the reaction trigger details are to be displayed. If not specified, the command will display the reaction trigger for all IP SLA operations.
- `json`: (Optional) Formats the output as JSON for easy parsing and automation.

Example:

```
soodar# show ip sla reaction-trigger 12
Entry number: 12
  Target entry number: 13
  Operational state: pending
```

1.9.2 Track

A "track" is a feature used for tracking the status of an object, such as an interface or IP address, and taking action based on its state. The track feature is used with other features such as IP SLA and Routing protocols.

The purpose of the track feature is to provide a mechanism for the router to monitor the status of a tracked object and to change the router's behavior based on the state of that object. For example, a track can be configured to monitor the status of a specific interface. If the interface goes down, the track can be set up to trigger an action, such as changing the state of a route.

The track feature can also be used to monitor the state of an IP address. This is useful for failover scenarios where a backup device can take over the IP address of a primary device in case of failure. The track feature can be configured to monitor the primary device's IP address, and if it becomes unreachable, the backup device can take over the IP address.

Defining a track

To define a track, an administrator can use the `track` command followed by a track number (between 1 and 1000), along with a specific condition to monitor, such as the reachability of an IP address or the status of an interface. The track can then be associated with a number of different actions.

Tracks are particularly useful in high availability environments, where administrators need to quickly detect and respond to changes in network conditions to maintain service uptime.

`track (1-1000) interface IFNAME line-protocol`

This command configures the router to track the line protocol status of a specified interface as an object with a specific number (between 1 and 1000).

Here's what each option in the command does:

- `(1-1000)`: Specifies the number assigned to the tracked object. This number must be unique and between 1 and 1000.
- `IFNAME`: Specifies the name of the interface to be tracked.

Example:

```
soodar(config)# track 1 interface ge2 line-protocol
```

`track (1-1000) ip route A.B.C.D/M reachability [A.B.C.D|IFNAME] [vrf VRF]`

Creates a tracking object that monitors the reachability of a specified IP address or network prefix. If the address or prefix becomes unreachable, the tracking object will transition to a "down" state.

Here's what each option in the command does:

- `(1-1000)`: Specifies the number assigned to the tracked object. This number must be unique and between 1 and 1000.
- `A.B.C.D/M`: This option is used to specify the network prefix to be tracked.
- `A.B.C.D | IFNAME`: (Optional) Specifies the next-hop IP address or the outgoing interface that the device uses to reach the specified IP address or network.
- `VRF`: (Optional) Specifies the VRF instance in which the tracked route is located. This option is only required if the device has multiple VRFs configured.

Example:

```
soodar(config)# track 10 ip route 10.1.1.1/32 reachability 10.1.1.1 vrf my_vrf
```

In this example, we're creating a track object with ID 10 that will track the reachability of the IP address 10.1.1.1/32 via the interface associated with the IP address 10.1.1.1, in the VRF called `my_vrf`.

If the route to 10.1.1.1/32 becomes unreachable, the state of the track object will change to *down*. The state will change to *up* if the route becomes reachable again.

Note: Currently, the user can't specify both the next-hop IP address and outgoing interface.

`track (1-1000) ip sla (1-2147483647) <reachability|reaction \`
`<jitterAvg | jitterAvgPct | rtt | overThreshold | packetLoss | timeout> >`

The command is used to track the status of an IP SLA operation. The options within the command specify the type of tracking that should be performed.

Here is a breakdown of the command syntax:

- `(1-1000)`: Specifies the number assigned to the tracked object. This number must be unique and between 1 and 1000.
- `(1-2147483647)`: Specifies the IP SLA operation to track. The range is from 1 to 2147483647.
- `reachability|reaction`: Specifies the type of tracking to perform. If *reachability* is specified, the router tracks whether the IP SLA operation can reach its target. If *reaction* is specified, the router tracks the specified IP SLA reaction type.
- `jitterAvg`: Tracks the average jitter reaction state(in milliseconds) of the IP SLA operation.
- `jitterAvgPct`: Tracks the average percentile jitter reaction state of the IP SLA operation.
- `rtt`: Tracks the round-trip time reaction state(in milliseconds) of the IP SLA operation.
- `overThreshold`: Tracks overthreshold packets reaction state of the IP SLA operation.
- `packetLoss`: Tracks the percentage of packet loss reaction state of the IP SLA operation.
- `timeout`: Tracks whether the IP SLA operation timeout reaction state is *raised* or *fallen*.

Example:

Suppose you have a critical server in your network and you want to monitor its availability. You can use the ip sla feature to send a ping to the server at regular intervals and track its reachability using the track command.

First, you can configure an IP SLA operation to send a ping to the server using the following command:

```
soodar(config)# ip sla 1
soodar(config-ip-sla)# icmp-echo 10.1.1.10
soodar(config)# ip sla schedule 1 start-time now
```

This command will send an ICMP echo request to IP address 10.1.1.10 every 60 seconds for an hour.

Next, you can define a track object to monitor the reachability of the server using the following command:

```
soodar(config)# track 1 ip sla 1 reachability
```

`track (1-1000) list boolean <and|or>`

The command configures a list of tracking objects and specifies the boolean operator (AND or OR) to evaluate their states.

Here is a breakdown of the command options:

- `(1-1000)`: Specifies the number assigned to the tracked object. This number must be unique and between 1 and 1000.
- `<and|or>`: specifies the boolean operator to be used for the list evaluation

Example:

Suppose we have two tracked objects: object 1 and object 2. Object 1 tracks the reachability of a specific IP address, while object 2 tracks the state of a particular interface. We want to create a third tracked object that will be up only if object 1 and object 2 are up. To achieve this, we can use the following command:

```
soodar(config)# track 3 list boolean and
soodar(config-track)# object 1
soodar(config-track)# object 2
```

In this example, track 3 creates a new tracked object with an ID of 3, and *list boolean and* specifies that this object will be up only if both object 1 and object 2 are up. Finally, we add object 1 and object 2 to the list of tracked objects for track 3.

Using a track

Tracks can be used to dynamically install/uninstall static routes and quality of service (QoS). By using tracks, network administrators can build more resilient networks that can automatically adapt to changes in the network environment.

Routes with track

The ip route track command is used to create a static route that depends on the state of a tracked object.

See also:

Static Route Commands

Example:

In this example, the router is configured to track the reachability of IP SLA operation 1. The default gateway for the router is set to 192.168.1.1 and the route is tracked with track 1. If the IP SLA operation fails, the track will go down and the route will be removed from the routing table.

```
soodar(config)# ip sla 1
soodar(config-ip-sla)# icmp-echo 192.168.1.1
soodar(config)# ip sla schedule 1 start-time now
soodar(config)# track 1 ip sla 1 reachability
soodar(config)# ip route 0.0.0.0/0 192.168.1.1 track 1
```

Service policy with track

See also:

Apply to interface

Example:

Let's say we have a router connecting Network A(192.168.1.0/24) and Network B(192.168.2.0/24).

We want to apply a QoS policy to traffic that is leaving the router's interface towards Network B, but only if the interface towards Network A is up and reachable.

First, we would create a track object to track the state of the interface towards Network A:

```
soodar(config)# track 1 interface ge1 line-protocol
```

This track object will track the state of the ge1 interface, and will become "up" if the interface's line protocol is up.

Next, we would create a QoS policy map that defines the actions to take on the traffic:

```
soodar(config)# class-map c1
soodar(config-cmap)# match any
soodar(config)# policy-map p1
soodar(config-pmap)# class c1
soodar(config-pmap-c)# police 100k
```

In this example, we're limiting the available bandwidth to 100Kbps for the traffic that matches the "c1" class.

Finally, we would apply the policy map to the outgoing interface towards Network B, and specify the track object that we created earlier

```
soodar(config)# interface ge2
soodar(config)# service-policy output p1 track 1
```

Troubleshooting

To troubleshoot tracking, it is important to first ensure that the objects being tracked are functioning properly. This can involve checking the status of interfaces, verifying that IP SLA probes are running and reporting accurate results, and confirming that the routing table is correct and up-to-date.

If the objects being tracked are functioning properly, the next step is to check the tracking configuration itself. This involves verifying that the correct objects are being tracked and that the tracking configuration is properly configured with the correct parameters.

Debugging commands, such as "debug track event" and "debug ip sla event," can be used to view real-time information about the tracking status and to help identify any issues with the tracking configuration or the objects being tracked.

show track [(1-1000)] [json]

The show track command is used to display the current state of the configured track objects. This command displays information about the objects and their corresponding status.

The parameters of this command are as follows:

- `(1-1000)`: (Optional) Specifies the track object ID to display information for. The ID can be any value between 1 and 1000.
- `json`: (Optional) Displays the output in JSON format.

Example:

```
soodar# show track
Track 12
  IP SLA 12 reaction
  Reaction is Up
    1 change[s], last change 00:00:09
  Latest operation return code: OK
  Tracked by:
    Static IP Routing
    Track List 13
Track 13
  List boolean or
  List boolean OR is Down
    2 change[s], last change 00:00:08
    object 12 Up
  Tracked by:
    Static IP Routing
```

show ip route track-table

The command is used to display the route tracking information for IP routes. It displays the current status of IP route tracking in the router and shows which routes and tracking objects are being used. This command verifies that the route tracking configuration is working as intended.

Example:

```
soodar# show ip route track-table
ip route 2.0.0.0/8 200.1.2.2 100 track 12  state is [up]
```

debug track event

The command is used to display debugging messages related to track objects.

1.10 Access Control List

1.10.1 IP Access List

An IP access-list, also known as an access control list (ACL), is a list of rules that filters network traffic by specifying which types of traffic are allowed or denied. It is used to control the flow of traffic through a network device, such as a router or firewall, by permitting or denying traffic based on criteria such as source and destination IP addresses, protocol, port number, and other packet attributes.

IP access-lists can be configured in two formats: standard and extended. Standard access-lists only match based on the source IP address, whereas extended access-lists can match based on a variety of packet attributes. ACLs can be applied to inbound or outbound traffic on an interface and can be used for a variety of purposes such as network security, traffic shaping, and routing.

When configuring an IP access-list, it is important to carefully consider the traffic that needs to be allowed or denied and to test the ACL thoroughly to ensure that it is working as intended.

Soodar router is shipped with a rich *ip access list* set of tools. It supports *standard ACLs* and *extended ACLs* in a named manner. An access list uses a *first match* approach. That means the first entry that matches is selected as the result, and the whole process of evaluation is terminated.

Define an ACL

To define or modify an *ipv4* access list, issue the following command:

ip access-list ACL4

The `ACL4` argument in this command is the name of the access list being created or modified.

Example :

```
soodar(config)# ip access-list ACL_TEST
soodar(config-nacl)#
```

For *ipv6* access list, the command uses `ipv6` name.

ipv6 access-list ACL6

Example :

```
soodar(config)# ipv6 access-list ACL6_TEST
soodar(config-ipv6-acl)#
```

Delete an ACL

Use no command to remove an existing IP access-list

no ip access-list ACL4

no ipv6 access-list ACL6

Remark

remark LINE ..

The command is used within an IP access-list to insert a comment that describes the purpose of a particular access-list rule or section. It is essentially a way to document the access-list for future reference and is particularly useful when managing complex access-lists.

Define an entry

An access-list entry is a line in an access-list that specifies a set of criteria to match against an incoming or outgoing packet, and an action to take if the criteria are met. The criteria can include a source IP address, destination IP address, protocol, port numbers, and other packet attributes. The action can be either to permit the packet to pass through the router or to deny it. Access-list entries are evaluated in order, from top to bottom, and the first entry that matches the packet's characteristics determines the action taken. If no match is found, the packet is denied by default.

Rules(or entries) can be defined in access-list configuration mode

Since access lists use *first match* approach, entries have priorities. It is by default sequential(the first entry has the highest priority), and behavior can be changed by using *sequence numbers*

To enter the ACL config mode, it is just required to enter ACL's name.

```
soodar(config)# ip access-list ACL_TEST
soodar(config-nacl)# permit any
```

An access list can contain both the standard and extended ACL rules.

(1-4294967295) <deny|permit> <any|A.B.C.D/M>

Standard ACL entry rule. It is limited to controlling traffic only based on the source IP address.

- (1-4294967295): specifies the sequence number of the access-list entry. It can be any value between 1 and 4294967295.
- <deny|permit>: specifies whether the traffic should be denied or permitted.
- <any|A.B.C.D/M>: specifies the source IP address or range of addresses that the access-list entry applies to. any means any source IP address, while A.B.C.D/M represents a specific network prefix.

For example, the following commands creates an access-list entry with a sequence number of 20 that permits traffic from any source IP address and an access-list entry with a sequence number of 10 that denies traffic from the IP address 192.168.1.1:

```
soodar(config)# ip access-list ACL_TEST
soodar(config-nacl)# 20 permit any
soodar(config-nacl)# 20 deny 192.168.1.1/32
```

(1-4294967295) <deny|permit> <any|A.B.C.D/M> <any|A.B.C.D/M> [reflect] [exact-match]

The command is used to create an access-list entry that specifies whether traffic is allowed or denied based on the source and destination IP addresses.

- (1-4294967295): specifies the sequence number of the access-list entry. It can be any value between 1 and 4294967295.
- <deny|permit>: This specifies whether the traffic matching the access-list entry should be allowed or denied.
- <any|A.B.C.D/M>: This specifies the source IP address or network for the traffic. any can be used to match any source IP address. Alternatively, A.B.C.D/M represents a specific network prefix.
- <any|A.B.C.D/M>: This specifies the destination IP address or network for the traffic. any can be used to match any destination IP address. Alternatively, a specific network prefix can be specified using the A.B.C.D/M format.
- reflect: The reflect option make this entry stateful. When using this option, the router will keep track of the outgoing connection(s) and it will automatically add the return entry to the access-list.
- exact-match: When it is entered, the prefixes are also checked and should be the same(192.168.1.1/24 is not matched 192.168.1.1/16). This is useful for applications such as BGP route filtering, where a specific match is required rather than a range of addresses.

Note: The dynamic reflexive entry is added to the same ACL, so for the ACL to be completely useful, it should be applied on both directions on the interface.

Note: exact-match option is used in route-maps and has no effects in normal packet filtering.

For example, to create an access-list entry that allows traffic from any source to the destination network 192.168.1.0/24 , the command would be:

```
soodar(config)# ip access-list TEST_ACL1
soodar(config-nacl)# 10 permit any 192.168.1.0/24
```

(1-4294967295) <deny|permit> <any|X:X::X:X/M>

The command is used to create IPv6 ACL entries that can be used to filter traffic based on source IPv6 addresses.

- (1-4294967295): specifies the sequence number of the access-list entry. It can be any value between 1 and 4294967295.
- <deny|permit>: This specifies whether the traffic matching the access-list entry should be allowed or denied.
- <any|X:X::X:X/M>: The any keyword can be used to match any source IPv6 address. Alternatively, a specific IPv6 address or subnet can be specified using the X:X::X:X/M notation, where M is the prefix length.

(1-4294967295) <deny|permit> <any|X:X::X:X/M> <any|X:X::X:X/M> [reflect] [exact-match]

this command can be used to filter IPv6 traffic based on specific source and/or destination addresses.

- (1-4294967295): specifies the sequence number of the access-list entry. It can be any value between 1 and 4294967295.
- <deny|permit>: This specifies whether the traffic matching the access-list entry should be allowed or denied.

- <any|X:X::X:X/M>: The any keyword can be used to match any source IPv6 address. Alternatively, a specific IPv6 address or subnet can be specified using the X:X::X:X/M notation, where M is the prefix length.
- <any|X:X::X:X/M>: The any keyword can be used to match any destination IPv6 address. Alternatively, a specific IPv6 address or subnet can be specified using the X:X::X:X/M notation, where M is the prefix length.
- reflect: The reflect option make this entry stateful. When using this option, the router will keep track of the outgoing connection(s) and it will automatically add the return entry to the access-list.
- exact-match: When it is entered, the prefixes are also checked and should be the same. This is useful for applications such as BGP route filtering, where a specific match is required rather than a range of addresses.

In this example, the access-list is named TEST_ACLV6 and it permits traffic from the source IPv6 address 2001:db8:1234::1/64 to any destination, while denying all other traffic.

```
soodar(config)# ipv6 access-list TEST_ACLV6
soodar(config-ipv6-acl)# permit ipv6 2001:db8:1234::1/64 any
```

(1-4294967295) <deny|permit> PROTOSERVICE <any|A.B.C.D/M> <any|A.B.C.D/M> [reflect] [exact-match]

(1-4294967295) <deny|permit> PROTOSERVICE <any|X:X::X:X/M> <any|X:X::X:X/M> [reflect] [exact-match]

The command is used to create an access-list entry to control traffic based on protocol type and service.

- (1-4294967295): specifies the sequence number of the access-list entry. It can be any value between 1 and 4294967295.
- <deny|permit>: This specifies whether the traffic matching the access-list entry should be allowed or denied.
- PROTOSERVICE: Specifies the protocol or service for which the access list will apply. The protocol is specified as either a name or a number(e.g: pim, sctp, eigrp…), and the service(e.g: ssh, http, …) is specified as a name.
- <any|A.B.C.D/M>: This specifies the source IP address or network for the traffic. any can be used to match any source IP address. Alternatively, A.B.C.D/M represents a specific network prefix.
- <any|A.B.C.D/M>: This specifies the destination IP address or network for the traffic. any can be used to match any destination IP address. Alternatively, a specific network prefix can be specified using the A.B.C.D/M format.
- reflect: The reflect option make this entry stateful. When using this option, the router will keep track of the outgoing connection(s) and it will automatically add the return entry to the access-list.
- exact-match: When it is entered, the prefixes are also checked and should be the same. This is useful for applications such as BGP route filtering, where a specific match is required rather than a range of addresses.

Example :

```
soodar(config)# ip access-list SSH_DENY
soodar(config-nacl)# deny ssh any any
soodar(config-nacl)# permit any
soodar(config-nacl)# ipv6 access-list SSH6_DENY
soodar(config-ipv6-acl)# deny ssh any any
```

(continues on next page)

(continued from previous page)

```
soodar(config-ipv6-acl)# permit any

Deny any ``SSH`` connection. This entry creates a rule which denies *TCP*␣
↪connection to port *22* from the source.

::

   ip access-list TEST_ACL1 deny tcp any le 65535 any eq 22
```

(1-4294967295) <deny|permit> tcp <any|A.B.C.D/M> SRC_PORT <any|A.B.C.D/M> DST_PORT [TCP FLAGS] [reflect] [exact-match]

(1-4294967295) <deny|permit> tcp <any|X:X::X:X/M> SRC_PORT <any|X:X::X:X/M> DST_PORT [TCP FLAGS] [reflect] [exact-match]

The command is used to create an access list that matches TCP traffic based on the source IP address, destination IP address, source port, destination port, and TCP flag values.

- `(1-4294967295)`: specifies the sequence number of the access-list entry. It can be any value between 1 and 4294967295.
- `<deny|permit>`: This specifies whether the traffic matching the access-list entry should be allowed or denied.
- `tcp`: This indicates that the access list entry is for TCP traffic.
- `<any|A.B.C.D/M>`: This specifies the source IP address or network for the traffic. `any` can be used to match any source IP address. Alternatively, A.B.C.D/M represents a specific network prefix.
- **SRC_PORT: Specifies the source port(s) that the traffic must match using an operator keyword and a source port. A source port can be a number or a service name. The port number or named service can range from 0 to 65535. Examples of named services include "ftp-data", "http", "telnet", "ssh", and many others. The operator keywords are defined below:**
  - `gt PORT`: means "greater than", and will match any TCP traffic with a source or destination port number greater than the specified number or service.
  - `lt PORT`: means "less than", and will match any TCP traffic with a source or destination port number less than the specified number or service.
  - `eq PORT`: means "equal to", and will match any TCP traffic with a source or destination port number equal to the specified number or service.
  - `range LOWER_PORT UPPER_PORT`: Specifies a range of ports and will match any TCP traffic with a source or destination port number between the lower and upper ports.
- `<any|A.B.C.D/M>`: This specifies the destination IP address or network for the traffic. `any` can be used to match any destination IP address. Alternatively, a specific network prefix can be specified using the A.B.C.D/M format.
- `DST_PORT`: like `SRC_PORT`, but applies for destination port.
- `TCP FLAGS`: This specifies the TCP flag values to be matched. It can include any combination of the following flags: FIN, SYN, RST, PSH, ACK, and URG.
- `reflect`: The reflect option make this entry stateful. When using this option, the router will keep track of the outgoing connection(s) and it will automatically add the return entry to the access-list.
- `exact-match`: When it is entered, the prefixes are also checked and should be the same. This is useful for applications such as BGP route filtering, where a specific match is required rather than a range of addresses.

For example, the following example creates an IP access list named "PERMIT_TRUSTED" that permits TCP traffic from any source IP address in the 10.0.0.0/8 network with a source port number less than 1000 to any destination IP address and any destination port number.

```
soodar(config)# ip access-list PERMIT_TRUSTED
soodar(config-nacl)# permit tcp 10.0.0.0/8 lt 1000 any
```

Or consider this example:

```
soodar(config)# ip access-list DENY_FROM_HTTP_TO_HTTPS
soodar(config-nacl)# deny tcp 10.0.0.0/8 range 80 443 any
soodar(config-nacl)# permit any any
```

In the above example, The first command denies TCP traffic with source IP addresses in the 10.0.0.0/8 subnet and destination ports between 80 and 443 (inclusive) to any destination IP address. This effectively blocks web traffic(and other TCP services that use a port between 80 and 443) from hosts in the 10.0.0.0/8 subnet to any destination.

The second command permits all other traffic to any destination, effectively allowing all traffic that is not blocked by the first command.

The IP6 equivalent of above access-lists is like this:

```
soodar(config)# ipv6 access-list PERMIT_TRUSTED6
soodar(config-ipv6-acl)# permit tcp fc00::/8 lt 1000 any
soodar(config)# ipv6 access-list DENY_FROM_HTTP_TO_HTTPS6
soodar(config-nacl)# deny tcp fc00::/8 range 80 443 any
soodar(config-nacl)# permit any any
```

Also, the DENY_FROM_HTTP_TO_HTTPS could be rewritten like this:

```
soodar(config)# deny tcp 10.0.0.0/8 range http https any
soodar(config)# permit any any
```

(1-4294967295) <deny|permit> udp <any|A.B.C.D/M> SRC_PORT <any|A.B.C.D/M> DST_PORT [reflect] [exact-match]

(1-4294967295) <deny|permit> udp <any|X:X::X:X/M> SRC_PORT <any|X:X::X:X/M> DST_PORT [reflect] [exact-match]

The command is used to create an access list that matches UDP traffic based on the source IP address, destination IP address, source port, destination port, and UDP flag values.

- (1-4294967295): specifies the sequence number of the access-list entry. It can be any value between 1 and 4294967295.
- <deny|permit>: This specifies whether the traffic matching the access-list entry should be allowed or denied.
- udp: This indicates that the access list entry is for UDP traffic.
- <any|A.B.C.D/M>: This specifies the source IP address or network for the traffic. any can be used to match any source IP address. Alternatively, A.B.C.D/M represents a specific network prefix.
- **SRC_PORT: Specifies the source port(s) that the traffic must match using an operator keyword and a source port. A source port can be a number or a service name. The port number or named service can range from 0 to 65535. Examples of named services include "ftp-data", "http", "telnet", "ssh", and many others. The operator keywords are defined below:**

  - `gt PORT`: means "greater than", and will match any UDP traffic with a source or destination port number greater than the specified number or service.
  - `lt PORT`: means "less than", and will match any UDP traffic with a source or destination port number less than the specified number or service.
  - `eq PORT`: means "equal to", and will match any UDP traffic with a source or destination port number equal to the specified number or service.
  - `range LOWER_PORT UPPER_PORT`: Specifies a range of ports and will match any UDP traffic with a source or destination port number between the lower and upper ports.
- <any|A.B.C.D/M>: This specifies the destination IP address or network for the traffic. any can be used to match any destination IP address. Alternatively, a specific network prefix can be specified using the A.B.C.D/M format.
- DST_PORT: like SRC_PORT, but applies for destination port.
- reflect: The reflect option make this entry stateful. When using this option, the router will keep track of the outgoing connection(s) and it will automatically add the return entry to the access-list.
- exact-match: When it is entered, the prefixes are also checked and should be the same. This is useful for applications such as BGP route filtering, where a specific match is required rather than a range of addresses.

```
soodar(config)# ip access-list DENY_DNS
soodar(config-nacl)# deny udp any eq domain any

   The purpose of this ACL is to deny all DNS traffic (UDP port 53) from any source␣
↪to any destination. This can be used to block DNS traffic from leaving a network␣
↪or to prevent external DNS servers from being used by devices on the network.
```

(1-4294967295) <deny|permit> icmp <any|A.B.C.D/M> <any|A.B.C.D/M> ICMP_TYPE_CODES [reflect] [exact-match]"

(1-4294967295) <deny|permit> icmp <any|X:X::X:X/M> <any|X:X::X:X/M> ICMPV6_TYPE_CODES [reflect] [exact-match]"

This command is used to create an access control list (ACL) rule that either permits or denies Internet Control Message Protocol (ICMP) traffic between two IP addresses or subnets. The command consists of the following parameters:

- (1-4294967295): specifies the sequence number of the access-list entry. It can be any value between 1 and 4294967295.
- <deny|permit>: This specifies whether the traffic matching the access-list entry should be allowed or denied.
- icmp: This indicates that the access list entry is for ICMP traffic.
- <any|A.B.C.D/M>: This specifies the source IP address or network for the traffic. any can be used to match any source IP address. Alternatively, A.B.C.D/M represents a specific network prefix.
- <any|A.B.C.D/M>: This specifies the destination IP address or network for the traffic. any can be used to match any destination IP address. Alternatively, a specific network prefix can be specified using the A.B.C.D/M format.
- ICMP_TYPE_CODES: Specifies the type and code of the ICMP traffic being matched. It could be a name(For example, "echo-request" (type 8, code 0) or "echo-reply" (type 0, code 0)) or two numbers indicating the type and code.
- reflect: The reflect option make this entry stateful. When using this option, the router will keep track of the outgoing connection(s) and it will automatically add the return entry to the access-list.

- `exact-match`: When it is entered, the prefixes are also checked and should be the same. This is useful for applications such as BGP route filtering, where a specific match is required rather than a range of addresses.

Negate an entry

Just use `no` form of command

Example in config mode:

```
soodar(config)# ip access-list ACL_TEST
soodar(config-nacl)# no 100
soodar(config-nacl)# no 10 deny tcp 10.0.0.0/8 eq www 64.233.185.113/32
soodar(config-nacl)# no deny udp any 8.8.8.8 eq 53
```

You can negate an entry by using its sequence number, its definition, or both of them.

ACL Resequencing

ACL resequencing is a process of changing the sequence numbers of access control list (ACL) entries. ACLs are applied to network devices to filter traffic based on specific rules. When there is a need to modify an existing ACL, it may be necessary to add or delete an entry in the middle of the list. In such cases, resequencing is performed to maintain the logical order of ACL entries and to ensure that traffic is processed correctly.

For example, suppose an ACL has ten entries, and a new entry needs to be added between entries five and six. To insert the new entry, the administrator needs to change the sequence numbers of entries six through ten. The new entry will be added with a sequence number of six, and the existing entries will be shifted down accordingly. This ensures that traffic is processed in the correct order and that the new entry is evaluated before the subsequent entries. But another solution is to resequence all entries in such a way that there is at least 1 value in between each entry(an increment of 2).

`ip access-list resequence ACL4 (1-2147483647)$start-seq-num (1-2147483647)$increment`

`ipv6 access-list resequence ACL6 (1-2147483647)$start-seq-num (1-2147483647)$increment`

The ip access-list resequence command is used to resequence an existing access control list (ACL). The command is used to change the sequence numbers of the ACL entries, allowing you to insert or delete rules in the ACL without having to completely recreate it.

- `ACL`: Specifies the name of the ACL you want to resequence.
- `start-seq-num`: Specifies the starting sequence number for the resequenced ACL.
- `increment`: Specifies the increment value to use for the new sequence numbers.

For example, if you have an ACL named ACL_TEST and you want to start the sequence numbers at 1 with an increment of 2, you would use the following command:

```
soodar(config)# ip access-list resequence ACL_TEST 1 2
```

This will change the sequence numbers for the ACL entries to start at 1 and increment by 2 for each subsequent rule.

Apply ACL

Applying an Access Control List (ACL) to an interface on a network device allows the network administrator to control the traffic that flows through that interface. The ACL is used to permit or deny traffic based on criteria such as source/destination IP address, protocol, and port number. The ACL is created and then applied to the inbound or outbound direction of the interface. The rules in the ACL are then processed in order from top to bottom until a match is found, and the corresponding action (permit or deny) is taken. This can help secure a network by preventing unwanted traffic from entering or leaving a particular interface.

ip access-group ACL4 in

The command is used to apply an IPv4 access control list (ACL) to an inbound interface. This command specifies that packets entering the interface should be checked against the ACL. If a packet matches a permit statement in the ACL, it is forwarded. If it matches a deny statement, it is dropped. This command is commonly used to filter incoming traffic from untrusted networks and restrict access to resources in a network.

- `ACL4`: Specifies the name of the ACL you want to apply.
- `in`: Indicates the ACL is applied to incoming traffic.

ipv6 traffic-group ACL6 in

The command is used to apply an IPv6 Access Control List (ACL) to an ingress interface of a device. It directs the device to evaluate incoming IPv6 packets against the specified ACL before forwarding them to their destination.

When the device receives a packet on the ingress interface, it checks the packet against the ACL rules specified in ACL6. If the packet matches a permit rule in the ACL, the device forwards the packet to the next hop. If the packet matches a deny rule in the ACL, the device drops the packet.

- `ACL6`: Refers to the name of the IPv6 ACL that has been configured on the device.
- `in`: specifies that the ACL is being applied to the ingress traffic of the interface.

The negating form, detaches an ACL from interface's input.

For example, the following command applies the IPv4 ACL named "IN_ACL" and the IPv6 ACL name "IN_ACL6" to the inbound traffic of interface ge3:

```
soodar(config)# interface ge3
soodar(config-if)# ip access-group IN_ACL in
soodar(config-if)# ipv6 traffic-group IN_ACL6 in
```

ip access-group ACL4 out

The command is used to apply an IPv4 access control list (ACL) to an interface for traffic leaving the interface.

- `ACL4`: Specifies the name of the ACL you want to apply.
- `out`: Indicates the ACL is applied to leaving traffic.

ipv6 traffic-group ACL6 out

When this command is configured on an interface, it causes the specified IPv6 access list to be applied to all traffic that leaves the interface.

- `ACL6`: Refers to the name of the IPv6 ACL that has been configured on the device.
- `out`: specifies the direction of traffic flow to which the access list should be applied.

The negating form, detaches an ACL from interface's output.

ip access-group ACL4 in out

ipv6 traffic-group ACL6 in out

Apply ACL to both ways of traffic. The negating form, detaches an ACL from interface.

Debug

show ip access-list [NAME] [json]

show ipv6 access-list [NAME] [json]

The command is used to display the details of an access control list (ACL).

When executed without any options, the command displays a list of all configured ACLs on the router along with their type (standard or extended) and their current line count. When a specific ACL is specified with the NAME parameter, the command will display the details of that particular ACL, including each line of the ACL and whether each line permits or denies traffic.

- NAME (Optional): Specifies the name of the access list to display.
- json (Optional): Displays the output in JSON format.

```
soodar# show ip access-list
IP access list TEST
    seq 10 permit tcp 1.1.1.10/32 eq 200 2.1.1.0/24 gt 5060
    seq 20 deny   1.2.1.0/24 3.1.1.0/24
soodar# show ip access-list json
{
  "ZEBRA":{
    "TEST":{
      "type":"Zebra",
      "addressFamily":"IPv4",
      "remark":"",
      "rules":[
        {
          "sequenceNumber":10,
          "filterType":"permit",
          "protocol":6,
          "prefix":"1.1.1.10\/32",
          "prefix-dest":"2.1.1.0\/24",
          "src-port-first":200,
          "src-port-last":65535,
          "src-port-operator":"equal",
          "dest-port-first":5060,
          "dest-port-last":65535,
          "dest-port-operator":"greater-than",
          "tcp-flags":"",
          "exact-match":false
        },
        {
          "sequenceNumber":20,
          "filterType":"deny",
          "protocol":0,
          "prefix":"1.2.1.0\/24",
          "prefix-dest":"3.1.1.0\/24",
          "src-port-first":0,
          "src-port-last":65535,
          "src-port-operator":"range",
          "dest-port-first":0,
          "dest-port-last":65535,
          "dest-port-operator":"range",
```

(continues on next page)

(continued from previous page)

```
          "tcp-flags":"",
          "exact-match":false
        }
      ]
    }
  }
}
```

show ip access-list interfaces

The command is used to display information about all interfaces where an access list is applied or where an access list is not applied. This command is useful to verify which interfaces are being affected by an access list, and it can be helpful when troubleshooting access list-related issues.

The output of the command includes the following information for each interface:

- Interface Name
- Egress ACLs
- Ingress ACLs

Debugging logs can be set in case of need.

[no] debug acl event

log data plane installation processes and results

1.11 VRF

1.11.1 VRF

Define a VRF

Virtual Routing and Forwarding (VRF) is a technology that enables multiple isolated routing tables on a single physical router. Each VRF instance maintains its own routing table, which is separate and independent from the global routing table of the router. This allows multiple virtual networks to coexist on the same physical infrastructure while maintaining their privacy and isolation from one another.

Each VRF has its own set of interfaces, routes, and routing protocols, which means that traffic that enters one VRF is unaware of the existence of other VRFs on the same router. This allows for a more flexible and secure network design, where different departments or customers can have their own virtual network that is logically separated from other networks.

VRFs are commonly used in service provider environments, where they provide a way to offer VPN services to customers using the same infrastructure. VRFs can also be used in enterprise networks to separate traffic between departments or projects, or to connect to different cloud providers or internet service providers using separate routing tables.

vrf VRF_NAME

The vrf command is used to create a Virtual Routing and Forwarding (VRF) instance in a device.

- `VRF_NAME`: is the name of the VRF instance to be created.

Example :

```
soodar(config)# vrf vrf-green
```

This creates a VRF instance named `vrf-green`. Once the VRF instance is created, it can be used to configure routing protocols, interface settings, and other network services for that particular VRF.

Add an interface to VRF

`ip vrf forwarding NAME`

The command is used to enable a Virtual Routing and Forwarding (VRF) instance and associate it with an interface or sub-interface.

- `NAME`: is the name of the VRF that will be enabled and associated with the interface.

Example :

```
soodar(config)# vrf vrf-green
soodar(config)# interface ge1
soodar(config-if)# ip vrf forwarding vrf-green
```

In the above example, the VRF named vrf-green is enabled and associated with the ge1 interface. Any traffic that enters this interface is associated with the VRF and is forwarded according to the routing table configured for that VRF.

`no ip vrf forwarding [NAME]`

The command is used to disable the forwarding of packets to a Virtual Routing and Forwarding (VRF) instance on an interface. In other words, it associates the interface with the default VRF.

Note: When adding/removing interfaces to/from a VRF, make sure no valid IP is set on it.

Example:

```
n1(config)# int ge3
n1(config-if)# ip vrf forwarding vrf-green
n1(config-if)# ip address 200.1.2.20/24
```

VRF Configuration examples

VRF Trunking

Example :

```
n1(config)# int ge1.100
n1(config-if)# encapsulation dot1q 100
n1(config-if)# ip vrf forwarding vrf-green
n1(config-if)# ip address 200.1.2.20/24
n1(config)# int ge2
n1(config-if)# rewrite tag push 1 dot1q 300
```

Dynamic routing in VRF

Example:

```
soodar3(config)# router ospf vrf vrf-green
soodar3(config-router)# network 200.2.3.0/24   area 0
soodar3(config-router)# network 3.2.1.0/24   area 0
soodar3(config-if) # end
```

VRF FIB

Example:

```
soodar1# sh ip ospf vrf vrf-green route
soodar1# sh ip fib vrf vrf-green
soodar1# sh ip fib vrf all
```

Display VRF info

show vrf

Example:

```
n1# sh vrf
vrf vrf-blue id 5 table 300
```

Logging

Debugging logs can be set in case of need.

[no] debug vrf event

logs data plane installation processes and results

1.12 MPLS

1.12.1 MPLS

Enable MPLS on interface

mpls ip

mpls ipv6

Example :

```
soodar(config)# int ge0
soodar(config-if)# mpls ip
soodar(config)# int ge3
soodar(config-if)# mpls ipv6
```

Note: LDP router-id and discovery transport-address should be set before enabling MPLS.

1.13 Security

1.13.1 Tunnels

In computer networking, a tunnel is a virtual point-to-point connection that allows for the encapsulation of one type of network protocol within another. Tunnels can be used to connect networks that are physically separated by a public network, such as the Internet, or to create secure connections between remote locations.

Tunnels are commonly used to provide secure remote access to private networks, as well as to connect geographically dispersed networks. They are also used in conjunction with other network technologies, such as virtualization, to provide additional security and flexibility.

SoodarOS support many Layer 2 and Layer 3 tunnels, including: `GRE`, `IPIP`, `VXLAN` and `VPLS`

Layer 3 Tunnels

Layer 3 Tunnels Includes:

- GRE
- IPIP

GRE

A GRE (Generic Routing Encapsulation) tunnel is a type of tunneling protocol that encapsulates one protocol over another protocol to create a virtual point-to-point connection between two network endpoints. In GRE tunneling, a tunnel is established between two routers or endpoints, and the router at one end encapsulates the payload packets of one protocol inside the payload of another protocol before transmitting them over the tunnel to the other end.

The GRE header provides the necessary information to enable the transport of the encapsulated packets over the tunnel. The GRE header includes a protocol type field that identifies the payload protocol, a checksum field to ensure data integrity, a key field for security purposes, and a sequence number field to ensure data order. GRE tunnels can be used to transport any network-layer protocol, including IPv4, IPv6, and multicast traffic.

One of the primary applications of GRE tunneling is to create secure virtual private network (VPN) connections over the Internet. GRE tunnels are often used in combination with other tunneling protocols such as IPsec to provide security and encryption for data transmitted over the tunnel. GRE tunnels can also be used to connect remote branch offices or to provide connectivity between cloud-based services and on-premises networks.

IPIP

IP-IP tunnel, also known as IP-in-IP tunnel or IPIP tunnel, is a type of tunneling protocol that encapsulates one IP packet within another IP packet. The protocol allows for the creation of a virtual point-to-point link between two network nodes over an IP network.

In an IP-IP tunnel, the original IP packet is encapsulated in a new IP packet and transmitted over the network. The original packet becomes the payload of the new packet, with the source and destination addresses of the original packet becoming the new IP packet's tunnel endpoints. The new IP packet is then transmitted across the network to the receiving endpoint, where it is decapsulated, revealing the original packet. Security measures, such as encryption and authentication, can also be implemented to protect tunnel traffic from unauthorized access.

Create L3 Tunnel

interface tunnel [vrf VRF] (0-1023)

The interface tunnel command is used in network configuration to create a tunnel interface. A tunnel interface is a logical interface that is used to encapsulate one protocol inside another protocol.

- `(0-1023)`: is used to identify the tunnel interface.
- `VRF`: (Optional) specifies the name of the VRF (Virtual Routing and Forwarding) instance to which the tunnel interface belongs.

tunnel source <A.B.C.D|X:X::X:X>

The tunnel source command is used to specify the source IP address of a tunnel interface

- `<A.B.C.D|X:X::X:X>`: specifies the source IP address of the tunnel. It can be an IPv4 or IPv6 address.

It is important to configure the correct source IP address for the tunnel, as this will determine which interface the encapsulated packets appear to originate from on the receiving end.

tunnel destination <A.B.C.D|X:X::X:X>

The command is used to configure the tunnel endpoint address for a GRE or IP-IP tunnel.

- `A.B.C.D`: specifies the IPv4 address of the destination endpoint.
- `X:X::X:X`: specifies the IPv6 address of the destination endpoint.

tunnel vrf VRF

The command is used to configure the VRF in which the tunnel lookup for its destination.

- `VRF`: specifies the VRF name to look up the destination from.

tunnel mode ipip

The command is used to configure a tunnel with the IP-IP encapsulation method.

tunnel mode ipsec

The command is used to configure a tunnel with the protected IP-IP encapsulation method. An IPSec tunnel is an IP-IP tunnel that should always be protected.

tunnel mode gre

The command is used to configure the tunneling protocol as GRE (Generic Routing Encapsulation) on a network device.

tunnel protection ipsec profile IPSECPROFILE

The command is used to configure a tunnel with IPsec encryption. It specifies an IPsec profile that is applied to the tunnel for securing the data that is transmitted over the tunnel.

- `profile IPSECPROFILE`: This specifies the name of the IPsec profile that will be used for the tunnel. The IPsec profile defines the security policies and algorithms that are used for securing the data that is transmitted over the tunnel.

See also:

Profile

Warning: Currently, the protection mode is only supported in P2P tunnels.

Note: When a tunnel is in protected mode, It is put in shutdown mode until the IPSec SAs are established.

Note: When IPSec SAs protecting a tunnel are gone, the tunnel immediately shuts down. Reestablishing SAs make tunnel available again.

Logging

Debugging logs can be set in case of need.

`debug tunnel event`
log data plane installation processes and results

GRE configuration example

In the first peer we have:

```
soodar1(config)# interface tunnel 10
soodar1(config-if)# tunnel source 200.1.2.1
soodar1(config-if)# tunnel destination 200.1.2.2
soodar1(config-if)# tunnel mode gre
soodar1(config-if)# ip address 192.168.1.1/32
```

In the second peer we have:

```
soodar2(config)# interface tunnel 10
soodar2(config-if)# tunnel source 200.1.2.2
soodar2(config-if)# tunnel destination 200.1.2.1
soodar2(config-if)# tunnel mode gre
soodar2(config-if)# ip address 192.168.1.2/32
```

And then, we add IP routes:

```
soodar1(config)# ip route 2.1.1.0/24 tunnel10
```

```
soodar2(config)# ip route 1.1.1.0/24 tunnel10
```

GRE-MP configuration example

Currently, only NHRP static mapping is available.

In the first peer we have:

```
soodar1(config)# interface tunnel 10
soodar1(config-if)# tunnel source 200.1.2.1
soodar1(config-if)# tunnel mode gre multipoint
soodar1(config-if)# ip address 192.168.1.1/32
soodar1(config-if)# ip nhrp map 192.168.1.2 200.1.2.2
```

In the second peer we have:

```
soodar2(config)# interface tunnel 10
soodar2(config-if)# tunnel source 200.1.2.2
soodar2(config-if)# tunnel mode gre multipoint
soodar2(config-if)# ip address 192.168.1.2/32
soodar2(config-if)# ip nhrp map 192.168.1.1 200.1.2.1
```

And then, we add IP routes:

```
soodar1(config)# ip route 2.1.1.0/24 192.168.1.2
```

```
soodar2(config)# ip route 1.1.1.0/24 192.168.1.1
```

Layer 2 tunnels

VXLAN

VXLAN, or Virtual Extensible LAN, is a network virtualization technology used to extend Layer 2 Ethernet networks over an IP-based network infrastructure. It is designed to address the scalability issues that arise in large cloud computing environments.

The basic idea behind VXLAN is to encapsulate Ethernet frames in UDP packets, with an additional VXLAN header added to the packet. The VXLAN header contains a VNI (VXLAN Network Identifier) that is used to identify the virtual network that the packet belongs to.

The VXLAN tunnel endpoint (VTEP) is the device responsible for encapsulating and decapsulating Ethernet frames into VXLAN packets. VTEPs can be implemented in software or hardware and can be located on hypervisors, switches, routers, or servers.

VXLAN also supports multicast and unicast traffic forwarding. Multicast traffic is forwarded using multicast group addresses, while unicast traffic is forwarded to a specific VTEP address.

VXLAN allows for network virtualization at scale, providing a way to create virtual networks that span across physical data center boundaries. It enables workload mobility, allowing virtual machines to be migrated between data centers without the need for manual IP address reconfiguration. Additionally, VXLAN can help overcome the limitations of VLANs, such as the maximum number of VLAN IDs and the scalability issues associated with large Layer 2 domains.

`interface nve (0-100000000)`

The command is used to create and configure a Network Virtualization Endpoint (NVE) interface. NVE is a technology used for virtualizing network segments over a Layer 3 infrastructure.

- `(0-100000000)`: specifies the NVE interface number. The interface number can be any number between 0 and 100000000.

Example :

```
soodar(config)# interface nve 40
```

source-ip <A.B.C.D|X:X::X:X>

This command is used to specify the source IP address for the VXLAN tunnel endpoint (VTEP) that is associated with the NVE interface.

- `<A.B.C.D|X:X::X:X>`: is the IPv4 or IPv6 address that will be used as the source address for the VTEP.

ingress-replication A.B.C.D

The ingress-replication command is used to specify the IP address for the ingress replication. Ingress replication is a VXLAN technology used to forward traffic between two VXLAN VTEPs when multicast is not available in the underlay network. In ingress replication, the traffic is replicated by the source VTEP and sent to all the destination VTEPs directly. The ingress replication method is also known as "head-end replication."

- `A.B.C.D`: is the IP address of the NVE that will receive replicated traffic from the source VTEP.

Note: Currently, only one address can be used for replication.

Note: SoodarOS does not support multicast for VXLAN.

---

member vni (1-16777214)

The command is used in the context of an NVE (Network Virtualization Endpoint) interface to configure a VNI (VXLAN Network Identifier) as a member of the interface. A VNI is an identifier used to differentiate multiple VXLAN overlay networks running on the same physical network infrastructure.

- `(1-16777214)`: Specifies the VNI number to be added to the NVE interface. The range is from 1 to 16777214.

Note: For now, each NVE interface can associate to 1 VNI

---

member vni (1-16777214) associate-vrf

Associate NVE to VNI number and VRF that use this VNI number. Now tunnel lookup its *ingress-replication*'s path from the VRF that shares the same VNI with the tunnel.

Note: Each VRF can associate to 1 VNI

Example :

```
soodar(config)# interface nve 10
soodar(config-if)# source-ip 200.1.3.1
soodar(config-if)# ingress-replication 156.25.4.89
soodar(config-if)# member vni 40
soodar(config-if)# bridge-group 120
soodar(config-if)# int ge0
soodar(config-if)# no shutdown
soodar(config-if)# bridge-group 120
```

```
soodar(config)# vrf green
soodar(config-vrf)# vni 40
soodar(config)# int ge1
soodar(config-if)# ip vrf forwarding green
soodar(config-if)# ip address 200.1.3.1/24
soodar(config)# interface nve 10
soodar(config-if)# source-ip 200.1.2.1
soodar(config-if)# ingress-replication 200.1.3.3
soodar(config-if)# member vni 40 associate-vrf
soodar(config-if)# bridge-group 120
soodar(config-if)# int ge0
soodar(config-if)# no shutdown
soodar(config-if)# bridge-group 120
```

Logging

Debugging logs can be set in case of need.

debug vxlan event

log data plane installation processes and results

VPLS

Virtual Private LAN Service(VPLS) is a method to extend LANs on the network. SoodarOS Supports VPLS on an MPLS core network. First, we need to create a `mpls-tunnel` interface to achieve this. This interface acts as a pseudowire by adding another *MPLS label* to its passing traffic. Using a TLDP(Targeted LDP) session, the `tunnel label` can be negotiated between two routers.

Note: To use VPLS, the connection should be full-mesh. If three router `r1`, `r2` and `r3` are going to form a VPLS, 3 connection is needed: `r1-r2`, `r2-r3` and `r1-r3`

---

interface mpls-tunnel

Creates a `mpls-tunnel` interface

l2vpn NAME type vpls

Create an L2VPN using VPLS technology

member pseudowire PW

Add a `mpls-tunnel` to this L2VPN and enter member pseudowire configuration mode. PW is mpls-tunnel's name

neighbor lsr-id A.B.C.D

Target's LSR-ID of this pseudowire.

Note: We should have a route to target's LSR-ID to establish a targeted session.

---

pw-id (1-4294967295)

An ID to distinguish pseudowires. If PW-IDs differ, the session will not be established.

Example :

```
soodar(config)# interface ge3
soodar(config-if)# bridge-group 200
soodar(config-if)# no shutdown
soodar(config)# interface mpls-tunnel0
soodar(config-if)# bridge-group 200 split-horizon group 100
soodar(config-if)# no shutdown
soodar(config)# interface mpls-tunnel1
soodar(config-if)# bridge-group 200 split-horizon group 100
soodar(config-if)# no shutdown
soodar(config)# mpls ldp
soodar(config-ldp)# router-id 222.1.1.1
soodar(config)# l2vpn exemplary-vpls type vpls
soodar(config-l2vpn)# member pseudowire mpls-tunnel0
soodar(config-l2vpn-pw)# neighbor lsr-id 222.7.7.7
soodar(config-l2vpn-pw)# pw-id 170
soodar(config-l2vpn)# member pseudowire mpls-tunnel1
soodar(config-l2vpn-pw)# neighbor lsr-id 222.14.14.14
soodar(config-l2vpn-pw)# pw-id 1140
```

Note: Note how mpls-tunnels share the same split-horizon group id. It's to prevent loops in packets(since bridge flooding is enabled and our topology is full-mesh).

Logging

Debugging logs can be set in case of need.

debug vpls event

log data plane installation processes and results

1.13.2 PKI

PKI stands for Public Key Infrastructure. It is a set of technologies, policies, and procedures used to create, manage, distribute, use, store, and revoke digital certificates and public-private key pairs.

The goal of PKI is to provide secure and trustworthy communication over the network by establishing a trusted digital identity for individuals, devices, and organizations. It is used for a variety of purposes, such as authentication, digital signatures, encryption, and secure email.

PKI is based on the use of asymmetric cryptography, where each entity has a public key and a private key. The public key can be shared with others to encrypt messages, while the private key is kept secret and used to decrypt messages. Digital certificates are used to bind the public key to a specific entity and provide proof of its authenticity.

Digital Certificate

A digital certificate is an electronic document that is issued by a trusted third-party organization, such as a Certificate Authority (CA), to verify the identity of an individual, organization, or device. It contains information about the identity of the entity, such as its name, public key, and other relevant details, and is digitally signed by the CA to guarantee its authenticity.

Digital certificates play a critical role in Public Key Infrastructure (PKI) and are used for a variety of purposes, such as:

1. Authentication: Digital certificates are used to verify the identity of an individual, organization, or device, allowing for secure communication and transactions.
2. Encryption: Digital certificates can be used to encrypt sensitive data, ensuring that only the intended recipient can access it.
3. Digital signatures: Digital certificates can be used to create and verify digital signatures, which assure that a message or document has not been altered in transit.

CA

CA (Certificate Authority) stands for Certificate Authority. It is a trusted third-party organization that is responsible for issuing and managing digital certificates used in a Public Key Infrastructure (PKI).

The main function of a CA is to verify the identity of an individual, organization, or device requesting a digital certificate, and then issue a certificate that contains the verified public key and other information about the entity. The CA's signature on the certificate serves as proof that the certificate is genuine and the public key belongs to the identified entity.

CAs are essential in PKI as they help establish trust in digital communication by ensuring that the digital certificate belongs to the intended entity and has not been tampered with. CAs can be either commercial or operated by a government or a non-profit organization.

In addition to issuing digital certificates, CAs also manage certificate revocation and renewal and maintain the security and confidentiality of the private keys associated with the certificates they issue.

Note: All PKI actions are permanent jobs; It does not appear in `running config` but is preserved after the router reboot.

Note: Currently, no certificate revocation method is supported.

Key Generation

A pair of private/public keys are used for issuing certificate requests or used in other protocols.

`crypto key generate rsa label NAME modulus (2048|4096)`

The command is used to generate a Rivest-Shamir-Adleman (RSA) public-private key pair on a device.

Here is a breakdown of the command and its options:

- `label NAME`: This option specifies a label to be assigned to the generated key pair. The label can be any alphanumeric string and is used to identify the key pair when multiple key pairs are present on the device.
- `modulus (2048|4096)`: This option specifies the size of the RSA key modulus, which determines the strength of the encryption. The modulus size can be either 2048 or 4096 bits.

For example, to generate an RSA key pair with a label of "my_key" and a modulus size of 2048 bits, you would use the following command:

```
soodar(config)# crypto key generate rsa label my_key modulus 2048
```

Once the key pair is generated, the public key can be shared with other devices or clients to establish secure communication using encryption and digital signatures. The private key must be kept confidential and secure, as it is used to decrypt encrypted traffic and sign digital messages.

crypto key generate x25519 label LABEL

This command generates a new X25519 key pair with a specific label that can be used for secure communication or other cryptographic purposes.

- `label`: Theparameter is used to specify a descriptive label for the generated key pair.

crypto key generate raw label LABEL bytes (32-1024)

The crypto key generate raw command is used to generate a raw key that can be used for various cryptographic functions.

- `label`: The name or label of the key to be generated. This is an alphanumeric string up to 20 characters long.
- `bytes`: The size of the key to be generated, in bytes. This can be any value between 32 and 1024.

When this command is executed, a raw key of the specified size is generated and stored on the device. It's important to note that this command generates a raw key, which is a binary string of random data, and not a passphrase or password.

crypto key generate ssh modulus (2048|4096)

This command is used to generate an RSA public/private key pair for Secure Shell (SSH) on the devices.

- `modulus`: Specifies the size of the RSA modulus to be used. The modulus size can be either 2048 or 4096 bits.

Note: Currently, generated private keys are non-exportable.

Note: Although the keys are non-exportable, there's an option to take a backup from the device keys.

Trustpoint

A trustpoint is a certificate management tool used to establish trust between a device and other devices or services. It is used to store and manage digital certificates and keys that are used for secure communication.

Trustpoints can be used to manage self-signed certificates, as well as certificates issued by trusted third-party Certificate Authorities (CAs).

In order to use a trustpoint, a digital certificate must first be obtained and imported into the trustpoint. In case of using a self-signed CA, the trustpoint should first be authenticated and import the CA, then import the certificate. Once the certificate is imported, it can be used to establish secure connections with other devices and services that recognize the certificate as trusted.

Importing a CA

First, to import a CA, we need to define a `trustpoint`. After defining `trustpoint`, authentication is needed to import the CA. This certificate could be self-signed, and the SSH terminal is the input(SoodarOS administrator should copy/paste the certificate).

Note: All inputs/outputs(including certificate, CSR and...) are in PEM format

`crypto pki trustpoint NAME`

The command is used to configure a PKI (Public Key Infrastructure) trustpoint on a device. A trustpoint is a configuration object that specifies a trusted identity or entity in the context of PKI operations.

- `NAME`: The argument specifies the name of the trustpoint being configured.

`crypto pki authenticate TP`

Authenticate the trustpoint TP and write its CA to the non-volatile memory.

Example :

```
n1(config)# crypto pki trustpoint root-ca
n1(config)# crypto pki authenticate root-ca
Enter the base 64 encoded CA certificate
End with a blank line or the word "quit" on a line by itself

-----BEGIN CERTIFICATE-----
MIIDSzCCAjOgAwIBAgIIQMT8Qv03sXYwDQYJKoZIhvcNAQELBQAwNTELMAkGA1UE
BhMCSVIxEzARBgNVBAoTClRlbXAgQ29ycC4xETAPBgNVBAMTCHRlbXAuY29tMB4X
DTIxMDEyMDExNDIzNFoXDTI0MDEyMDExNDIzNFowNTELMAkGA1UEBhMCSVIxEzAR
BgNVBAoTClRlbXAgQ29ycC4xETAPBgNVBAMTCHRlbXAuY29tMIIBIjANBgkqhkiG
9w0BAQEFAAOCAQ8AMIIBCgKCAQEAy1KPgdCS6BB7PCdeggnsf6NjW4KBxeG6H18R
lOHYoTBMlR3QrvCrpgoZv3DtGR8T6Ch0/HdL1GdFJ7RcJqZPbaxdepqI08SZG4VD
CcZbIOdCNgKWD+jaO0vgyfcK2cXKY70bdyUuJLwNvSvPEPhzH1UNx7kfBdvGn2Vg
s/XyYhsn3xc6ioODT+HUAAd2WvBIOzd+RUo0yANJRKbPnLPgpNEiE1wG6Bj6orjR
ajnC8SYt5XGqD0DX7JGi7bELHw0JGdDk1acr9GQyjwVobDYCKDTuW4ELDsS+2GIK
E76rmlAGrJGy3po2itVbmMprhbTl3EOpxPzl78qkG/r0i4lUXQIDAQABo18wXTAP
BgNVHRMBAf8EBTADAQH/MA4GA1UdDwEB/wQEAwIBBjAdBgNVHQ4EFgQU7CsuL8vJ
o0kfANvQjVQkaR4K/WQwGwYDVR0RBBQwEoIQb3RoZXIuZG9tYWluLmNvbTANBgkq
hkiG9w0BAQsFAAOCAQEAHe8iOUjW8+BNBCfyyfcQOokd7UuK/0DE40wEXVRpMzyv
4IoLNnz5SmWBZo5WdtkIUfGMc9ll8uRsBpIcqHOR8ZSRkjswtOFn+C5KxNXum1pQ
cLmNpxn2ecsr2K2qW6IRfig8cQwzpFe3c59zFf13gKdr6g0B+lpx/hMBdhyaUn6A
9uXtvgeCzAqdJehpo12IKNnYeL+GrfHcFe7R7BRLD2XzoAgjFR48w24h3FbrxM8I
1jqEwbvnGT7FECGZbyKGBEM/dY1gbVD19GTJlaZ8z3HrHdaRFvCYgAqFLTVtU8Q+
lq+EWiCSMRlPPx1OiLDddbxRw2JIjdF7XIsU3WGhtw==
-----END CERTIFICATE-----
Updating certificates in /etc/ssl/certs...
1 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d...
done.

n1# show crypto pki certificate root-ca
Trustpoint: root-ca
CA:
  subject:  "C=IR, O=Temp Corp., CN=temp.com"
```

(continues on next page)

(continued from previous page)

```
issuer:   "C=IR, O=Temp Corp., CN=temp.com"
validity:  not before Jan 20 15:12:34 2021, not valid yet (valid in 58 seconds)
           not after  Jan 20 15:12:34 2024, ok (expires in 1095 days)
serial:    40:c4:fc:42:fd:37:b1:76
altNames:  other.domain.com
flags:     CA CRLSign self-signed
subjkeyId: ec:2b:2e:2f:cb:c9:a3:49:1f:00:db:d0:8d:54:24:69:1e:0a:fd:64
pubkey:    RSA 2048 bits
keyid:     cf:d8:04:82:62:b9:f1:a9:84:75:56:e7:1b:5b:ac:4a:c8:ba:ae:21
subjkey:   ec:2b:2e:2f:cb:c9:a3:49:1f:00:db:d0:8d:54:24:69:1e:0a:fd:64
Fingerprint: 954E9105EEE221C7BCDF351BBA0184E950F82C75
```

Generate a certificate and CSR

Users can request a certificate signing and import that certificate. To do this, a `trustpoint` needs to have these parameters set: 1. Certificate's SN(and optionally some SANs) 2. A RSA key pair to create and sign the CSR. 3. An enrollment method. Currently, only the SSH terminal(copy and paste) method is available and could be skipped. After setting up `trustpoint`, and authenticating it, a CSR should be generated. If terminal enrollment is used, the PKCS#10 format CSR is printed on the screen, and SoodarOS administrator needs to copy it and sign it by a CA. To import this signed certificate, an authentication for this `trustpoint` is needed.

subject-name LINE...

The command is used within a trustpoint configuration to specify the distinguished name (DN) of the certificate subject.

- `LINE`: One or more lines of DN.

For example, to configure a trustpoint with the DN "CN=example.com, O=Example Inc., C=IR", the following commands could be used:

```
soodar(config)# crypto pki trustpoint TP1
soodar(ca-trustpoint)# subject-name CN=example.com, O=Example Inc., C=IR
```

subject-alt-name LINE

The command is used in a trustpoint configuration mode to specify additional subject names for the certificate.

- `LINE`: Specifies the subject alternative name value.

This command can be used to specify additional subject names for the certificate, such as DNS names, email addresses, or IP addresses.

Note: Enter the command multiple times to set multiple SANs. Up to 100 SANs are supported.

---

no subject-alt-name LINE

Remove a SAN from trustpoint.

rsakeypair KEY

Use previously-generated key pair KEY to sign CSR

enrollment terminal pem

Enroll via terminal(copy and paste), including PEM encapsulation boundaries.

crypto pki enroll TP

Generate a Certificate Signing Request for trustpoint TP. If terminal enrollment is used, the PKCS#10 format CSR is printed on the screen

crypto pki import TP certificate

Import the trustpoint TP's general-purpose certificate and write it to non-volatile memory.

Note: Imported general-purpose certificate should be signed by the same CA that the `trustpoint` is authenticated, or else, it fails to import.

Example :

```
n1(config)# crypto key generate rsa label mycert-key modulus 2048
n1# show crypto key mycert-key
Keypair Label: mycert-key
  Algorithm:   RSA
  Modulus:     2048 bits
  Subject key: fcc893035eda7e736d0a612bad1d000612c87724
  Key ID:      E5611192FEAD3FDFA877A0BAC5F336480A8C2D97
n1(config)# crypto pki trustpoint mycert
n1(ca-trustpoint)# subject-name C=IR, O=My Org, CN=my.org
n1(ca-trustpoint)# subject-alt-name other.my.org
n1(ca-trustpoint)# subject-alt-name other2.my.com
n1(ca-trustpoint)# rsakeypair mycert-key
n1(config)# crypto pki authenticate mycert
Enter the base 64 encoded CA certificate
End with a blank line or the word "quit" on a line by itself

-----BEGIN CERTIFICATE-----
MIIDODCCAiCgAwIBAgIIM7DVFqEvgxgwDQYJKoZIhvcNAQELBQAwOjELMAkGA1UE
BhMCQ0gxEzARBgNVBAoTCnN0cm9uZ1N3YW4xFjAUBgNVBAMTDXN0cm9uZ1N3YW4g
Q0EwHhcNMjAxMTExMDk1OTEzWhcNMjMxMTExMDk1OTEzWjA6MQswCQYDVQQGEwJD
SDETMBEGA1UEChMKc3Ryb25nU3dhbjEWMBQGA1UEAxMNc3Ryb25nU3dhbiBDQTCC
ASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBANjhd9ZFsCS4O3TcnXWFy/cr
wXnVCxev6g5XecHG0A+jaOS6MyJowjJU/CY5S8/LWKIBlKFhdswDT0LaPodnKw8e
RVGwAfQSYb8OymUeHByzxxfhqcCjYu0qWdb2Tf9yVadkt//qW5n2F78j3prFlZ4o
pbG1sLhACY+729iJxB7dg5DKXxECBzSiMo2dScZpQKuADiev4g7TmEH0u3MUa9zU
CzIhoqjzEJ1wF4YC7Y6BZxQU4c04RZGctaOmKRUT0NfVGbqseJHsJVZSCDFud/ls
48tDmQ08GULFNFlFAeGWcUnLle2sorsB+zjfQrJQJBtE/RuoKZ3ODK+ZwGH8wHEC
AwEAAaNCMEAwDwYDVR0TAQH/BAUwAwEB/zAOBgNVHQ8BAf8EBAMCAQYwHQYDVR0O
BBYEFNET3aeJu4082kUYI8TpeBK4w61sMA0GCSqGSIb3DQEBCwUAA4IBAQCB2ciJ
D197+CIwL/DveAJf7Bt0cMD2lPwY4hsHUyHridX2B/t6EMOOujWPouSeBYjLBz7s
akHwh3G9Yx4wlS+k+du5AbkQHMnYigeO4rul+tCg7FzouxFtKEcD6T707DnSEkP+
iA9mLeKxCK3P4vGY2H9x6McqZ1aM55xmdEbvD3QhUMLePBk4aMVKyOr4yWRQgUPB
oBqRVSEvthOyXEWtPkqxY72O/5IQmHDSncBP/D+wiC2wQsYQZhmDoN6d74OqkcBr
HMWDCUM1b8RfVBTeIKvkvQ14BgwPveO99E+P6rrNhdxRA8BwmnNyMvrd81Z1FDU/
J+XkIuPRfz33vO00
-----END CERTIFICATE-----
Updating certificates in /etc/ssl/certs...
1 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d...
done.
```

(continues on next page)

(continued from previous page)

```
n1(ca-trustpoint)# enrollment terminal pem
n1(config)# crypto pki enroll mycert
-----BEGIN CERTIFICATE REQUEST-----
MIICrTCCAZUCAQAwLzELMAkGA1UEBhMCSVIxDzANBgNVBAoTBk15IE9yZzEPMA0G
A1UEAxMGbXkub3JnMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAtvWm
Xi+CtKrJndPw09hPOnTO8DSDIJqi3GdcNDVRcdKb/FB+/C++Vyb2vOLNICxCmRJH
RnoZKPnWqRHWyHeVCNr+Da+bFYHXd4LyaZtCzEoUrmULMyBWGmbUfUlfFpOCa4yq
28qV1BjYXEm93X56XIaT/WpqXELihJC2nnBPxhkLHA80fLmQPZdOzytrjeJt1Rvn
I/PpI+OzEN9/pUvGLv29wfzUN2T9WGdIY/SJuyafQ2972juRA2OTTSsMSOxM4fuj
Mk116RixYvHCd454gehPKOqMUHbXKZ7tQXPaDFtiQIgNqBMz4AlT40Wn3GsODV8Y
AtJ9UOvhmMW1iTHC2wIDAQABoDkwNwYJKoZIhvcNAQkOMSowKDAmBgNVHREEHzAd
ggxvdGhlci5teS5vcmeCDW90aGVyMi5teS5jb20wDQYJKoZIhvcNAQELBQADggEB
AKwvB+bPTMpU2t3HE6CA0mLA9ufc9EqWx2YCTyddTJ8Qp7xhdXywzB64R5Um/mqy
x7lMEyS69pZzTMivm28piIEplSdjKSiHmRpVZsXGWvhpz1alqA6h5IaWlm9s3Bga
YKBmaC0uEsuhXnAxFBPtbwWSaGN0uD5kKTkwZXMxKv4gVkTbrdZfZ2uJR2CiZu1q
yb7u47MeZF4xfcnvFZCuUjlLmpFXMLXjYuNywJP6U/i1DpSG07mDYcnEfS9Ku/o/
gdNBahSspRtBVOx4QtnN4bGZ0MDEn5cEBuWcN4dNbE30dn70NKaNe1DhdKQ/lUxQ
qyIP+5tc2i8GoJsL9wyWJIo=
-----END CERTIFICATE REQUEST-----

n1(config)# crypto pki import mycert certificate
Enter the base 64 encoded CA certificate
End with a blank line or the word "quit" on a line by itself

-----BEGIN CERTIFICATE-----
MIIDMTCCAhmgAwIBAgIIVmyRIVfPsKowDQYJKoZIhvcNAQELBQAwNTELMAkGA1UE
BhMCSVIxEzARBgNVBAoTClRlbXAgQ29ycC4xETAPBgNVBAMTCHRlbXAuY29tMB4X
DTIxMDEyMDExNDgzNloXDTI0MDEyMDExNDgzNlowLzELMAkGA1UEBhMCSVIxDzAN
BgNVBAoTBk15IE9yZzEPMA0GA1UEAxMGbXkub3JnMIIBIjANBgkqhkiG9w0BAQEF
AAOCAQ8AMIIBCgKCAQEAtvWmXi+CtKrJndPw09hPOnTO8DSDIJqi3GdcNDVRcdKb
/FB+/C++Vyb2vOLNICxCmRJHRnoZKPnWqRHWyHeVCNr+Da+bFYHXd4LyaZtCzEoU
rmULMyBWGmbUfUlfFpOCa4yq28qV1BjYXEm93X56XIaT/WpqXELihJC2nnBPxhkL
HA80fLmQPZdOzytrjeJt1RvnI/PpI+OzEN9/pUvGLv29wfzUN2T9WGdIY/SJuyaf
Q2972juRA2OTTSsMSOxM4fujMk116RixYvHCd454gehPKOqMUHbXKZ7tQXPaDFti
QIgNqBMz4AlT40Wn3GsODV8YAtJ9UOvhmMW1iTHC2wIDAQABo0swSTAfBgNVHSME
GDAWgBTsKy4vy8mjSR8A29CNVCRpHgr9ZDAmBgNVHREEHzAdggxvdGhlci5teS5v
cmeCDW90aGVyMi5teS5jb20wDQYJKoZIhvcNAQELBQADggEBAGbt3R0FyA48FWUh
eoud1zh6ujrg0PgFjOhAMnWaln8nXdhMjJJvOI/MZtcyl7fghXr1Asr2M9I3KMxh
BbBefCci5+94g+QucP/R0v5/fzFpiV8gRYXD8o7UWyYanQG5SUyTCdpR5vXxVbEW
FXp3Yk1HBYXDe09AK9AGwRVFHTkaaPze8U5FyJpbrjDZuD/cbkN4lFn+lw49JahO
cVqYXyY84rHjvbq98081NsittSa4QUqBNo8nUXYj+yLuNiV39Zh1pWzl/kugy0yR
mvrqC3irZGXeJbSLDaAT1LdJhiu2Axc7EjwKxcNK+GiXyN/B/7JJrWLL0u6xaA9L
ezbvqQw=
-----END CERTIFICATE-----
Installed successfully

n1# show crypto pki certificate mycert
Trustpoint: n1Cert
CA:
  subject:  "C=IR, O=Temp Corp., CN=temp.com"
  issuer:   "C=IR, O=Temp Corp., CN=temp.com"
```

(continues on next page)

(continued from previous page)

```
  validity:  not before Jan 20 15:12:34 2021, ok
             not after  Jan 20 15:12:34 2024, ok (expires in 1094 days)
  serial:    40:c4:fc:42:fd:37:b1:76
  altNames:  other.domain.com
  flags:     CA CRLSign self-signed
  subjkeyId: ec:2b:2e:2f:cb:c9:a3:49:1f:00:db:d0:8d:54:24:69:1e:0a:fd:64
  pubkey:    RSA 2048 bits
  keyid:     cf:d8:04:82:62:b9:f1:a9:84:75:56:e7:1b:5b:ac:4a:c8:ba:ae:21
  subjkey:   ec:2b:2e:2f:cb:c9:a3:49:1f:00:db:d0:8d:54:24:69:1e:0a:fd:64
  Fingerprint: 954E9105EEE221C7BCDF351BBA0184E950F82C75

General Purpose Certificate:
  subject:  "C=IR, O=My Org, CN=my.org"
  issuer:   "C=IR, O=Temp Corp., CN=temp.com"
  validity:  not before Jan 20 15:18:36 2021, ok
             not after  Jan 20 15:18:36 2024, ok (expires in 1094 days)
  serial:    56:6c:91:21:57:cf:b0:aa
  altNames:  other.my.org, other2.my.com
  flags:
  authkeyId: ec:2b:2e:2f:cb:c9:a3:49:1f:00:db:d0:8d:54:24:69:1e:0a:fd:64
  subjkeyId: fc:c8:93:03:5e:da:7e:73:6d:0a:61:2b:ad:1d:00:06:12:c8:77:24
  pubkey:    RSA 2048 bits
  keyid:     e5:61:11:92:fe:ad:3f:df:a8:77:a0:ba:c5:f3:36:48:0a:8c:2d:97
  subjkey:   fc:c8:93:03:5e:da:7e:73:6d:0a:61:2b:ad:1d:00:06:12:c8:77:24
  Keypair:   mycert-key
  Fingerprint: D51636591648DBDE21FEEFA4C6DF4B38A96502B5
```

Self-signed Trustpoints

Self-signed certificates are available to generate in SoodarOS PKI system. Set the enrollment method of `trustpoint` to selfsigned, and you are good to go. A self-signed certificate can't be imported or authenticated. Enrolling this `trustpoint` generates the certificate.

Example :

```
n1(config)# crypto key generate rsa label self-signed-key
n1(config)# crypto pki trustpoint self-signed-tp
n1(ca-trustpoint)# enrollment selfsigned
n1(ca-trustpoint)# rsakeypair self-signed-key
n1(ca-trustpoint)# subject-name C=IR, O=Independent Ltd., CN=self.indie.com
n1(config)# crypto pki enroll self-signed-tp
n1# show crypto pki certificate self-signed-tp
Trustpoint: self-signed-tp
CA:
  subject:  "C=IR, O=Independent Ltd., CN=self.indie.com"
  issuer:   "C=IR, O=Independent Ltd., CN=self.indie.com"
  validity:  not before Jan 20 15:45:09 2021, ok
             not after  Jan 20 15:45:09 2024, ok (expires in 1094 days)
  serial:    15:9a:3b:16:34:f9:79:49
  flags:     CA CRLSign self-signed
  subjkeyId: 33:74:e2:a1:5e:d1:49:bf:c7:bf:f7:23:4c:c6:53:a0:07:56:24:09
```

(continues on next page)

(continued from previous page)

```
pubkey:    RSA 2048 bits
keyid:     bd:12:cd:f2:1a:b7:d2:27:82:26:db:51:01:d2:60:0d:48:24:bf:3d
subjkey:   33:74:e2:a1:5e:d1:49:bf:c7:bf:f7:23:4c:c6:53:a0:07:56:24:09
Fingerprint: 89177619D312F1AEFAC0A5C8B9DE5E0196B56F16
```

Removing a private key

Admin can remove unused private keys. Removing is done securely by shredding and zeroing the key file.

crypto key zeroize RSAKEY

Shred a key pair.

Note: Removing a key makes the `trustpoints` using them invalid. It's the admin's duty to take care of this situation and remove unused keys or remove all certificates depending on that key.

Removing a trustpoint

Admin can remove a trustpoint. This action removes the CA and general-purpose certificate(if available) and updates the system CA database.

no crypto pki trustpoint TPNAME

Viewing installed Certificates and keys

After installing a certificate, one can see that certificate with a `show` command.

show crypto pki certificate [CA]

Show available certificates on device. If CA name is not provided, all certificates on the system are shown.

Example :

```
n1# show crypto pki certificate mycert
Trustpoint: n1Cert
CA:
subject:  "C=IR, O=Temp Corp., CN=temp.com"
issuer:   "C=IR, O=Temp Corp., CN=temp.com"
validity:  not before Jan 20 15:12:34 2021, ok
            not after  Jan 20 15:12:34 2024, ok (expires in 1094 days)
serial:    40:c4:fc:42:fd:37:b1:76
altNames:  other.domain.com
flags:     CA CRLSign self-signed
subjkeyId: ec:2b:2e:2f:cb:c9:a3:49:1f:00:db:d0:8d:54:24:69:1e:0a:fd:64
pubkey:    RSA 2048 bits
keyid:     cf:d8:04:82:62:b9:f1:a9:84:75:56:e7:1b:5b:ac:4a:c8:ba:ae:21
subjkey:   ec:2b:2e:2f:cb:c9:a3:49:1f:00:db:d0:8d:54:24:69:1e:0a:fd:64
Fingerprint: 954E9105EEE221C7BCDF351BBA0184E950F82C75

General Purpose Certificate:
subject:  "C=IR, O=My Org, CN=my.org"
```

(continues on next page)

(continued from previous page)

```
issuer:   "C=IR, O=Temp Corp., CN=temp.com"
validity:  not before Jan 20 15:18:36 2021, ok
            not after  Jan 20 15:18:36 2024, ok (expires in 1094 days)
serial:    56:6c:91:21:57:cf:b0:aa
altNames:  other.my.org, other2.my.com
flags:
authkeyId: ec:2b:2e:2f:cb:c9:a3:49:1f:00:db:d0:8d:54:24:69:1e:0a:fd:64
subjkeyId: fc:c8:93:03:5e:da:7e:73:6d:0a:61:2b:ad:1d:00:06:12:c8:77:24
pubkey:    RSA 2048 bits
keyid:     e5:61:11:92:fe:ad:3f:df:a8:77:a0:ba:c5:f3:36:48:0a:8c:2d:97
subjkey:   fc:c8:93:03:5e:da:7e:73:6d:0a:61:2b:ad:1d:00:06:12:c8:77:24
Keypair:   mycert-key
Fingerprint: D51636591648DBDE21FEEFA4C6DF4B38A96502B5
```

show crypto key [[KEY] [json]] [ssh]

Show key information. If a key name is not provided, all keys on the system are shown. Output can be JSON(except SSH)

Example :

```
n1# show crypto key mycert-key-rsa
Keypair Label: mycert-key-rsa
  Algorithm:   RSA
  Modulus:     2048 bits
  Subject key: FCC893035EDA7E736D0A612BAD1D000612C87724
  Key ID:      E5611192FEAD3FDFA877A0BAC5F336480A8C2D97
n1# show crypto key x25519-key
Keypair Label: x25519-key
  Algorithm:   X25519
  Public key:  DEE5089576AD02780EFEF6908034E6BD471C2C6DF7FE68FC77F12C5DFCDB9D59
  Public key base64:  3uUIlXatAngO/vaQgDTmvUccLG33/mj8d/EsXfzbnVk=
n1# show crypto key raw-key
Keypair Label: raw-key
  Algorithm:   Raw
  Length:      256 bits
n1# show crypto key json
{
  "keys":[
    {
      "label":"mycert-key-rsa",
      "algorithm":"RSA",
      "modulus":2048,
      "subject_key":"FCC893035EDA7E736D0A612BAD1D000612C87724",
      "key_id":"E5611192FEAD3FDFA877A0BAC5F336480A8C2D97"
    },
    {
      "label":"x25519-key",
      "algorithm":"X25519",
      "public_key":"DEE5089576AD02780EFEF6908034E6BD471C2C6DF7FE68FC77F12C5DFCDB9D59
↪"
    },
    {
```

(continues on next page)

(continued from previous page)

```
      "label":"raw-key",
      "algorithm":"RAW",
      "length":256
    }
  ]
}
```

1.13.3 Wireguard

WireGuard is a communication protocol and free and open-source software that implements encrypted virtual private networks, and was designed with ease of use, high-speed performance, and low attack surface.

SoodarOS supports Wireguard both as a wireguard server and as a wireguard client.

Interface

To start a wireguard tunnel, first, admin should create a wireguard interface.

`interface wireguard (0-1023)`
: Create a wireguard interface instance.

`wireguard source A.B.C.D`
: Set wireguard tunnel source. Use `0.0.0.0` to automatically select the source IP.

`wireguard private-key X25519KEY`
: Use x25519 key with `X25519KEY` label as wireguard private key.

`wireguard port (1000-65535)`
: Wireguard's UDP listens port. If not provided, use 51820 as the port.

Server

Each wireguard server instance can have multiple peers. Each peer consists of its public-key and allowed IPs(IP ranges that should be routed via tunnel).

`wireguard peer PEER`
: Create a wireguard peer named PEER

`public-key LINE [base64]`
: Peer's x25519 public key in hexadecimal or base64

`allowed-ip A.B.C.D/M`
: Add A.B.C.D/M to peer's allowed IP ranges.

> **Note:** admin can add multiple ranges by issuing allowed-ip command various times.

`no allowed-ip [A.B.C.D/M]`
: Remove A.B.C.D/M from peer's allowed IP ranges. If run without any input, remove all allowed IPs.

`vrf VRF`
: Assign a VRF's FIB to Lookup peer destination.

keepalive (5-120)

Set a persistent keepalive timer.

Note: Currently, disabling persistent keepalive is not implemented.

---

description LINE ...

Add a description of peers.

Client

A wireguard client instance has only one peer, and its peer is the server. This peer should have public-key, allowed IPs(IP ranges that should be routed via tunnel), the server address and its listening port.

endpoint A.B.C.D port (1000-65535)

Set/remove the peer's endpoint address and port.

Modes

Wireguard acts as a static source for routing tables. Each `allowed-IP` entry adds a static route to FIB, and there are no means of adding routes dynamically(for example, by using an OSPF process). A new mode of operation is added to add dynamic routing support to wireguard. This mode is called `routing mode` and supports dynamic routing protocols. In this mode, `allowed-IPs` are no longer used, and all routes are dynamically learned from neighbors.

Note: Each peer is considered a neighbor, and to work correctly, adding the peer's wireguard interface IP as an `allowed-IP` is needed.

Note: Wireguard interface in routing mode acts like a node in a point to multipoint network.

---

wireguard mode <normal|routing>

Set wireguard interface operation mode.

Debug

Admin can see wireguard status by using `show wireguard` command.

show wireguard [(1-1024) PEER] [json]

Show wireguard status. If the instance is not indicated, show all wireguards and their peers' status. Users can specify which instance and which peer of that instance is in their interest. By adding `json` option to the command, the output transforms to JSON.

```
server# show wireguard
Wireguard 10
  Mode: Normal
  Source: 200.1.3.1
  Key: key1
  Public key: 7D61BA2FA556FD7B4AA0D54114575DF6FBC5AB9B96337C4A438E85CDFC77ED7C
  Public-key Base64: fWG6L6VW/XtKoNVBFFdd9vvFq5uWM3xKQ46Fzfx37Xw=
  Port: 5100
```

(continues on next page)

(continued from previous page)

```
  Peer n3:
    Public key: 950A6657CDE2193C786FF4771A46318AB86B9CB60BA071E344E8C094EBEEF662
    Public-key Base64: lQpmV83iGTx4b/R3GkYxirhrnLYLoHHjROjAlOvu9mI=
    Current Endpoint: 200.1.3.3
    Current Source: 200.1.3.1
    Persistent keepalive: 25
    Description: Tehran
    Allowed IPs:
     - 3.1.1.0/24
Wireguard 20
  Source: 200.1.2.1
  Key: key2
  Public key: DF6DEA63D7F5E11F9115C1CAA08D78FAFE6BA003739952B8094BC7AE744D235A
  Public-key Base64: 323qY9f14R+RFcHKoI14+v5roANzmVK4CUvHrnRNI1o=
  Port: 5100

  Peer n2:
    Public key: 1D819A6950BBC16F04D86FBF8AA660434AEE12D77888E2F534641E9E7C51EEE2
    Public-key Base64: HYGaaVC7wW8E2G+/iqZgQ0ruEtd4iOL1NGQennxR7uI=
    Persistent keepalive: 25
    Allowed IPs:
     - 2.1.1.0/24
server# show wireguard 20
Wireguard 20
  Mode: Normal
  Source: 200.1.2.1
  Key: key2
  Public key: DF6DEA63D7F5E11F9115C1CAA08D78FAFE6BA003739952B8094BC7AE744D235A
  Public-key Base64: 323qY9f14R+RFcHKoI14+v5roANzmVK4CUvHrnRNI1o=
  Port: 5100

  Peer n2:
    Public key: 1D819A6950BBC16F04D86FBF8AA660434AEE12D77888E2F534641E9E7C51EEE2
    Public-key Base64: HYGaaVC7wW8E2G+/iqZgQ0ruEtd4iOL1NGQennxR7uI=
    Persistent keepalive: 25
    Allowed IPs:
     - 2.1.1.0/24
n3# show wireguard 10 server
Wireguard 10
  Peer server:
    Public key: 7D61BA2FA556FD7B4AA0D54114575DF6FBC5AB9B96337C4A438E85CDFC77ED7C
    Public-key Base64: fWG6L6VW/XtKoNVBFFdd9vvFq5uWM3xKQ46Fzfx37Xw=
    Endpoint: 200.1.3.1
    Current Endpoint: 200.1.2.2
    Current Source: 200.1.2.1
    Persistent keepalive: 25
    Port: 5100
    Allowed IPs:
     - 1.1.1.0/24
server# show wireguard json
{
```

(continues on next page)

(continued from previous page)


```
  "wireguards":[
    {
      "instance":10,
      "mode":"normal",
      "source":"200.1.3.1",
      "key":"temp",
      "public_key":"7D61BA2FA556FD7B4AA0D54114575DF6FBC5AB9B96337C4A438E85CDFC77ED7C",
      "port":5100,
      "peers":[
        {
          "name":"n3",
          "public_key":"950A6657CDE2193C786FF4771A46318AB86B9CB60BA071E344E8C094EBEEF662
↪",
          "endpoint":"0.0.0.0",
          "current_endpoint":"200.1.3.3",
          "current_source":"200.1.3.1",
          "keepalive":25,
          "port":0,
          "vrf":"default",
          "description":"Tehran",
          "allowed_ips":[
            "3.1.1.0/24"
          ]
        }
      ]
    },
    {
      "instance":20,
      "mode":"normal",
      "source":"200.1.2.1",
      "key":"temp",
      "public_key":"DF6DEA63D7F5E11F9115C1CAA08D78FAFE6BA003739952B8094BC7AE744D235A",
      "port":5100,
      "peers":[
        {
          "name":"n2",
          "public_key":"1D819A6950BBC16F04D86FBF8AA660434AEE12D77888E2F534641E9E7C51EEE2
↪",
          "endpoint":"0.0.0.0",
          "keepalive":25,
          "port":0,
          "vrf":"default",
          "description":"",
          "allowed_ips":[
            "2.1.1.0/24"
          ]
        }
      ]
    }
  ]
}
```


show wireguard [(1-1024) PEER] stats [json]

```
n1# show wireguard stats
Wireguard 10
  Peer n2:
    Packets received: 0 packets, 0 bytes
    Packets transmitted: 36 packets, 5328 bytes

  Peer n1:
    Packets received: 0 packets, 0 bytes
    Packets transmitted: 0 packets, 0 bytes

  Peer n3:
    Packets received: 27 packets, 1676 bytes
    Packets transmitted: 25 packets, 1568 bytes
n1# show wireguard 10 n3 stats
Wireguard 10
  Peer n3:
    Packets received: 36 packets, 2196 bytes
    Packets transmitted: 31 packets, 1880 bytes
n1# show wireguard 10 n3 stats json
{
  "wireguards":[
    {
      "instance":10,
      "peers":[
        {
          "name":"n3",
          "rx_packets":36,
          "rx_bytes":2196,
          "tx_packets":31,
          "tx_bytes":1880
        }
      ]
    }
  ]
}
```

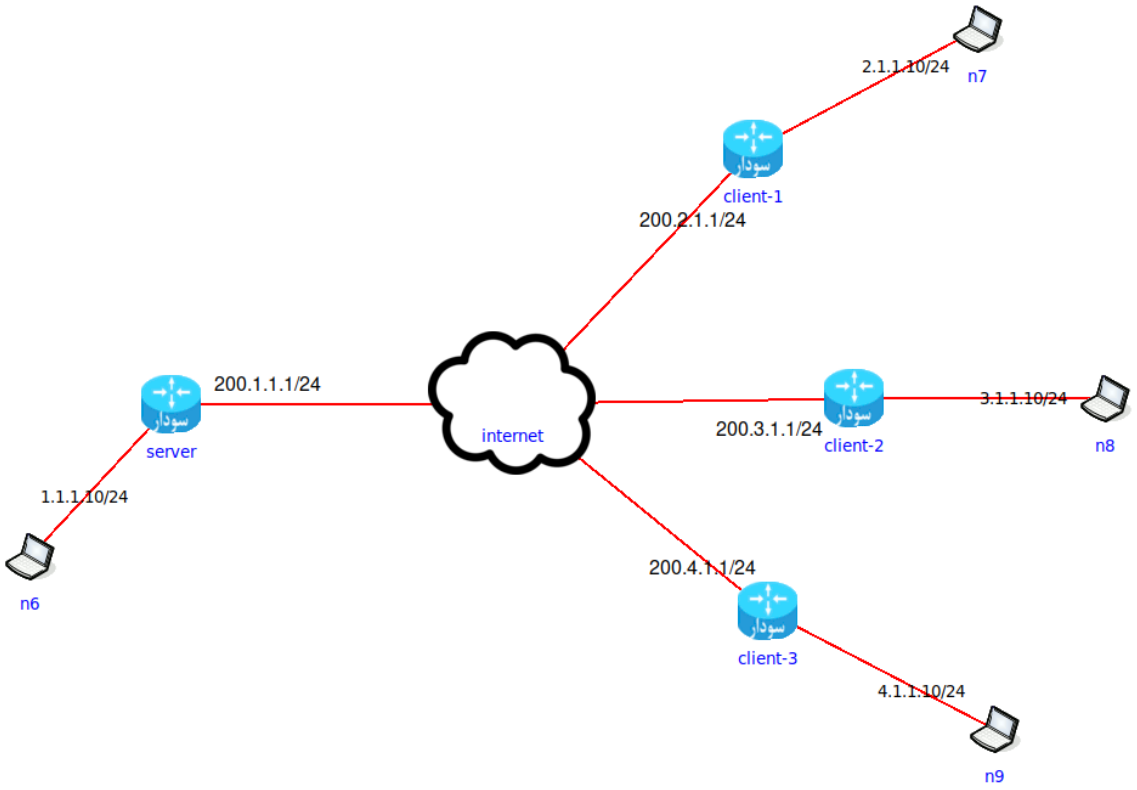
Example

In the following scenario, we want to establish a wireguard server with 3 clients.

First, we create an x25519 private key in the server:

```
server(config)# crypto key generate x25519 label server-key
server# show crypto key server-key
Keypair Label: server-key
  Algorithm:   X25519
  Public key:  D889D845BEED407332B486A1C0A36D310781DD6BE2BB48855636125F16FC8142
```

Then we set up wireguard interface:




```
server(config)# interface wireguard 10
server(config-if)# wireguard source 200.1.1.1
server(config-if)# wireguard private-key server-key
server(config-if)# wireguard port 51820
```

Now we need to set up peers:

```
client-1(config)# crypto key generate x25519 label client1-key
client-1# show crypto key client1-key
Keypair Label: client1-key
  Algorithm:   X25519
  Public key:  85DC0E1B1E8FA87B544863BD44FB7809B85853E4B1FF16E0EFAC70990BA17467
client-1(config)# interface wireguard 1
client-1(config-if)# wireguard private-key client1-key
client-1(config-if)# wireguard source 200.2.1.1
client-1(config-if)# wireguard port 51820
client-1(config-if)# wireguard peer server
client-1(config-wg-peer)# allowed-ip 1.1.1.0/24
client-1(config-wg-peer)# public-key␣
↪D889D845BEED407332B486A1C0A36D310781DD6BE2BB48855636125F16FC8142
client-1(config-wg-peer)# endpoint 200.1.1.1 port 51820
```

We do the same thing for the remaining two clients.

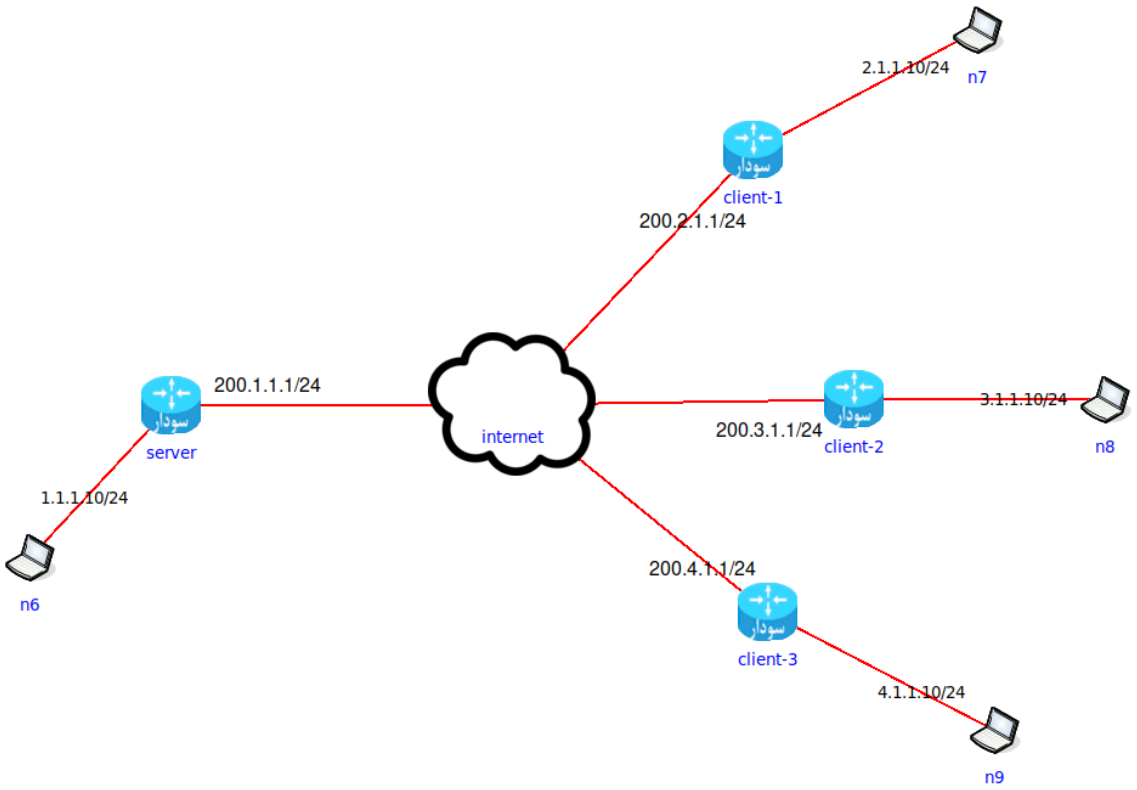
Now that peers are set up, we add peer information to the server:

```
server(config)# interface wireguard 10
server(config-if)# wireguard peer client-1
server(config-wg-peer)# allowed-ip 0.0.0.0/0
server(config-wg-peer)# public-key␣
↪85DC0E1B1E8FA87B544863BD44FB7809B85853E4B1FF16E0EFAC70990BA17467
server(config-if)# wireguard peer client-2
server(config-wg-peer)# allowed-ip 0.0.0.0/0
server(config-wg-peer)# public-key␣
↪5A0882D0D6B757692FDCDFC7BB2413042A333F96EECEE34B69D0E2D7107C7672
server(config-if)# wireguard peer client-3
server(config-wg-peer)# allowed-ip 0.0.0.0/0
server(config-wg-peer)# public-key␣
↪87BC5D410DCDED2D5A9CC443053AC2888947E9724C247CE1FCBE40E12A400293
```

Now on the first packet from each client to `1.1.1.0/24`, the tunnel establishes.

Example With OSPF

In the following scenario, we want to establish a wireguard server with 3 clients.



First, we create an x25519 private key in the server:

```
server(config)# crypto key generate x25519 label server-key
server# show crypto key server-key
Keypair Label: server-key
```

(continues on next page)

(continued from previous page)

```
  Algorithm:   X25519
  Public key:  D889D845BEED407332B486A1C0A36D310781DD6BE2BB48855636125F16FC8142
```

Then we set up wireguard interface:

```
server(config)# interface wireguard 10
server(config-if)# wireguard source 200.1.1.1
server(config-if)# wireguard private-key server-key
server(config-if)# wireguard port 51820
server(config-if)# ip address 10.0.0.1/32
```

Now we need to set up peers:

```
client-1(config)# crypto key generate x25519 label client1-key
client-1# show crypto key client1-key
Keypair Label: client1-key
  Algorithm:   X25519
  Public key:  85DC0E1B1E8FA87B544863BD44FB7809B85853E4B1FF16E0EFAC70990BA17467
client-1(config)# interface wireguard 1
client-1(config-if)# wireguard private-key client1-key
client-1(config-if)# wireguard source 200.2.1.1
client-1(config-if)# wireguard port 51820
client-1(config-if)# wireguard peer server
client-1(config-if)# wireguard mode routing
client-1(config-if)# ip address 10.0.0.2/32
client-1(config-wg-peer)# allowed-ip 10.0.0.1/32
client-1(config-wg-peer)# public-key␣
↪D889D845BEED407332B486A1C0A36D310781DD6BE2BB48855636125F16FC8142
client-1(config-wg-peer)# endpoint 200.1.1.1 port 51820
```

We do the same thing for the remaining two clients.

Now that peers are set up, we add peer information to the server:

```
server(config)# interface wireguard 10
server(config-if)# wireguard peer client-1
server(config-wg-peer)# allowed-ip 10.0.0.2/32
server(config-wg-peer)# public-key␣
↪85DC0E1B1E8FA87B544863BD44FB7809B85853E4B1FF16E0EFAC70990BA17467
server(config-if)# wireguard peer client-2
server(config-wg-peer)# allowed-ip 10.0.0.3/32
server(config-wg-peer)# public-key␣
↪5A0882D0D6B757692FDCDFC7BB2413042A333F96EECEE34B69D0E2D7107C7672
server(config-if)# wireguard peer client-3
server(config-wg-peer)# allowed-ip 10.0.0.4/32
server(config-wg-peer)# public-key␣
↪87BC5D410DCDED2D5A9CC443053AC2888947E9724C247CE1FCBE40E12A400293
```

To start the OSPF process run following commands on all routers

```
server(config)# router ospf
server(config-router)# redistribute connected
server(config-router)# interface wireguard10
```

(continues on next page)

(continued from previous page)

```
server(config-if)# ip ospf network point-to-multipoint
server(config-if)# ip ospf area 0
```

1.13.4 IKEv2

Internet Key Exchange (IKE) is a protocol used to establish a secure connection between two parties over the internet. It is used in combination with IPSec to provide secure communication between two networks or devices. IKE automates the key management process, allowing the devices to negotiate and establish a shared secret key without manual intervention.

There are two versions of IKE, IKEv1 and IKEv2. IKEv1 is an older version and was first introduced in 1998. It is still widely used today, but IKEv2 is gradually replacing it due to its improved features and security enhancements.

IKEv2 was introduced in 2005 as an improvement over IKEv1. It is designed to be faster, more secure, and more flexible than IKEv1. One of the most significant improvements in IKEv2 is its ability to handle network address translation (NAT) traversal more efficiently. IKEv2 uses fewer messages to establish a secure connection, reducing the setup time for the connection. IKEv2 also provides more advanced features, such as the ability to perform client authentication using digital certificates.

Like IPSec, IKEv2 uses a modular CLI for configuration.

`crypto ikev2 dpd (1-3600)`

Set IKEv2 dead peer detection parameters. The first parameter indicates how often a liveness check is performed.

IKEv2 proposal

In IKEv2 (Internet Key Exchange version 2), a proposal is a set of cryptographic algorithms and parameters used for negotiating security associations between two endpoints. A security association (SA) is a set of policies and keys that define how to secure communication between two network entities.

A proposal contains the following parameters:

- `Encryption Algorithm`: Specifies the algorithm used to encrypt data. Examples include AES, DES, and 3DES.
- `Integrity Algorithm`: Specifies the algorithm used to verify the integrity of data. Examples include SHA-1, SHA-256, and MD5.
- `Diffie-Hellman Group`: Specifies the Diffie-Hellman group used to establish the shared secret key. Examples include Group 2, Group 14, and Group 19.
- `PRF Algorithm`: Specifies the pseudorandom function (PRF) used to generate keying material. Examples include SHA-1, SHA-256, and MD5.

IKEv2 supports multiple proposals, allowing endpoints to negotiate the best set of algorithms and parameters for a given connection. During IKEv2 negotiation, each endpoint sends a proposal to the other endpoint, and the two endpoints negotiate to agree on a common proposal.

IKEv2 proposals offer several advantages over IKEv1 proposals, including support for more modern cryptographic algorithms and the ability to negotiate multiple proposals simultaneously. This flexibility allows for greater security and better compatibility with a wider range of network devices.

Note: Currently, SoodarOS only support one set of proposals

`crypto ikev2 proposal IKEPOSAL`

The command is used to create an IKEv2 proposal. An IKEv2 proposal is a set of cryptographic parameters and attributes that are presented during the IKEv2 negotiation process between two devices.

- `IKEPOSAL`: Indicates the name of the IKEv2 proposal being created.

`encryption ALGORITHM`

The encryption command is used to specify the encryption algorithm used for the IKEv2 proposal. Available encryption algorithms are:

- `aes-{128|192|256}`: Specifies the AES CBC encryption algorithm with key lengths of 128, 192, or 256 bits.
- `aes-ctr-{128|192|256}`: Specifies the AES CTR encryption algorithm with key lengths of 128, 192, or 256 bits.
- `aes-gcm-{128|192|256}`: Specifies the AES GCM encryption algorithm with key lengths of 128, 192, or 256 bits.

Note: Only one encryption algorithm can be specified per IKEv2 proposal.

`integrity ALGORITHM`

The integrity command in an IKEv2 proposal is used to specify the integrity algorithm that will be used to ensure data integrity for the IKEv2 session. Available encryption algorithms are:

- `sha-96`: Specifies the SHA-1 hash algorithm for data integrity.
- `sha-256`: Specifies the SHA-256 hash algorithm for data integrity.
- `sha-384`: Specifies the SHA-384 hash algorithm for data integrity.
- `sha-512`: Specifies the SHA-512 hash algorithm for data integrity.

Note: Using a GCM algorithm for encryption makes the integrity algorithm obsolete.

`group GROUP`

The "group" command is used in the IKEv2 proposal configuration mode to specify the Diffie-Hellman (DH) group to be used for key exchange in the proposal. Available groups are:

- `14`: Specifies the use of Diffie-Hellman group 14, which uses a 2048-bit prime modulus.
- `19`: Specifies the use of Diffie-Hellman group 19, which uses a 256-bit elliptic curve.
- `20`: Specifies the use of Diffie-Hellman group 20, which uses a 384-bit elliptic curve.
- `21`: Specifies the use of Diffie-Hellman group 21, which uses a 521-bit elliptic curve.
- `28`: Specifies the use of Diffie-Hellman group 28, which uses a 256-bit Brainpool ECP group.
- `29`: Specifies the use of Diffie-Hellman group 29, which uses a 384-bit Brainpool ECP group.
- `30`: Specifies the use of Diffie-Hellman group 30, which uses a 512-bit Brainpool ECP group.
- `31`: Specifies the use of Diffie-Hellman group 31, which uses Curve 25519.
- `32`: Specifies the use of Diffie-Hellman group 32, which uses Curve 448.

The choice of Diffie-Hellman group affects the strength of the generated keys, with larger groups providing stronger security but also requiring more computational resources.

Example :

```
soodar(config)# crypto ikev2 proposal sample-proposal
soodar(config-ikev2-proposal)# encryption aes-192
soodar(config-ikev2-proposal)# integrity sha-96
soodar(config-ikev2-proposal)# group 28
```

The given commands configure an IKEv2 proposal named "sample-proposal" with the following attributes:

- Encryption algorithm: AES-192
- Integrity algorithm: SHA1-96
- Key exchange group: 28

IKEv2 keyring

IKEv2 Keyring is a configuration element that defines a set of preshared keys or digital certificates that can be used during the negotiation process of the Internet Key Exchange (IKEv2) protocol to establish a secure tunnel between two endpoints.

In an IKEv2 VPN, the keyring is used to authenticate the identity of the remote endpoint and to establish a secure connection. The keyring can store multiple authentication methods (pres shared keys or digital certificates), and they are assigned to a specific peer during the configuration.

`crypto ikev2 keyring IKEKEYRING`

The command is used to configure the IKEv2 keyring, which is a repository for preshared keys (PSKs) and digital certificates used to authenticate peers during IKEv2 negotiation.

- `IKEKEYRING`: Specifies the keyring name.

`peer PEER`

The peer command in IKEv2 keyring configuration mode is used to create a remote peer.

- `PEER`: Specifies the peer name. This name is used locally and has nothing to do with peer's identity.

Note: The `peer` command can be used multiple times under a single keyring to specify multiple remote peers. In this case, each peer is identified by a unique name.

Note: The peer command must be followed by other IKEv2 keyring configuration commands that specify the pre-shared key or certificate authentication and identity for the peer.

`pre-shared-key LINE`

The `pre-shared-key` command specifies a pre-shared key (PSK) used for authentication between IKE peers.

- `psk`: Specifies the pre-shared key value for authentication.

Note: Currently, the same PSK is used for both remote and local authentication. This could change in future releases.

`identity address <A.B.C.D|X:X::X:X>`

It is used to specify the identity of the remote peer as an IPv4 or IPv6 address for which the keyring is being configured. This address is used by the local device to identify the remote peer during the IKE negotiation process.

- <A.B.C.D>: is the IPv4 address identity of the remote peer.
- <X:X::X:X>: is the IPv6 address identity of the remote peer.

identity fqdn FQDN

The command is used to specify the Fully Qualified Domain Name (FQDN) of the remote peer. This FQDN is used by the local device to identify the remote peer during the IKE negotiation process.

- FQDN: The Fully Qualified Domain Name of the remote peer

identity email MAIL

The command is used to specify the email address identity of a remote peer. This command can be useful when establishing VPN connections with peers that are identified by their email addresses.

- MAIL: is the email address of the remote peer.

Example :

```
soodar(config)# crypto ikev2 keyring keyring-1
soodar(config-ikev2-keyring)# peer PC-1
soodar(config-ikev2-keyring-peer)# identity email home@sweet.home
soodar(config-ikev2-keyring-peer)# pre-shared-key 123@321
soodar(config-ikev2-keyring)# peer PC-2
soodar(config-ikev2-keyring-peer)# identity address 1.1.1.1
soodar(config-ikev2-keyring-peer)# pre-shared-key ITSAHARDPASSWD!!
```

The commands configure an IKEv2 keyring named "keyring-1" with two peers, "PC-1" and "PC-2", using different identities and pre-shared keys for authentication.

The first peer, "PC-1", is configured with an email identity of "home@sweet.home" and a pre-shared key of "123@321". The second peer, "PC-2", is configured with an address identity of "1.1.1.1" and a pre-shared key of "ITSAHARDPASSWD!!". When an IKEv2 tunnel is initiated with either of these peers, the local device will attempt to authenticate the remote device using the pre-shared key specified in the keyring configuration. If the remote device can authenticate successfully, the tunnel will be established and traffic can be encrypted and sent between the two devices.

IKEv2 profile

IKEv2 profiles provide a flexible and scalable way to define and control how the IKEv2 protocol is used in a specific context. They allow administrators to tailor IKEv2 to the needs of their specific VPN implementations.

match address local A.B.C.D

The "match address local" command is used in IKEv2 profiles to specify the local address to use for the negotiation process with a remote peer. This command ensures that the local device's address is matched with the address specified in the command for the negotiation process.

This command is useful when the local device has multiple interfaces and IP addresses, and you want to specify a particular IP address to use for IKEv2 negotiations with a remote peer.

- A.B.C.D: Specifies the IP address of the local device.

Note: It's a good practice to set local addresses explicitly. It can prevent problems caused by changes in routes, leading to a change in the source IP address of packets and being rejected by an IKEv2 peer.

`identity local address <A.B.C.D|X:X::X:X>`

The command is used in the configuration of an IKEv2 profile to specify the local identity of the router in the form of an IP address.

In IKEv2, the local identity is used during the authentication process to identify the router to the remote peer. The IP address specified in this command is used as the local identity when the router initiates an IKEv2 session with a remote peer.

- `A.B.C.D`: Specifies the IPv4 address of the local identity.
- `X:X::X:X`: Specifies the IPv4 address of the local identity.

Note: It's important to note that the identity used in IKEv2 negotiations may differ from the actual source IP address used in the IPsec traffic that follows.

`identity local fqdn FQDN`

The command is used to specify the fully qualified domain name (FQDN) for the local identity of the router. The FQDN is a unique name that identifies a device on the network and is composed of the device name and the domain name. The remote peer uses this identity to verify the identity of the local router during the negotiation process.

- `FQDN`: Specifies the fully qualified domain name of the local router

`identity local email MAIL`

The command is used to specify the email address of the local endpoint identity.

- `MAIL`: Specifies the email address of the local endpoint.

`authentication local rsa-sig`

Specify the local authentication method for the router. This command specifies that the router will use RSA digital signature authentication for IKEv2 connections.

RSA signatures provide a more secure method of authentication than PSKs because they rely on the exchange of digital certificates instead of a shared secret. With RSA signatures, each endpoint has its own private key and a public key certificate that can be verified by the other endpoint. The certificate exchange and signature verification process is more complex than using a PSK, but it provides stronger security guarantees.

Note: Since the chosen certificate to use is the one that has the same `SAN` as `local identity`, It is better to use RSA digital signature authentication with the `FQDN` identity.

Note: Note that in order to use RSA signatures for IKEv2 authentication, both endpoints must have a valid digital certificate issued by a trusted Certificate Authority (CA).

`authentication local pre-share`

The command is used to specify that pre-shared key authentication will be used for local authentication in IKEv2 negotiations.

Note: By default, IKEv2 uses a pre-shared key (PSK) for authentication

---

`authentication remote rsa-sig`

This command is used to specify that the remote endpoint must authenticate using RSA digital signatures.

Note: Certificate SAN field is as remote FQDN identity and should be available in keyring.

authentication remote pre-share

The command is used to specify that pre-shared key authentication will be used for remote authentication in IKEv2 negotiations.

match identity remote address <A.B.C.D|X:X::X:X>

This command is used in the IKEv2 profile configuration to match the identity of the remote peer during the IKEv2 negotiation. If the remote peer's identity matches the specified address, the negotiation process continues. If the remote peer's identity does not match the specified address, the negotiation process is terminated.

- A.B.C.D: specifies an IPv4 address identity of the remote peer.
- X:X::X:X: specifies an IPv6 address identity of the remote peer.

Other information about this peer(like PSK) is looked up in the keyring.

match identity remote fqdn FQDN

This command is used to specify the fully qualified domain name (FQDN) of the remote peer that the profile should match.

This command is useful in situations where the remote peer is authenticated using RSA signature. the FQDN value should be the same as the received certificate SAN.

- FQDN: specifies the fully qualified domain name of the remote peer that the profile should match.

Other information about this peer(like PSK) is looked up in the keyring.

match identity remote email EMAIL

specify the email address of the remote identity that the router or firewall is attempting to establish a connection with.

- EMAIL: is the email address of the remote identity.

Other information about this peer(like PSK) is looked up in the keyring.

match certificate

Match against DN fields and values as peer identity. using wildcards is allowed

Note: The peer should use RSA Digital Signature as authentication method, and it should use its DN as identity.

Example:

```
soodar(config-ikev2-profile)# match certificate C=IR, CN=*.temp.ir
```

Match against all peers that have DN as their identity and this certificate is issued from Iran and is a sub-domain of temp.ir

keyring local IKEKEYRING

The keyring local command is used to specify the local keyring that contains the preshared key information for authenticating during IKE negotiations.

- IKEKEYRING: Specifies the name of the keyring that contains the preshared key information.

proposal IKEPOSAL

The "proposal" command in the IKEv2 profile configuration mode is used to specify the IKEv2 proposal to be used for the IKEv2 negotiation.

- IKEPOSAL: Specifies the name of the IKEv2 proposal previously configured.

lifetime <120-86400>

The lifetime command is used in the configuration of IKEv2 profiles and specifies the duration of the security association (SA) in seconds. When the lifetime of the SA expires, a new one is negotiated between the peers.

- <120-86400>: Specifies the number of seconds the SA should remain active, ranging from 120 to 86400 seconds.

Note: Default lifetime for an IKEv2 SA is 14400 seconds.

Example :

```
soodar(config)# crypto ikev2 profile VPN
soodar(config-ikev2-profile)# identity local 192.168.1.1
soodar(config-ikev2-profile)# match identity remote home@sweet.home
soodar(config-ikev2-profile)# keyring local keyring-1
soodar(config-ikev2-profile)# proposal sample-proposal
```

This configuration creates an IKEv2 profile named "VPN" on the device. It specifies the local identity for this profile to be 192.168.1.1, and remote identity to be "home@sweet.home". It then specifies that the "keyring-1" should be used for authentication purposes. Finally, it references an IKEv2 proposal named "sample-proposal" to be used for this profile.

1.13.5 IPSec

IPSec (Internet Protocol Security) is a protocol suite used to provide secure communication over IP networks. It is used to protect data transmitted between two endpoints and can be used to provide confidentiality, authentication, and data integrity. IPSec has two modes of operation: * Transport mode: In transport mode, only the payload (the data being transmitted) is encrypted and the header of the original IP packet is left intact. This mode is typically used for end-to-end communication between hosts. * Tunnel mode: In tunnel mode, the entire original IP packet is encapsulated within a new IP packet with a new header. The original packet's header is encrypted along with the payload. This mode is typically used for site-to-site communication between networks.

IPSec can be used in combination with other protocols, such as IKE (Internet Key Exchange) which is used to establish the IPSec tunnel, and ESP (Encapsulating Security Payload) which provides confidentiality, integrity, and authentication for the data being transmitted.

AH Mode

AH stands for Authentication Header, which is a protocol used in IPsec to provide data authentication and integrity protection. In AH mode, the entire IP packet is authenticated and protected, including the IP header and data payload. The authentication process is achieved by generating a hash of the packet contents and appending it to the packet. This hash is calculated using a shared secret key, which is negotiated during the IPsec phase 1 and 2 negotiations. AH mode does not provide encryption of the packet contents, so it is typically used in combination with ESP (Encapsulating Security Payload) mode to provide both authentication and encryption. AH mode is commonly used in scenarios where data integrity is a critical requirement, such as in financial transactions.

ESP Mode

ESP stands for Encapsulating Security Payload. It is a protocol used in IPsec to provide confidentiality, data origin authentication, and integrity for IP packets. ESP provides confidentiality by encrypting the payload of IP packets, which means that the data being sent cannot be read by unauthorized parties. It also provides data origin authentication and integrity by adding a message authentication code (MAC) to the packet, which ensures that the packet has not been tampered with during transmission.

ESP is used in conjunction with the Authentication Header (AH) protocol in IPsec to provide end-to-end security for IP packets. ESP and AH can be used together or separately, depending on the security requirements of the network. ESP is commonly used in VPNs (Virtual Private Networks) to provide secure communication over the Internet.

Note: Currently, Only ESP or AH could be used and a combination of both is not possible in SoodarOS

IPSec Phases

IPsec uses a two-phase approach to establish a secure connection between two devices:

Phase 1: In the first phase, the devices negotiate and establish a secure channel to protect the subsequent negotiation of the cryptographic keys. The main goals of Phase 1 are:

1. Authentication of the communicating devices
2. Negotiation of a secure method for exchanging encryption keys
3. Establishment of a secure channel for Phase 2 negotiations

Phase 2: Once the secure channel has been established in Phase 1, the devices can negotiate the specific parameters and methods to be used to secure the actual data being transmitted. The main goals of Phase 2 are:

1. Establishment of IPsec Security Associations (SAs) for each direction of communication
2. Negotiation of the specific encryption, authentication, and other security protocols to be used
3. Setting up the keys and other parameters for the selected security protocols

Both of these phases are done automatically with IKEv2.

IKEv2 is a protocol used to establish and manage IPsec VPN tunnels. IKEv2 has three main phases:

- `IKE_SA_INIT`: In this phase, the two endpoints negotiate a security association (SA) for protecting the IKEv2 traffic itself. This involves exchanging proposals for encryption, authentication, and other parameters, and selecting a set of proposals that both sides support. If successful, this phase ends with the establishment of an IKE SA that will be used to protect subsequent IKEv2 traffic.
- `IKE_AUTH`: In this phase, the two endpoints authenticate each other and negotiate a set of IPsec parameters to be used for protecting the user traffic. The authentication process involves exchanging identity information (such as IP addresses or digital certificates) and verifying it using pre-shared keys, certificates, or other mechanisms.
- `CHILD_SA`: In this phase, the two endpoints use the IKE SA to negotiate a set of parameters for the IPsec traffic that will be sent through the tunnel. This involves exchanging proposals for encryption, authentication, and other parameters, and selecting a set of proposals that both sides support. If successful, this phase ends with the establishment of a Child SA that will be used to protect user traffic.

The first and second phases of IKEv2 are for IPSec Phase 1 and the subsequent phase corresponds to the second phase of IPSec.

Transform set

In IPSec, a transform set is a combination of security protocols, and algorithms used to define the security properties of a VPN tunnel. It specifies how the data is encrypted, and authenticated before transmission.

A transform set consists of one or more individual security protocols, which are defined using the following parameters:

- Authentication algorithm: used to authenticate the packet data, ensuring that it has not been tampered with during transmission.
- Encryption algorithm: used to encrypt the packet data, providing confidentiality and preventing unauthorized access.

crypto ipsec transform-set IPSECTS ah hmac HMAC_ALG

The command is used to define an IPsec transform set that uses AH protocol for authentication of IP packets. This command also specifies the hash algorithm used for integrity check of the packet, by specifying the HMAC algorithm.

The options available for the HMAC algorithm are:

- `sha-96`: Uses the SHA-1 algorithm with a 96-bit truncation.
- `sha-256`: Uses the SHA-256 algorithm with a 256-bit truncation.
- `sha-384`: Uses the SHA-384 algorithm with a 384-bit truncation.
- `sha-512`: Uses the SHA-512 algorithm with a 512-bit truncation.

Note: Note that the authentication algorithm specified in the transform set must match on both the local and remote devices in order for the IPsec tunnel to be established.

---

crypto ipsec transfor-set IPSECTS esp {hmac HMAC_ALG |cipher CIPHER_ALG}

The command is used to define an IPsec transform set that uses ESP protocol for authentication and encryption of IP packets. This command also specifies the hash algorithm used for integrity check of the packet, by specifying the HMAC algorithm and, the encryption algorithm by specifying the Cipher algorithm.

The options available for the HMAC algorithm are:

- `sha-96`: Uses the SHA-1 algorithm with a 96-bit truncation.
- `sha-256`: Uses the SHA-256 algorithm with a 256-bit truncation.
- `sha-384`: Uses the SHA-384 algorithm with a 384-bit truncation.
- `sha-512`: Uses the SHA-512 algorithm with a 512-bit truncation.

The options available for the cipher algorithm are:

- `aes-{128|192|256}`: Specifies the AES CBC encryption algorithm with key lengths of 128, 192, or 256 bits.
- `aes-ctr-{128|192|256}`: Specifies the AES CTR encryption algorithm with key lengths of 128, 192, or 256 bits.
- `aes-gcm-{128|192|256}`: Specifies the AES GCM encryption algorithm with key lengths of 128, 192, or 256 bits.

Note: HMAC algorithm is not needed/ignored when using GCM cipher algorithms.

---

mode transport

Specifies that the IPsec transport mode should be used.

mode tunnel

Specifies that the IPsec tunnel mode should be used.

Example :

```
soodar(config)# crypto ipsec transform-set ipsec-tunnel-TS esp hmac sha-96 cipher␣
↪aes-192
soodar(cfg-crypto-trans)# mode transport
```

The result of these commands is to create an IPsec transform set named ipsec-tunnel-TS that uses ESP protocol with HMAC-SHA-96 integrity and AES-192 encryption algorithms in transport mode.

Profile

crypto ipsec profile IPSECPROFILE

Create a new profile IPSECPROFILE.

set transform-set IPSECTS

The command is used in an IPsec profile configuration to specify the transform set that will be used for IPsec encryption and authentication.

- `IPSECTS`: The name of the transform set to be used for IPsec encryption and authentication. The transform set should be pre-configured using the crypto ipsec transform-set command.

Note: An IPSec profile without Transform set is useless.

---

set pfs GROUP

The command is used in the context of configuring IPSec (Internet Protocol Security) on network devices, such as routers and firewalls. It specifies a particular group for Perfect Forward Secrecy. *GROUP* ` specifies the Diffie-Hellman (DH) group to be used for key exchange in the rekey. Available groups are:

- `14`: Specifies the use of Diffie-Hellman group 14, which uses a 2048-bit prime modulus.
- `19`: Specifies the use of Diffie-Hellman group 19, which uses a 256-bit elliptic curve.
- `20`: Specifies the use of Diffie-Hellman group 20, which uses a 384-bit elliptic curve.
- `21`: Specifies the use of Diffie-Hellman group 21, which uses a 521-bit elliptic curve.
- `28`: Specifies the use of Diffie-Hellman group 28, which uses a 256-bit Brainpool ECP group.
- `29`: Specifies the use of Diffie-Hellman group 29, which uses a 384-bit Brainpool ECP group.
- `30`: Specifies the use of Diffie-Hellman group 30, which uses a 512-bit Brainpool ECP group.
- `31`: Specifies the use of Diffie-Hellman group 31, which uses Curve 25519.
- `32`: Specifies the use of Diffie-Hellman group 32, which uses Curve 448.

PFS (Perfect Forward Secrecy) is a security property in IPSec that ensures that even if an attacker manages to compromise the long-term encryption keys, they will not be able to decrypt past or future communications that were or will be protected with those keys. PFS addresses the vulnerability associated with long-term keys by ensuring that a different, temporary key is used for each session or communication. It is acheived by using key exchange algorithms like Diffie-Hellman (DH) or Elliptic Curve Diffie-Hellman (ECDH).

set ikev2 profile IKEPROFILE

The command is used in an IPSec profile configuration to specify an IKEv2 profile for the IPSec tunnel.

- IKEPROFILE: Specifies the name of the IKEv2 profile that has been previously configured.

The IKEv2 profile is used in the first phase of the IPSec tunnel establishment to negotiate the security parameters and create the IKEv2 security association (SA). The IPSec profile specifies the transform set that will be used in the second phase of the tunnel establishment.

Note: An IKEv2 profile should be exclusive to one profile. And a profile using an IKEv2 profile should be bound to a tunnel and can't be used on two tunnels. So for each tunnel, we need an IKEv2 profile and an IPSec profile.

---

set security-association lifetime second (120-28800)

The command is used in an IPSec profile to set the lifetime for the security association (SA) for the IPSec tunnel. The lifetime is the time for which an SA will be active before it expires and a new one must be established.

- (120-28800): The lifetime value in seconds. Valid values are between 120 and 28800 seconds.

Note: Default value is 3600 seconds.

Note: IPSec SAs are installed when the IPSec profile protects a tunnel.

Example: Setup an IPSec profile using IKEv2 and PSK

```
soodar1(config)# crypto ikev2 proposal PROPOSAL
soodar1(config-ikev2-proposal)# integrity sha-96
soodar1(config-ikev2-proposal)# encryption des
soodar1(config-ikev2-proposal)# group 28
soodar1(config-ikev2-proposal)# crypto ikev2 keyring KEY-1
soodar1(config-ikev2-keyring)# peer PC-1
soodar1(config-ikev2-keyring-peer)# idnetity email pc1@local.net
soodar1(config-ikev2-keyring-peer)# pre-shared-key PSKPASS
soodar1(config-ikev2-keyring-peer)# crypto ikev2 profile profile-ike
soodar1(config-ikev2-profile)# identity local address 9.9.9.9
soodar1(config-ikev2-profile)# match identity remote email pc1@local.net
soodar1(config-ikev2-profile)# keyring local KEY-1
soodar1(config-ikev2-profile)# proposal PROPOSAL
soodar(config)# crypto ipsec transform-set ipsec-tunnel-TS esp hmac sha-96 cipher aes-192
soodar1(config)# crypto ipsec profile ipsec-transport-profile
soodar1(ipsec-profile)#  set transform-set ipsec-tunnel-TS
soodar1(ipsec-profile)#  set ikev2 profile profile-ike

These commands configure an IKEv2 VPN between soodar1 and PC-1. The configuration␣
↪includes:

* Creation of an IKEv2 proposal called PROPOSAL with integrity algorithm SHA-96,␣
↪encryption algorithm DES, and Diffie-Hellman group 28.
* Configuration of an IKEv2 keyring called KEY-1, with PC-1 as a peer. PC-1 is␣
↪identified by its email identity pc1@local.net. A pre-shared key PSKPASS is configured␣
```

(continues on next page)

(continued from previous page)

```
↪for authentication.
* Creation of an IKEv2 profile called profile-ike with local identity address 9.9.9.9␣
↪and remote identity email pc1@local.net. The profile uses the keyring KEY-1 and the␣
↪proposal PROPOSAL.
* Configuration of an IPsec transform set named ipsec-tunnel-TS that uses ESP protocol␣
↪with HMAC-SHA-96 integrity and AES-192 encryption algorithms
* Configuration of an IPsec profile called ipsec-transport-profile, which uses the␣
↪transform-set ipsec-tunnel-TS and the IKEv2 profile profile-ike.
```

and in the other router:

```
soodar2(config)# crypto ikev2 proposal PROPOSAL
soodar2(config-ikev2-proposal)# integrity sha-96
soodar2(config-ikev2-proposal)# encryption des
soodar2(config-ikev2-proposal)# group 28
soodar2(config-ikev2-proposal)# crypto ikev2 keyring KEY-1
soodar2(config-ikev2-keyring)# peer PC-2
soodar2(config-ikev2-keyring-peer)# idnetity address 9.9.9.9
soodar2(config-ikev2-keyring-peer)# pre-shared-key PSKPASS
soodar2(config-ikev2-keyring-peer)# crypto ikev2 profile profile-ike
soodar2(config-ikev2-profile)# identity local email pc1@local.net
soodar2(config-ikev2-profile)# match identity remote address 9.9.9.9
soodar2(config-ikev2-profile)# keyring local KEY-1
soodar2(config-ikev2-profile)# proposal PROPOSAL
soodar(config)# crypto ipsec transform-set ipsec-tunnel-TS esp hmac sha-96 cipher aes-192
soodar2(config)# crypto ipsec profile ipsec-transport-profile
soodar2(ipsec-profile)#  set transform-set ipsec-tunnel-TS
soodar2(ipsec-profile)#  set ikev2 profile profile-ike
```

Example: Setup an IPSec profile using IKEv2 and RSA-Sig

We have 2 routers, `soodar1` and `soodar2`. We have a valid CA and a signed certificate for authentication on each of them. `soodar1`'s certificate has `n1.local.net` as `SAN, `` and ``soodar2`'s certificate has `n2.local.net` as `SAN`:

```
soodar1(config)# crypto ikev2 proposal PROPOSAL
soodar1(config-ikev2-proposal)# integrity sha-384
soodar1(config-ikev2-proposal)# encryption aes
soodar1(config-ikev2-proposal)# group 28
soodar1(config)# crypto ikev2 profile profile-ike
soodar1(config-ikev2-profile)# identity local fqdn n1.local.net
soodar1(config-ikev2-profile)# lifetime 2400
soodar1(config-ikev2-profile)# match identity remote fqdn n2.local.net
soodar1(config-ikev2-profile)# authentication local rsa-sig
soodar1(config-ikev2-profile)# authentication remote rsa-sig
soodar1(config-ikev2-profile)# proposal PROPOSAL
soodar1(config)# crypto ipsec profile ipsec-transport-profile
soodar1(ipsec-profile)#  set transform-set ipsec-tunnel-TS
soodar1(ipsec-profile)#  set ikev2 profile profile-ike
```

and in the other router:

```
soodar2(config)# crypto ikev2 proposal PROPOSAL
soodar2(config-ikev2-proposal)# integrity sha-384
soodar2(config-ikev2-proposal)# encryption aes
soodar2(config-ikev2-proposal)# group 28
soodar2(config)# crypto ikev2 profile profile-ike
soodar2(config-ikev2-profile)# identity local fqdn n2.local.net
soodar2(config-ikev2-profile)# lifetime 2400
soodar2(config-ikev2-profile)# match identity remote fqdn n1.local.net
soodar2(config-ikev2-profile)# authentication local rsa-sig
soodar2(config-ikev2-profile)# authentication remote rsa-sig
soodar2(config-ikev2-profile)# proposal PROPOSAL
soodar2(config)# crypto ipsec profile ipsec-transport-profile
soodar2(ipsec-profile)#  set transform-set ipsec-tunnel-TS
soodar2(ipsec-profile)#  set ikev2 profile profile-ike
```

Troubleshooting

To track ipsec/ikev2 profiles state and their initiation state, users can view logs(for initiation state/errors) and SA details.

Logging

Debugging logs can be set in case of need.

debug ipsec event
: log data plane installation processes and results

debug ipsec vici json
: log all incoming VICI messages as json

debug ipsec vici detail
: log all incoming VICI messages as json and raw

To view IPSec detailed logs, use *show log ipsec* command to view StrongSwan logs.

Show commands

To view current state of ipsec SAs or IKEv2 SAs the following commands are provided in the router:

show crypto ikev2 sa [detailed] [json]
: Shows installed IKEv2 SAs details

```
soodar# show crypto ikev2 sa

Profile ike-n1-n2
  Status:           ESTABLISHED
  Local:            200.1.2.1
  Remote:           200.1.2.2/500
  Encr:             AES_CBC-128
  Hash:             HMAC_SHA2_384_192
  DH Grp:           ECP_256_BP
  Life/Active Time: 240/190 sec
```

```
soodar# show crypto ikev2 sa detailed

Profile ike-n1-n2
  Status:           ESTABLISHED
  Local:            200.1.2.1
  Remote:           200.1.2.2/500
  Encr:             AES_CBC-128
  Hash:             HMAC_SHA2_384_192
  DH Grp:           ECP_256_BP
  Life/Active Time: 240/195 sec
  Local ID:         n1.local.net
  Remote ID:        n2.local.net
  Local SPI:        8b545f20ca649813
  Remote SPI:       8b545f20ca649813
  Configured DPD:   10 sec
  Rekey in:         33 sec
```

```
soodar# show crypto ikev2 sa json
[
    {
        "name": "ike-n1-n2",
        "id": "3",
        "state": "ESTABLISHED",
        "dpd": 10,
        "lifetime": 240,
        "local-host": "200.1.2.1",
        "local-port": "500",
        "local-id": "n1.local.net",
        "remote-host": "200.1.2.2",
        "remote-port": "500",
        "remote-id": "n2.local.net",
        "initiator": false,
        "initiator-spi": "39afd77a4c51edc0",
        "responder-spi": "39afd77a4c51edc0",
        "established": "218",
        "rekey-time": "21",
        "encr-alg": "AES_CBC-128",
        "integ-alg": "HMAC_SHA2_384_192",
        "prf-alg": "PRF_HMAC_SHA2_384",
        "dh-group": "ECP_256_BP",
        "child-sas": [
            {
                "name": "ipsec-n1-n2",
                "id": "13",
                "state": "INSTALLED",
                "mode": "TUNNEL",
                "protocol": "ESP",
                "spi-in": "ccb6c76d",
                "spi-out": "c6ccd9e1",
                "encr-alg": "AES_CBC-128",
                "integ-alg": "HMAC_SHA2_256_128",
                "bytes-in": "0",
```

(continues on next page)

(continued from previous page)

```
                "packets-in": "0",
                "bytes-out": "0",
                "packets-out": "0",
                "rekey-time": "32",
                "life-time": "45",
                "install-time": "21"
            }
        ]
    }
]
```

show crypto ipsec sa [detailed] [json]

Shows installed IPSec SAs details

```
soodar# show crypto ipsec sa

Profile ipsec-n1-n2
  Status:            INSTALLED
  IKEv2 Profile:     ike-n1-n2
  Mode:              TUNNEL
  Protocol:          ESP
  Encr:              AES_CBC-128
  Hash:              HMAC_SHA2_256_128
  Life/Active Time:  66/32 sec
```

```
soodar# show crypto ipsec sa detailed

Profile ipsec-n1-n2
  Status:            INSTALLED
  IKEv2 Profile:     ike-n1-n2
  Mode:              TUNNEL
  Protocol:          ESP
  Encr:              AES_CBC-128
  Hash:              HMAC_SHA2_256_128
  Life/Active Time:  66/32 sec
  Bytes Decrypted:   0
  Packets Decrypted: 0
  Bytes Encrypted:   0
  Packets Encrypted: 0
  Inbound SPI:       c322afbc
  Outbound SPI:      c9211ed0
  Rekey in:          25 sec
```

```
soodar# show crypto ipsec sa json
[
    {
        "name": "ike-n1-n2",
        "id": "4",
        "state": "ESTABLISHED",
        "dpd": 10,
        "lifetime": 240,
```

(continues on next page)

(continued from previous page)


```
        "local-host": "200.1.2.1",
        "local-port": "500",
        "local-id": "n1.local.net",
        "remote-host": "200.1.2.2",
        "remote-port": "500",
        "remote-id": "n2.local.net",
        "initiator": false,
        "initiator-spi": "9fc4c36e3ecc04ca",
        "responder-spi": "9fc4c36e3ecc04ca",
        "established": "133",
        "rekey-time": "85",
        "encr-alg": "AES_CBC-128",
        "integ-alg": "HMAC_SHA2_384_192",
        "prf-alg": "PRF_HMAC_SHA2_384",
        "dh-group": "ECP_256_BP",
        "child-sas": [
            {
                "name": "ipsec-n1-n2",
                "id": "16",
                "state": "INSTALLED",
                "mode": "TUNNEL",
                "protocol": "ESP",
                "spi-in": "c93bfde8",
                "spi-out": "caaea224",
                "encr-alg": "AES_CBC-128",
                "integ-alg": "HMAC_SHA2_256_128",
                "bytes-in": "0",
                "packets-in": "0",
                "bytes-out": "0",
                "packets-out": "0",
                "rekey-time": "48",
                "life-time": "56",
                "install-time": "10"
            }
        ]
    }
]
```


1.14 L2 Features

1.14.1 L2 Abilities

ARP Table

ARP stands for Address Resolution Protocol, which is a protocol used to map a network layer address (such as an IP address) to a physical address (such as a MAC address) in a local network. It is necessary because data communication on a local network requires the physical address of a destination device to transmit data frames.

When a device needs to send data to another device on the same local network, it checks its ARP table (also called ARP cache) to see if it already has the MAC address of the destination device. If the MAC address is not found in the ARP table, the device sends an ARP request to ask for the MAC address of the destination device. The destination device

responds to the ARP request with its MAC address, and the requesting device adds the MAC address to its ARP table for future use.

The ARP table is a table that stores the mapping between IP addresses and MAC addresses that have been learned by the device through ARP requests and responses. It is a crucial component of a device's network stack as it enables efficient and accurate communication with other devices on the same local network.

Each ARP entry in an ARP table has four fields: `L3 Address``( IP Address), ``L2 Address``( MAC Address), ``Interface`, and `State`.

Entry `States` are:

- Permanent: This entry never expires and doesn't need verification.
- Noarp: Normally expires but doesn't need verification.
- Reachable: Verified and will normally expire.
- Stale: Still usable but needs verification.
- Delay: ARP request is scheduled.
- Probe: ARP request is sending.
- Incomplete: First ARP request sent.

`show ip arp [IFNAME]`

The command is used to display the Address Resolution Protocol (ARP) cache or ARP table.

- `IFNAME`: (Optional) specifies the name of the interface for which the ARP table is to be displayed. If this parameter is omitted, the command displays the ARP table for all interfaces.

Example:

In this example, the command displays the ARP table for all interfaces. The table shows that the IP address 200.1.2.2 has been learned and is associated with the MAC address 02:fe:6e:7f:c7:75 through interface ge0. The state column indicates that this entry is verified and reachable.

`clear ip arp [IFNAME A.B.C.D]`

The command is used to clear the ARP cache on the device.

- `interface`: (Optional) Specifies the name of the interface. This option clears the ARP cache entries for the specified interface. If this option is not specified, the command clears the ARP cache for all interfaces.
- `address`: (Optional) Specifies the IP address of the ARP entry to be cleared. If this option is not specified, the command clears all the entries in the ARP cache.

For example, to clear the ARP cache for interface ge0, the command would be:

```
soodar# clear ip arp ge0
```

To clear a specific ARP entry with IP address 192.168.1.1 on interface ge0, the command would be:

```
soodar# clear ip arp ge0 192.168.1.1
```

VLAN

VLAN stands for Virtual Local Area Network, and it is a technology used in computer networking that allows the creation of logical network segments within a single physical network. A VLAN essentially enables a network administrator to logically divide a single switch into multiple, independent switches.

VLANs have several benefits, including improved network performance and security. They can be used to reduce network congestion by isolating traffic to specific VLANs, improving overall network performance. VLANs can also help improve network security by enabling network administrators to implement different security policies for each VLAN, preventing unauthorized access to specific network resources.

Subinterfaces

A subinterface is a logical interface that is created on a physical interface of the device. It allows the physical interface to be divided into multiple logical interfaces, each with its unique network address and VLAN membership.

Once the subinterface is created, you can configure it with its IP address, subnet mask, and VLAN membership using the same commands as you would use for a regular interface. Additionally, you can configure access control lists (ACLs), and other network settings specific to the subinterface.

`interface IFNAME.(0-4095)`

Creates a subinterface.

- `IFNAME`: Refers to the name of the physical interface on which the subinterface is being created.
- `(0-4095)`: Subinterface identifier.

Note: Subinterface identifier and VLAN identifier could be different. Although it is recommended that both be the same.

Example :

```
soodar(config)# interface ge1.100
```

`encapsulation dot1q (1-4094) [exact] [second-dot1q (1-4094)]`

This command is used to configure 802.1Q VLAN tagging on a subinterface. The options and their meanings are:

- `(1-4094)`: specifies the VLAN ID that is being configured. The VLAN ID can range from 1 to 4094.
- `exact`: (Optional) Specifies that input packets must have the same number of VLAN tags as the configuration.
- `second-dot1q (1-4094)`: (optional) specifies a second VLAN ID to be used for the inner tag in a QinQ configuration. This is only used when the interface is configured for QinQ encapsulation.

Note: Although we can add two tags, it's a good practice to use dot1ad encapsulation for this purpose since dot1q was designed for one tag, and adding two tags, heavily depends on router implementation and its configuration.

Note: A subinterface before this command is not ready to use and can't be added to bridges.

---

encapsulation dot1ad (1-4094) dot1q (1-4094)

This command is used in devices to configure a double VLAN tagging protocol, also known as QinQ. QinQ is used to allow service providers to extend VLANs across their network while maintaining the customers' VLAN IDs.

The command has the following parameters:

- `dot1ad (1-4094)`: This specifies the outer VLAN ID and is also known as the Service VLAN ID (S-VID). The range of valid VLAN IDs is from 1 to 4094.
- `dot1q (1-4094)`: This specifies the inner VLAN ID and is also known as the Customer VLAN ID (C-VID). The range of valid VLAN IDs is from 1 to 4094.

This command can be used in interfaces that support QinQ. When an interface is configured with this command, it will add two tags to the Ethernet frame, with the outer tag representing the service provider's VLAN ID and the inner tag representing the customer's VLAN ID.

encapsulation default

All packets with VLAN IDs not matched to other subinterfaces are sent to this subinterface

Example :

```
soodar(config)# interface ge1.100
soodar(config-if)# encapsulation dot1q 100
soodar(config-if)# ip address 200.1.2.20/24
soodar(config-if)# interface ge1.200
soodar(config-if)# encapsulation default
```

Tag rewrite

An interface can be set up to add or remove(and in the future, translate) VLAN tags.

rewrite tag push <1|2> <dot1q|dot1ad> (0-4095) [(0-4095)]

Push 1 or 2 tags to ingress traffic. The `no` form negates all changes caused by this command.

Note: if `dot1ad` is used, only the first tag is dot1ad and the second tag is dot1q

---

rewrite tag pop <1|2>

Pop 1 or 2 tags from ingress traffic. The `no` form negates all changes caused by this command.

Example :

```
soodar(config)# int ge2
soodar(config-if)# rewrite tag push 1 dot1q 300
```

Bridge

A bridge is a networking device that connects multiple network segments together and forwards traffic between them. Bridges operate at the Data Link Layer (Layer 2) of the OSI model and are used to divide large networks into smaller, more manageable subnetworks.

Add an interface to a bridge

bridge-group (1-65535) [split-horizon group (0-255)]

Add an interface to a bridge-group. The bridge-group is identified by a number. When a new packet arrives at the interface, if the destination MAC address is not available in the bridge-group MAC table, the packet is flooded to all interfaces in the same bridge-group(except the one from which it was received and the ones who share the same split-horizon group with interface).

A split-horizon group of 0 means the interface is not in any split-horizon group and is the default value for SHG.

- `(1-65535)`: specifies the bridge group number to which the interface is assigned. The range is from 1 to 65535.
- `(0-255)`: (optional) enables the split horizon feature for the specified bridge group number. Split horizon is a technique used in bridged networks to prevent loops by not advertising routes back out the interface on which they were learned. The range is from 0 to 255.

Example : .. code-block:: frr

n1(config-if)# bridge-group 600 split-horizon group 2

Debugging bridge

show bridge (1-65535)

Example :

```
n1(config)# do sh bridge 600
 |-------------------------------------------------|
 | Domain | Interface   | Split-Horizon Group | BVI |
 |--------+-------------+---------------------+-----|
 | 600    | loopback4   | 0                   | *   |
 |--------+-------------+---------------------+-----|
 |        | ge1         | 0                   | -   |
 |-------------------------------------------------|
```

Monitoring traffic

To monitor incoming/outgoing traffics of the device interfaces, SoodarOS provides 2 tools:

- SPAN
- Trace

SPAN

SPAN (Switched Port Analyzer) stands for Switched Port Analyzer, which is a feature that allows network administrators to monitor network traffic passing through the device. SPAN enables the administrator to copy or mirror network traffic from one or more source ports to a destination port. This can be useful for network troubleshooting, security analysis, and performance monitoring. With SPAN, administrators can capture and analyze traffic without disrupting the normal operation of the network.

SPAN is also known as port mirroring or port monitoring.

Configuration

Each monitor session can have multiple source interfaces. But only one destination interface.

`monitor session (1-66) source interface INTERFACE [both|rx|tx]`

The command is used to specify the source of the SPAN session, which can be an interface or VLAN

- `(1-66)`: is the number assigned to the SPAN session. Valid values are from 1 to 66.
- `INTERFACE` is the interface or VLAN to be monitored. This can be a physical or logical interface.
- `both`: specifies that both ingress and egress traffic on the interface will be monitored.
- `rx`: specifies that only ingress traffic on the interface will be monitored.
- `tx`: rx specifies that only egress traffic on the interface will be monitored.

Note: A session is not established unless a valid destination is available.

Note: By default both ingress and egress traffic will be monitored.

`monitor session (1-66) destination interface INTERFACE`

The `destination interface` option is used to specify the interface where the copied traffic will be sent to for analysis. This command can be used in conjunction with the `source interface` option to define the source ports for the SPAN session.

- `(1-66)`: Specifies the session number for the SPAN session. This number can range from 1 to 66.
- `INTERFACE`: Specifies the interface that will be used as the destination for the copied traffic. This can be a physical interface or a VLAN interface.

Example :

```
soodar(config)# monitor session 12 source interface ge0
soodar(config)# monitor session 12 destination interface ge3
soodar(config)# interface ge3
soodar(config-if)# no shutdown

The commands configure a :abbr:`SPAN (Switched Port Analyzer)` session with session␣
↪number 12. The SPAN session copies traffic from the source interface ge0 and␣
↪sends it to the destination interface ge3.

The last command no shutdown enables the interface ge3.
```

Note: Since the exact packet is mirrored on port(without changing anything), the interface on the receiver side must be in promiscuous mode

Trace

A capturing tool to save traffics as pcap and export them for further analyzes.

`monitor capture start`

Start capturing traffic. User can provide the interface to be captured(default all interfaces traffics are captured), the direction of traffic to be captured(in, out or both) and limit the number of captured packets(default is 100).

`monitor capture stop`

Stop the current capturing process and save captured pcap.

Note: Currently only one capture is stored in memory. So starting a new capture will overwrite the previous one.

`monitor capture export scp:`

Export the stored capture pcap to a remote server.

Note: *scp* URI is: *scp:[user]:[password]@[host]:[address]*

`monitor dispatch-trace start`

Start capturing traffic with Dataplane debugging info(viewable by wireshark). The only parameters settable by admin is the limit of captured packets

Note: Each incoming/outgoing packet could produce 2 or more dispatch-trace packets. This is because each step is individually stored in pcap.

Warning: Generated pcaps in this way could become very huge in size(hundereds of megabytes and more).

Warning: Using this trace could cause performance degradation(and in rare cases instability of system).

`monitor dispatch-trace stop`

Stop current dispatch tracing process and save resulst as a pcap file.

`monitor dispatch-trace export scp:`

Export the stored dispatch trace pcap to a remote server.

Logging

Debugging logs can be set in case of need.

debug vlan event

log data plane installation processes and results

debug bridge event

log data plane installation processes and results

debug span event

log data plane installation processes and results

1.14.2 LACP

LACP (Link Aggregation Control Protocol) stands for Link Aggregation Control Protocol. It is a protocol used in computer networking to bundle multiple physical links into a single logical link to increase bandwidth and provide redundancy. LACP is used in conjunction with the Link Aggregation Group (LAG) to automatically detect and configure the bundled links between switches, routers, or servers. LACP allows the end devices to exchange information about the physical links and decide which ones should be bundled together to form the logical link.

LACP provides several benefits, including:

- Increased bandwidth: By bundling multiple physical links, the logical link can provide higher bandwidth than a single link.
- Redundancy: In the event that one of the physical links fails, traffic can be automatically rerouted through the remaining links.
- Load balancing: LACP can distribute traffic across the bundled links to optimize network performance.

LACP is defined in the IEEE 802.3ad standard and is supported by many networking devices, Soodar.

Bundle interface

A bundle interface refers to a logical interface that combines multiple physical interfaces into a single logical interface for increased bandwidth and redundancy. This is also known as link aggregation or port-channeling.

When multiple physical interfaces are combined into a bundle interface, they appear as a single logical interface to the network. This logical interface can then be configured with an IP address and other network settings, just like a physical interface.

interface bundle-ether (1-65535)

creates a bundle interface that can be used to bundle multiple physical interfaces into a single logical interface.

- `(1-65535)`: is the number of the bundle interface, from 1 to 65535.

set mode <rr|xor| active-backup |broadcast|lacp> <l2|l23|l34>

Set bundle-ether interface action mode. The first input determines the bonding algorithm, and the second is the load-balancing algorithm.

Available bonding modes are:

- `rr`: Round-robin mode, in which frames are distributed across the links in a sequential manner.
- `xor`: Exclusive-OR mode, which is similar to round-robin mode, but uses a load balancing algorithm to distribute frames based on.

- `active-backup`: Active-backup mode, in which one link is designated as active and all other links are in standby mode. If the active link fails, one of the standby links is activated.
- `broadcast`: Broadcast mode, in which all frames are forwarded over all links in the bundle.
- `lacp`: Link Aggregation Control Protocol (LACP) mode, which uses the LACP protocol to dynamically negotiate link aggregation with the remote device.

Load balancing can be based on different parts of a packet. The algorithms are:

- `l2`: Translates packets to flows by their source and destination MAC addresses.
- `123`: Translates packets to flows by source and destination MAC addresses and source and destination IP addresses.
- `l34`: Translates packets to flows by their source and destination IP addresses, protocol, and if available, TCP/UDP source and destination port number.

Note: Load balancing is available on LACP and XOR bonding algorithms only.

Note: Default bonding algorithm is LACP and default LB algorithm is l34.

Enslave an interface

`bundle id (1-65535)`

Enslave an interface to the bundle interface with the given ID. The `no` form detaches an interface from the bundle.

Note: Slave interface should be up.

Note: Slave interfaces should not have any valid IPs any subinterface and should not be in a bridge group.

Note: Subinterfaces should be created on `bundle-ether` interfaces.

Example :

```
soodar(config)# interface ge0
soodar(config-if)# bridge-group 100
soodar(config-if)# quit
soodar(config)# interface ge1
soodar(config-if)# ip address 10.0.0.1/16
soodar(config-if)# quit
soodar(config)# interface bundle-ether 55
soodar(config-if)# set mode lacp l23
soodar(config-if)# ip address 192.168.1.22/24
soodar(config-if)# quit
soodar(config)# interface ge0
soodar(config-if)# no shutdown
soodar(config-if)# no bridge-group 100
```

(continues on next page)

(continued from previous page)

```
soodar(config-if)# bundle id 55
soodar(config-if)# quit
soodar(config)# interface ge0
soodar(config-if)# no ip address 10.0.0.1/16
soodar(config-if)# bundle id 55
```

Logging

Debugging logs can be set in case of need.

debug bond event

log data plane installation processes and results

1.14.3 Port Security

Port security is a feature that provides Layer 2 security by allowing the switch to restrict input to an interface by limiting and identifying MAC addresses of the devices that are allowed to connect to the interface. Port security helps prevent unauthorized access to a network by controlling the number of allowed devices and limiting their location to specific switch ports. This feature can also be configured to dynamically learn and store MAC addresses and limit the number of MAC addresses per port.

Commands

port-security mac-address sticky X:X:X:X:X:X

The command is used to configure port security with sticky MAC addresses. When this command is executed, the router dynamically learns the MAC address and adds it to the running configuration as a sticky MAC address. This ensures that only the specific devices with the sticky MAC addresses are allowed to connect to the port. Sticky MAC addresses are saved in the configuration file and retained across router reboots. When a device is connected to a switch port with port security enabled, the MAC address of that device is dynamically learned and stored in the secure address table(if the maximum address configuration is met). Subsequent traffic from that MAC address is allowed on the port, while traffic from any other MAC address is blocked.

- `X:X:X:X:X:X`: is the MAC address that you want to configure as a sticky MAC address. It should be in the format of six groups of two hexadecimal digits separated by colons.

port-security maximum (1-100)

The command is used to set the maximum number of learned secure MAC addresses allowed on a port. When this limit is reached, the switch will restrict the port.

- `(1-100)`: is the maximum number of secure MAC addresses allowed on the port. The value can range from 1 to 100.

show port-security address [IFNAME]

The command can be used to view the learned MAC addresses, and the total secure MAC addresses allowed on the interface.

Note: this command only displays the secure MAC addresses and does not display any other MAC addresses learned by the switch. To see all MAC addresses learned by the device, use the *show ip arp* command.

`show port-security interface [IFNAME]`

The command displays the port security settings and status for a specific interface. It provides information about the number of secure MAC addresses that have been configured on the interface and the current secure MAC addresses.

- `IFNAME`: is the name of the interface that you want to view the port security configuration and status for.

1.15 PPPoE

1.15.1 Point-to-Point Protocol Over Ethernet

PPPoE (Point-to-Point Protocol over Ethernet) is a network protocol commonly used to establish a direct connection between a client device and an internet service provider (ISP) using Ethernet as the underlying medium. It enables users to connect to the internet through a broadband modem or DSL modem, providing a secure and efficient way to transmit data over the ISP's network.

In a PPPoE setup, the client device initiates a PPP session with the ISP's network by encapsulating PPP frames within Ethernet frames. This encapsulation allows the PPP traffic to be transmitted over the ISP's Ethernet-based network infrastructure, effectively creating a point-to-point connection.

The PPPoE process involves two main components:

1. PPPoE Client: This resides on the user's device, such as a computer or router, and is responsible for initiating and managing the PPP session. When the user wants to connect to the internet, the PPPoE client sends a PPPoE discovery packet to locate and communicate with the ISP's PPPoE server.
2. PPPoE Server: This resides on the ISP's network and handles incoming PPPoE connections. The server receives the PPPoE discovery packet, authenticates the client, and assigns a unique session ID (also known as a session identifier or SID) to establish a dedicated link for that particular connection.

The PPPoE connection process typically involves three stages:

a. Discovery: The PPPoE client sends out a discovery packet to locate the ISP's PPPoE server. This packet contains a special Ethernet frame with a PPPoE Active Discovery Initiation (PADI) message. The server responds with a PPPoE Active Discovery Offer (PADO) message, providing connection parameters and session options.

b. Session Initialization: After receiving the PADO message, the client selects the appropriate session options and sends a PPPoE Active Discovery Request (PADR) message back to the server. Upon successful validation of the client's credentials, the server sends a PPPoE Active Discovery Session-Confirmation (PADS) message, establishing the PPP session.

c. Data Transfer: Once the PPP session is established, data can be transmitted bidirectionally between the client and the server. This data encapsulation ensures that PPP frames are encapsulated within Ethernet frames, allowing them to traverse the ISP's Ethernet network.

In modern days, PPPoE continues to be used in certain scenarios for specific reasons:

- Authentication and Security: PPPoE provides a reliable authentication mechanism through protocols like PAP, CHAP, or MS-CHAP. This ensures that only authorized users with valid credentials can access the internet service. As security is a significant concern in modern networking, PPPoE's authentication capabilities are still relevant and valuable.
- ISP Resource Management: PPPoE aids ISPs in efficiently managing their network resources. It allows ISPs to control and allocate bandwidth, enforce quality of service (QoS) policies, and monitor individual user connections. This level of control helps optimize network performance and ensure fair usage among subscribers.

- NAT (Network Address Translation) Compatibility: PPPoE is compatible with NAT, which is commonly used to address the limited availability of IPv4 addresses. ISPs often employ PPPoE in combination with NAT, allowing multiple devices within a customer's network to share a single public IP address.

Note: Currently Soodar devices could only be PPPoE clients.

Configuring PPPoE

To create a PPPoE connection, first, we need to create a dialer interface. A dialer interface is a logical interface that stores PPPoE configurations.

interface dialer (1-255)

The command is used to configure a dialer interface on a device. Dialer interfaces are commonly used in Point-to-Point Protocol (PPP) and PPP over Ethernet (PPPoE) configurations.

- (1-255): Specifies the dialer interface instance. Valid values are integers from 1 to 255.

encapsulation ppp

The command is used to configure the Point-to-Point Protocol (PPP) encapsulation on a dialer interface. The "encapsulation ppp" command enables the PPP encapsulation on the specified dialer interface, allowing it to transmit PPP frames over the link.

Note: This is the default encapsulation for dialer interfaces.

dialer pool (1-255)

The command is used to associate a virtual dialer interface with a dialer pool. Dialer pools are used in Point-to-Point Protocol (PPP) configurations to manage multiple physical interfaces, such as ethernet interfaces, that connect to remote networks or Internet Service Providers (ISPs). The "dialer pool" command allows the virtual dialer interface to utilize one of the available physical interfaces in the pool for establishing the PPP connection.

- (1-255): Specifies the number of the dialer pool to associate with the virtual dialer interface. Valid values are integers from 1 to 255.

Note: Currently, only one physical interface could be in a dialer pool.

pppoe-client dial-pool-number (1-255)

The command is used to configure a PPPoE client session on a network interface and associate it with a specific dialer pool number. The "pppoe-client dial-pool-number" command enables the PPPoE client functionality and links it to a particular dialer pool, which contains physical interfaces used for PPPoE sessions.

- (1-255): Specifies the number of the dialer pool to associate with the PPPoE client. Valid values are integers from 1 to 255.

Below is an example of configuring the PPPoE client on an Ethernet interface and associating it with a dialer pool:

```
soodar(config)# interface ge0
soodar(config-if)# pppoe-client dial-pool-number 1
soodar(config)# interface dialer 1
soodar(config-if)# dialer pool 1
```

In this example, **ge0** is the Ethernet interface where the PPPoE client is configured. The **pppoe-client dial-pool-number 1** command associates this PPPoE client with the dialer pool number **1**. This dialer pool is used by the virtual dialer interface **dialer1** for establishing PPP sessions. The PPP packets are passed to the **ge0** interface and there, they are transmitted over Ethernet(PPPoE).

ppp pap sent-username USER password PASS

The command is used to configure the Password Authentication Protocol (PAP) credentials for Point-to-Point Protocol (PPP) authentication. PAP is a simple authentication method used to verify the identity of the PPP client when establishing a PPP connection. The command specifies the username and password that the local router (as a PPP client) sends to the remote router during the authentication process.

- `USER`: Specifies the username to be sent during PAP authentication.
- `PASS`: Specifies the password to be sent during PAP authentication.

ppp chap hostname HOSTNAME

The command is used to configure the hostname sent during the Challenge Handshake Authentication Protocol (CHAP) process in Point-to-Point Protocol (PPP) authentication. CHAP is a more secure authentication method compared to Password Authentication Protocol (PAP) and is commonly used to verify the identity of the PPP client when establishing a PPP connection. The "ppp chap hostname" command specifies the hostname used by the local router (as a PPP client) when responding to CHAP challenges from the remote router.

- `HOSTNAME`: Specifies the hostname to be used during CHAP authentication.

ppp chap password PASSWORD

The command is used to configure the shared secret password for Challenge Handshake Authentication Protocol (CHAP) in Point-to-Point Protocol (PPP) authentication. Both the local router (as a PPP client) and the remote router (as the authentication server) possess the same password. The "ppp chap password" command specifies the password to be used by the local router during the CHAP authentication process.

- `PASSWORD`: Specifies the shared secret password to be used during CHAP authentication.

ppp timeout idle (30-15552000)

The command is used to configure the idle timeout value for Point-to-Point Protocol (PPP) sessions on a network interface. The idle timeout defines the maximum duration of inactivity on the PPP link before it is automatically disconnected by the router. When no data is transmitted or received over the PPP link for the specified duration, the router terminates the PPP session to conserve network resources.

- `(30-15552000)`: Specifies the idle timeout duration in seconds. Valid values range from 30 to 15,552,000 seconds (approximately 6 months).

Note: timeout value is rounded to be a multiple of 10.

Note: Usage of # in PAP or CHAP username and password is prohibited.

---

show pppoe session

The command is used to display information about the active Point-to-Point Protocol over Ethernet (PPPoE) sessions on the router. The "show pppoe session" command provides a summary of active PPPoE sessions, including the session ID, local and remote MAC addresses, and the current session status.

Example

To establish a PPPoE session over interface `ge0` using the Dialer interface `dialer1` with both PAP and CHAP authentication, and with the username "test" and password "123" follow the example configuration below:

```
soodar(config)# interface dialer1
soodar(config-if)# encapsulation ppp
soodar(config-if)# dialer pool 1
soodar(config-if)# ppp chap hostname test
soodar(config-if)# ppp chap password 123
soodar(config-if)# ppp pap sent-username test password 123
soodar(config)# interface ge0
soodar(config-if)# no ip address
soodar(config-if)# pppoe-client dial-pool-number 1
soodar(config-if)# no shutdown
```

1. dialer1 is the virtual dialer interface that will be associated with the physical interface through the Dialer Pool. We configure PPP encapsulation and associate Dialer1 with dialer pool 1.
2. We configure CHAP authentication on dialer1, specifying the hostname ("test") and the password ("123") used for CHAP authentication with the remote router.
3. We also configure PAP authentication on dialer1, specifying the username ("test") and the password ("123") that the local router will send during the PAP authentication process.
4. We configure ge0 with no IP address and assign it to the dialer pool 1 for PPPoE sessions.

With this configuration, the router is now set up to establish a PPPoE session over ge0 using dialer1 as the virtual interface. The PPPoE session will use both PAP and CHAP authentication methods with the provided username "test" and password "123" during the authentication process.

```
soodar# show pppoe session
 |-------------------------------------------------------------------------|
 | SID | Remote MAC        | Local MAC         | Interface | Port | State |
 |-----+-------------------+-------------------+-----------+------+-------|
 | 3   | 08:19:21:ff:00:00 | 02:fe:53:5e:11:4a | dialer1   | ge0  | Up    |
 |-------------------------------------------------------------------------|
```

1.16 Appendix

1.16.1 Glossary

distance-vector
: A distance-vector routing protocol in data networks determines the best route for data packets based on distance. Distance-vector routing protocols measure the distance by the number of routers a packet has to pass. Some distance-vector protocols also take into account network latency and other factors that influence traffic on a given route. To determine the best route across a network, routers on which a distance-vector protocol is implemented exchange information with one another, usually routing tables plus hop counts for destination networks and possibly other traffic information. Distance-vector routing protocols also require that a router informs its neighbours of network topology changes periodically. [distance-vector-rp]

link-state
: Link-state algorithms (also known as shortest path first algorithms) flood routing information to all nodes in the internetwork. Each router, however, sends only the portion of the routing table that describes the state of its own

links. In link-state algorithms, each router builds a picture of the entire network in its routing tables. Distance vector algorithms (also known as Bellman-Ford algorithms) call for each router to send all or some portion of its routing table, but only to its neighbors. In essence, link-state algorithms send small updates everywhere, while distance vector algorithms send larger updates only to neighboring routers. Distance vector algorithms know only about their neighbors. [link-state-rp]

Bellman-Ford
: The Bellman–Ford algorithm is an algorithm that computes shortest paths from a single source vertex to all of the other vertices in a weighted digraph. [bellman-ford]

CHAPTER

TWO

INDICES AND TABLES

- genindex
- search

BIBLIOGRAPHY

[Draft-IETF-uttaro-idr-bgp-persistence] <https://tools.ietf.org/id/draft-uttaro-idr-bgp-persistence-04.txt>

[Draft-IETF-agrewal-idr-accept-own-nexthop] <https://tools.ietf.org/id/draft-agrewal-idr-accept-own-nexthop-00.txt>

[Draft-IETF-idr-link-bandwidth] <https://tools.ietf.org/html/draft-ietf-idr-link-bandwidth>

[Draft-IETF-mohanty-bess-ebgp-dmz] <https://tools.ietf.org/html/draft-mohanty-bess-ebgp-dmz>

[bgp-route-osci-cond] McPherson, D. and Gill, V. and Walton, D., "Border Gateway Protocol (BGP) Persistent Route Oscillation Condition", IETF RFC3345

[stable-flexible-ibgp] Flavel, A. and M. Roughan, "Stable and flexible iBGP", ACM SIGCOMM 2009

[ibgp-correctness] Griffin, T. and G. Wilfong, "On the correctness of IBGP configuration", ACM SIGCOMM 2002

[distance-vector-rp] https://en.wikipedia.org/wiki/Distance-vector_routing_protocol

[link-state-rp] https://en.wikipedia.org/wiki/Link-state_routing_protocol

[bellman-ford] https://en.wikipedia.org/wiki/Bellman-Ford_algorithm

INDEX

Symbols

A

B

C

D

E

K

L

M

N

O

P

R

S

T

U

V

W

Z