
# Soodar User Manual

***Release 21.04***

**Soodar**

**Apr 20, 2021**

# CONTENTS

CHAPTER

ONE

# INTRODUCTION

## 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.

### 1.1.1 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

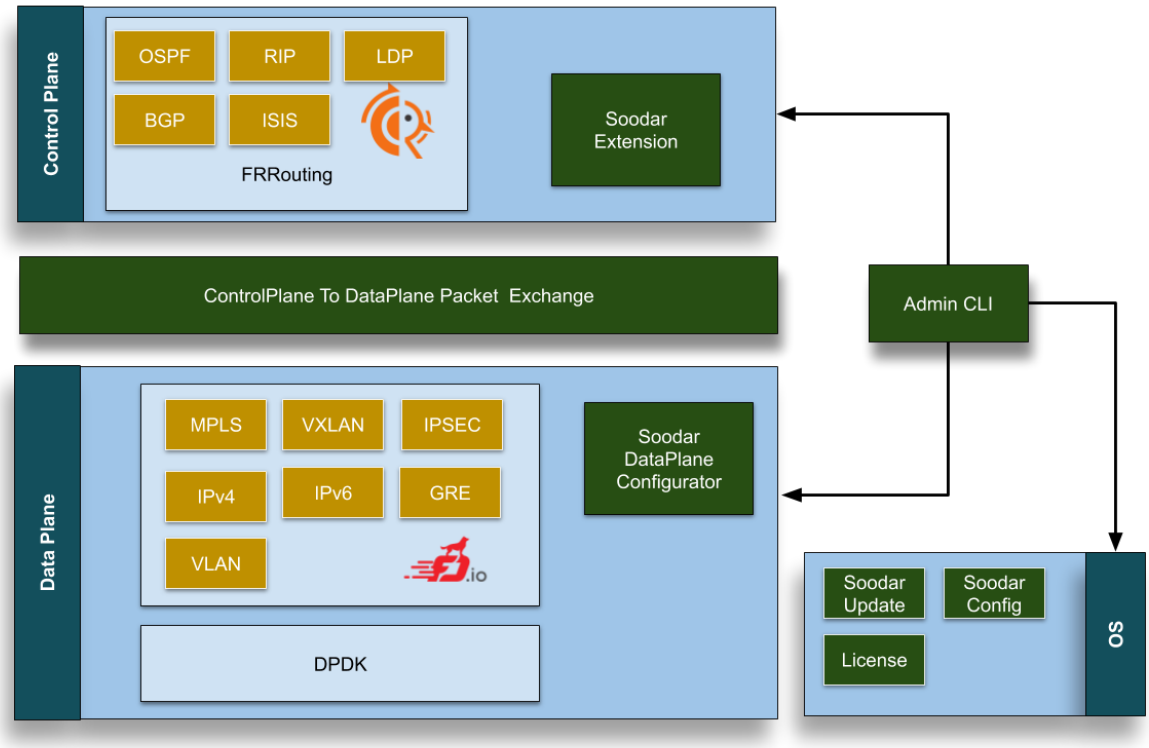
FRR is a software suite that provides TCP/IP based routing services with routing protocols support such as BGP, RIP, OSPF, IS-IS and more . FRR also supports special BGP Route Reflector and Route Server behavior. In addition to traditional IPv4 routing protocols, FRR also supports IPv6 routing protocols.

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


Control Plane
OSPF
RIP
LDP
BGP
ISIS
FRRouting
Soodar Extension
ControlPlane To DataPlane Packet Exchange
Admin CLI
Data Plane
MPLS
VXLAN
IPSEC
IPv4
IPv6
GRE
VLAN
.io
DPDK
Soodar DataPlane Configurator
Soodar Update
Soodar Config
License
OS

CHAPTER

TWO

# BASICS

## 2.1 Modes and user's configurations

### 2.1.1 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

#### Physical connection

The physical connection is the most privileged mode of connection. by *Console connection*, you don't need any password and you have all accesses. Altough, Connecting with a keyboard and monitor requires the *Access pasword*( see section *user-password-access*).

#### 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:
```

### 2.1.2 Users

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

### 2.1.3 Modes

- **View mode** Admin has access to some `show` commands to view state of router.

- **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 router.

### 2.1.4 Passwords

SoodarOS is protected by 3 levels of passwords:

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

#### Access password

It's the main password to log in with the user. A person without having the access password, can't have any access to the router( unless he is conencted via console) 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 admin issues `enable` command.

**enable password PASSWORD**
Set enable password

**no enable password PASSWORD**
Disable enable password.

#### Config password

Asked when admin input `configure` in 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 conencted 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 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.

### 2.1.5 Login Failures

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

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

### 2.1.6 Session Management

SoodarOS' admin can protect router from DoS attacks and prevent from network exhaustion by limiting the SSH authentication tries in a period of time and blocking the abuser's IP. Also he is able to see current 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 remote user

**clear line (0-530)**
Clear a TTY line and make it usable by terminate the session that is running 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 own 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 input of command.

### 2.1.7 MOTD

Sometime system administrator needs to set a message, so every user attempting to log in could 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.

### 2.1.8 SSH

Soodar serves both as a client for 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 user's authorized keys.

**no key HASH**
Remove a public key from user's authorized keys by its hash.

**no key (1-65535)**
Remove a public key from user's authorized keys by its index in keys list.

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+3DpBKYQ3of3KWNfLHRC
↪vzF3DkyanO6LnnbCYkg7SFzWE= temp@test
```

(continues on next page)

(continued from previous page)

```
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/
 ↪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 public key(s) of a server( provided by its IP or its host name) to known hosts list of current user.

**no known-host <A.B.C.D|X:X::X:X|HOST>**
Remove a server from known hosts of current user.

**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 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+EPmAH7mPoKkg2EGCpr889pR5mcBiXPVq69yUNFUG7U0D2aqDaGbaXk9TcfqCrktVmjGVF8rY91TaLMJBngVaYYsnT+
 ↪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:
```

```
soodar# show ip ssh client known-host 192.168.30.39
soodar# ssh test@192.168.30.39
The authenticity of host '192.168.30.39 (192.168.30.39)' can't be␣
 ↪established.
RSA key fingerprint is SHA256:lJ2gRSCd8Wh0CrcPU8sOlZJdrbff2QrGaJ5zBcZ2S4I.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.30.39' (RSA) to the list of known hosts.
test@192.168.30.39's password:
soodar# show ip ssh client known-host 192.168.30.39
192.168.30.39 RSA SHA256:lJ2gRSCd8Wh0CrcPU8sOlZJdrbff2QrGaJ5zBcZ2S4I
```

## 2.2 Basic Config Commands

**hostname HOSTNAME**
Set hostname of the router.

**log export removable-storage**
Export system logs to connected USB removable storage. System logs contains FRR logs and syslogs

**log export ssh HOST USER PATH**
Export system logs via SSH to HOST. System logs contains FRR logs and syslogs

**log export ssh HOST USER PATH**
Export system logs via FTP to HOST. System logs contains FRR logs and syslogs

**[no] ip host NAME A.B.C.D**
Add an entry to `know hosts`. The negation of this command cause the entry( if exists) be removed.

**[no] ip name-server A.B.C.D**
Add or remove a Name server.

**show clock**
Show current date and time

Example:

```
soodar# do 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
```

**clock timezone TIMEZONE**
Set system timezone. TIMEZONE is timezone's long name based on IANA TZDatabase

Example:

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

**service password-encryption**
Encrypt password.

## 2.3 Sample Config File

Below is a sample configuration file .

```
!
hostname soodar
enable password admin
enable config password configadmin
!
log stdout
!
!
```

! 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.

```
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.

### 2.3.1 Terminal Mode Commands

**write terminal**
Displays the current configuration to the vty interface.

**write file**
Write current configuration to configuration file.

**configure [terminal]**
Change to configuration mode. This command is the first step to configuration.

**list**
List all available commands.

**show version**
Show the current version of SoodarOS and its host information.

**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.

**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
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.

**find COMMAND...**
This command performs a simple substring 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 set router-id is:

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

**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
```

## 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.

### 2.4.1 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.

**no ip prefix-list NAME**

### 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.

**no ip prefix-list NAME description [DESC]**
Deletes the description from a prefix list. It is possible to use the command without the full description.

### ip prefix-list sequential number control

**ip prefix-list sequence-number**
With this command, the IP prefix list sequential number is displayed. This is the default behavior.

**no ip prefix-list sequence-number**
With this command, the IP prefix list sequential number is not displayed.

### Showing ip prefix-list

**show ip prefix-list**
Display all IP prefix lists.

**show ip prefix-list NAME**
Show IP prefix list can be used with a prefix list name.

**show ip prefix-list NAME seq NUM**
Show IP prefix list can be used with a prefix list name and sequential number.

**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**

**show ip prefix-list summary NAME**

**show ip prefix-list detail**

**show ip prefix-list detail NAME**

### 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.

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

**`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.

### 2.5.1 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*.

### 2.5.2 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 address IPV4_ADDR`**
This is a BGP specific match command. Matches the specified *ipv4_addr*.

**`match ipv6 next-hop IPV6_ADDR`**
This is a BGP specific match command. Matches the specified *ipv6_addr*.

**`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 source-protocol PROTOCOL_NAME**
This is a ZEBRA 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.

### 2.5.3 Route Map Set Command

**set tag TAG**
Set a tag on the matched route. This tag value can be from (1-4294967295). Additionally if you have compiled with the `--enable-realms` configure option. Tag values from (1-255) are sent to the Linux kernel as a realm value. Then route policy can be applied. See the tc man page.

**set ip next-hop IPV4_ADDRESS**
Set the BGP nexthop address to the specified IPV4_ADDRESS. For both incoming and outgoing route-maps.

**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*.

**[no] 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.

**[no] set metric <[+|-](1-4294967295)|rtt|+rtt|-rtt>**
Set the BGP attribute MED to a specific value. Use +/- to add or subtract the specified value to/from the 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 as-path prepend AS_PATH**
Set the BGP AS path to prepend.

**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.

### 2.5.4 Route Map Call Command

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

### 2.5.5 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

### 2.5.6 Route Map Optimization Command

**route-map optimization**
Enable route-map processing optimization. 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.

**no route-map optimization**
Disable the route-map processing optimization.

### 2.5.7 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.

## 2.6 System

### 2.6.1 System Logging

SoodarOS uses `systemd-journald` as main logging solution.

**[no] debug service snmp**
Enable logging for SNMP service. All SNMP logs appear in journald.

**[no] debug service mender**
Enable logging for mender update service. All mender logs appear in journald.

**[no] debug service ntpd**
Enable logging for NTP service. All NTP logs appear in journald.

**[no] debug dplane fib**
Enable data plane( VPP) FIB logs.

**[no] debug dplane ipsec**
Enable data plane( VPP) IPSec logs.

**log rotate max-file-size SIZE**
set `SIZE` as the limit of how large individual journal files may grow at most. When limit is reached, it rotates to next journal file.

**log rotate max-files (1-1000)**
control how many individual journal files to keep at most. Default is 100.

**log rotate max-use SIZE**
Control how much disk space the journal may use up at most. The `SIZE` is capped to 4G. After reaching the limit, it starts removing elder journal files.

**[no] 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.

**[no] 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 time span are deleted.

**[no] log file [LEVEL]**
If you want to enable log into a file, please enter command as in this example:

```
log file  informational
```

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) will be used. The `no` form of the command disables logging to a file.

**[no] 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) will be used. The `no` form of the command disables logging to syslog. Default log level for syslog is set to `error` level.

**[no] log syslog [A.B.C.D|HOST] tcp [tls [skip-host-verify]] [port (100-65535)]**
Define a remote host to send syslogs. make sure that `log syslog` is enabled to make this command work. Default port is `514`. User can enable TLS connection. By adding `skip-host-verify` option, remote hostname is not checked against provided certificate CN/SAN.

Example:

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

**[no] log syslog HOST loki [skip-host-verify] [port (100-65535)]**
Define a remote host to send syslogs. make sure that `log syslog` is enabled to make this command work. Default port is `3000`. By adding `skip-host-verify` option, in case of https connection remote hostname is not checked against provided certificate CN/SAN.

---

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

---

---

**Note:** Port is a separate option. User **must not** provide port in 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
```

**[no] 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) will be used. The `no` form of the command disables logging to terminal monitors.

**[no] 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.

**[no] 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.

**[no] 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 in the range 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.

**[no] log commands**
This command enables the logging of all commands typed by a user to all enabled log destinations. The note that logging includes full command lines, including passwords.

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

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

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

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

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

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

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

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

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

### 2.6.2 System update

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

#### Online update

Update system from a server. Disabled by default. When online update is enabled, the system automatically check the server for available updates and install if any is present.

#### Configuration

**system update enable**
Enable online update

**no system update enable**
Disable online update

**system update server-url WORD**
Set update server's URL

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

**system update update-poll-interval (5-2147483647)**
Check for update interval in seconds

**system update inventory-poll-interval (5-2147483647)**
Send system inventory in intervals. Unit is in 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 a 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. just reboot without commit to rollback)

**Note:** To use offline update, 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 system's booting, no removable storage should be plugged to router device or else boot will fail.

### 2.6.3 System backup and restore

The router is equipped with a set of backup/restore tools. currently only *startup config* could be backed up. The backup files could be stored in three ways:

1. To remote host and via SSH
2. To local storage
3. To removable storage

#### Backup and restore via SSH

#### Commands

**system config backup ssh HOST USER PATH**
Backup current startup config to a remote host. HOST is IP address or host name of desired destination USER is remote host user that SSH tunnel will be made to. And PATH is address to save file

Examples:

```
soodar(config)# ip host bckup-server 192.168.1.123
soodar(config)# system config backup ssh bckup-server admin ~/backups/n1-edge
soodar(config)# system config backup ssh 10.12.12.1 sysadmin /var/router-backups/
↪test_bkp
```

**system config backup list ssh HOST USER PATH**
List available backup files on remote hast. PATH is the remote destination *directory* that backups are stored

Example:

```
soodar(config)# system config backup list ssh bckup-server admin ~/backups

Tags in provided remote path:
    1   -rw-r--r-- 1 admin admin    0 Mar 17 10:33 n1-edge
    1   -rw-r--r-- 1 admin admin    0 Mar 16 19:52 n3
```

**system config restore ssh HOST USER PATH**
Example:

```
soodar(config)# system config backup restore ssh 10.12.12.1 sysadmin /var/router-
↪backups/test_bkp
```

#### Backup and restore to and from removable storage

#### Commands

**system config backup removable-storage NAME**
Backup current startup config to the removable storage. NAME is desired backup's name.

Examples:

```
soodar(config)# system config backup removable-storage before-a-big-change
```

**system config backup list removable-storage**
List available backup files on removable storage.

Example:

```
soodar(config)# system config backup list removable-storage

Tags in provided remote path:
    1   -rw-r--r-- 1 admin admin    0 Mar 17 10:51 before-a-big-change
```

**system config restore removable-storage NAME**

#### Backup and restore to and from local

#### Commands

**system config backup local NAME**
Backup current startup config to the local storage. NAME is desired backup's name.

Examples:

```
soodar(config)# system config backup local before-a-big-change
```

**system config backup list local**
List available backup files on local storage.

Example:

```
soodar(config)# system config backup list local

Tags in provided remote path:
    1   -rw-r--r-- 1 admin admin    0 Mar 17 10:51 before-a-big-change
```

**system config restore local NAME**

### 2.6.4 Prometheus Monitoring

Soodar OS supports both SNMP and Prometheus for monitoring purposes. Users can enable prometheus monitoring by running *soolog* service on router. After running and enabling *soomon* service, 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 behaviour could change in future.

---

### 2.6.5 System Services

To acheive the best performance on system, there are services running in 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 router.
- soolog: Soodar prometheus monitoring service.

**show system service status SERVICE**
Show service status based on output of systemd

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

---

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

---

### 2.6.6 System Security

To protect system from SYN flood attack, admin can set maximum TCP SYN limit.

**tcp syn-flood limit (1-4294967295)**
Set TCP SYN limit. Default limit is `256`.

## 2.7 SNMP

SNMP is a widely implemented feature for collecting network information from router and/or host. FRR itself does not support SNMP agent (server daemon) functionality but is able to connect to a SNMP agent using the the AgentX protocol (RFC 2741) and make the routing protocol MIBs available through it.

**agentx**
Start SNMP Daemon and AgentX on system

**no agentx**
Stop SNMP Daemon and AgentX on system

### 2.7.1 SNMP Users

In order 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]**
create a user named USER with authNoPriv secutiry level and as ROUser. the authentication protocol and password is provided by user. if *priv* is provided, a user named USER with authPriv security level and as ROUser is created.

---

**Note:** Passwrod 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
```

## 2.8 NTP

Network Time Protocol( NTP) is a well-known widely used protocol to synchronize the time of the device over the internet. Using *chrony*, SoodarOS can be an NTP client supporting Version 3 and Version 4 of the NTP protocol

### 2.8.1 Setting up NTP

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

**ntp server SERVER [OPTIONS]**
Specifies an NTP server that can be used as a time source. Available options are:

- **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
```

**no ntp server SERVER [OPTIONS]**
Remove an existing NTP server from list.

### 2.8.2 Setting up NTP Authentication

The NTP protocol supports a message authentication code (MAC) to prevent computers from having their system time upset by the rogue packets being sent to them. The MAC is generated as a function of a password specified in the `ntp key-authentication` list.

#### Add a New Key

Each key is made of an `id`, a `hash function` and the `key value`, so users need to provide these three to create a new key.

**ntp authentication-key (1-65535) sha1 WORD**
Add a new key to the list of authentication keys. Users can choose a key-id in `(1-65535)` range, `SHA1` as its hash function and `WORD` as the key itself.

#### Remove a key

To remove a key, knowing `key-id` is the only necessity.

**no ntp authentication-key (1-65535)**
Remove a key from keys database. The key-id should be provided.

#### Enablng and Disabling NTP Authentication

The NTP authentication mechanism only takes effetct 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 with simply disabling it.

**ntp authentication**
Enable NTP authentication mode.

**no ntp authentication**
Disable NTP authentication mode.

### 2.8.3 Showing NTP status

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

**show ntp sources**
Print current server information.

Example:

```
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
```

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

### 2.8.4 Manual clock

If you have no/restricted internet connection, you can disable NTP and set 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.

## 2.9 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.

### 2.9.1 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**
Enable IPv6 on the interface

**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).

### 2.9.2 Router Advertisement

**no ipv6 nd suppress-ra**
Send router advertisement messages.

**ipv6 nd suppress-ra**
Don't send router advertisement messages.

**ipv6 nd prefix ipv6prefix [valid-lifetime] [preferred-lifetime] [off-link] [no-autoconfig]**
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.

**[no] ipv6 nd ra-interval [(1-1800)]**
The maximum time allowed between sending unsolicited multicast router advertisements from the interface, in seconds. Default: `600`

**[no] ipv6 nd ra-interval [msec (70-1800000)]**
The maximum time allowed between sending unsolicited multicast router advertisements from the interface, in milliseconds. Default: `600000`

**[no] 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

**[no] 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`

**[no] 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`

**[no] 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`

**[no] 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`

**[no] 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

**[no] 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

**[no] 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

**[no] 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`

**[no] 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`

**[no] 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

**[no] ipv6 nd router-preference [(high|medium|low)]**
Set default router preference in IPv6 router advertisements per RFC4191. Default: medium

**[no] 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.

**[no] 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

**[no] 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

### 2.9.3 Router Advertisement Configuration Example

A small example:

```
interface ge0
 ipv6 enable
 ipv6 address 2001:1::1/64
 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)

## 2.10 IPFIX

Internet Protocol Flow Information Export (IPFIX) is an IETF protocol, as well as 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 to be formatted and transferred from an exporter to a collector.

### 2.10.1 IPFIX flow exporter

IPFIX exporter information is configured and saved as `flow exporter`

#### Commands

**flow exporter**
Enter flow exporter configuration mode

**destination A.B.C.D**
Set IPFIX flow collector IPv4 address

**source A.B.C.D**
Set IPFIX flow packets source. This address should be valid on router.

**transport udp (1-65535)**
Define destination port

### 2.10.2 IPFIX flow monitor

IPFIX flow definitions

#### Commands

**flow monitor**
Enter flow monitor configuration mode

**cache timeout active (1-604800)**
Set active flow cache timeout in seconds

**cache timeout inactive (1-604800)**
Set inactive flow cache timeout in seconds

**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**
Apply flow monitor on an interface output

**no ip flow monitor output**
Remove flow monitor on an interface output

### 2.10.3 Logging

Debugging logs can be set in case of need.

**[no] debug ipfix event**
log data plane installation processes and results

### 2.10.4 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
```

## 2.11 License

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

### 2.11.1 Default license

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

- Drop supporting ethernets faster than Gigabit ethernet.
- Support a maximum of 8 hardware interfaces.
- Limit VPLS interfaces count to 5.
- Limit VXLAN interfaces count to 4.
- Limit protected tunnels to 2 tunnels.
- Support up to 10 Access-list.
- Limit access-list entries to 10 per ACL.
- Support up to 2 Policy map.
- NAT44 IP pool is limited to 32 IPs.
- NAT44 static entries are limited to 32 entries.
- Limit VRFs to 2 VRFs( not counting default VRF).
- Limit routes per VRF to 64.

### 2.11.2 License request

To import a license, an enrollment is needed. To achieve this, SoodarOS will make a license request on user demand, and display it on screen. The displayed request, should be sent for signing.

**license generate license-request [terminal]**
Generate license request and display it on screen.

### 2.11.3 Import license

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

### 2.11.4 Check license

To verify the license, `license check` command is used. If there are errors, they are printed on screen.

**license check**
Check installed license.

### 2.11.5 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 resource is unlimited.

Example :

```
n1# show license
                      Name   Limit   Used
 ------------------------------------------
      Hardware Interfaces       8      3
 Hardware Interfaces Type       1      -
          VPLS Interfaces       5      0
         VXLAN Interfaces       4      0
        Protected Tunnels       2      1
               QoS Policy       2      0
           NAT44 Pool IPs      32      0
     NAT44 Static Entries      32      0
                      VRF       2      0
       VRF default routes      64      2
                      ACL      10      0

 BGP Support: Available
 MPLS Support: Available
 VRF Support: Available
 MP-GRE Support: Not available
 EIGRP Support: Not available
 IPv6 Support: Not available
```

CHAPTER

THREE

# PROTOCOLS

## 3.1 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 5883**

### 3.1.1 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 NA**
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.

**no peer <A.B.C.D|X:X::X:X>$peer [{multihop|local-address <A.B.C.D|X:X::X:X>$local|interface**
Stops and removes the selected peer.

**profile WORD**
Creates a peer profile that can be configured in multiple peers.

**no profile WORD**
Deletes a peer profile. Any peer using the profile will have their configurations reset to the default values.

**show bfd [vrf NAME] peers [json]**
Show all configured BFD peers information and current status.

**show bfd [vrf NAME$vrf_name] peer <WORD$label|<A.B.C.D|X:X::X:X>$peer [{multihop|local-addr**
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.

### 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-interval (10-60000)**
Configures the minimal echo receive transmission interval that this system is capable of handling.

**[no] echo-mode**
Enables or disables the echo transmission mode. This mode is disabled by default.

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).

**[no] shutdown**
Enables or disables the peer. When the peer is disabled an 'administrative down' message is sent to the remote peer.

**[no] 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`).

**[no] 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 overriden on a peer basis by specifying new 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.

**no neighbor <A.B.C.D|X:X::X:X|WORD> bfd**
Removes any notification registration for this neighbor.

**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.

**no neighbor <A.B.C.D|X:X::X:X|WORD> bfd check-control-plane-failure**
Disallow to write CBIT independence in BFD outgoing packets. Also disallow to ignore BFD down notification. This is the default behaviour.

**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.

**no neighbor <A.B.C.D|X:X::X:X|WORD> bfd profile BFDPROF**
Removes the BFD profile configuration from peer session(s).

### 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.

**no isis bfd**
Removes any notification registration for this interface peers.

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.

**no isis bfd profile BFDPROF**
Removes any BFD profile if present.

#### 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.

**no ip ospf bfd**
Removes any notification registration for this interface peers.

#### OSPF6 BFD Configuration

The following commands are available inside the interface configuration node.

**ipv6 ospf6 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.

**no ipv6 ospf6 bfd**
Removes any notification registration for this interface peers.

### 3.1.2 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
```

(continues on next page)

(continued from previous page)

```
  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
 !

 ! 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
```

### 3.1.3 Status

You can inspect the current BFD peer status with the following commands:

```
soodar# show bfd peers
BFD Peers:
        peer 192.168.0.1
```

(continues on next page)

(continued from previous page)

```
                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 transmission interval: disabled
                Remote timers:
                        Detect-multiplier: 3
                        Receive interval: 300ms
                        Transmission interval: 300ms
                        Echo transmission 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 transmission interval: disabled
                Remote timers:
                        Detect-multiplier: 3
                        Receive interval: 300ms
                        Transmission interval: 300ms
                        Echo transmission 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 transmission interval: disabled
                Remote timers:
                        Detect-multiplier: 3
                        Receive interval: 300ms
```

(continues on next page)

(continued from previous page)

```
                        Transmission interval: 300ms
                        Echo transmission 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-interval":50,"remote-receive-interval":300,"remote-transmit-
↪interval":300,"remote-echo-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
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
```

### 3.1.4 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:

**[no] debug bfd network**
: Toggle network events: show messages about socket failures and unexpected BFD messages that may not belong to registered peers.

**[no] debug bfd peer**
: Toggle peer event log messages: show messages about peer creation/removal and state changes.

**[no] debug bfd zebra**
: Toggle zebra message events: show messages about interfaces, local addresses, VRF and daemon peer registrations.

## 3.2 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.

### 3.2.1 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.

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.

### 3.2.2 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 BGP protocol uses the AS number 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*.

**no router bgp ASN**
Destroy a BGP protocol process with the specified ASN.

**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 zebra-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:**

bgp-vrf-route-leaking

**See also:**

zebra-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.

### Administrative Distance Metrics

**distance bgp (1-255) (1-255) (1-255)**
This command change distance value of BGP. The arguments are the distance values for 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

**[no] 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:
BGP router identifier 10.10.10.1, local AS number 65001 vrf-id 0
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
192.168.0.2     4      65002         8        10        0    0    0 00:03:09     ␣
↪       5 (Policy)
fe80:1::2222    4      65002         9        11        0    0    0 00:03:09     ␣
↪(Policy) (Policy)
```

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

**[no] bgp reject-as-sets**

This command enables rejection of incoming and outgoing routes having AS_SET or AS_CONFED_SET type.

### Disable checking if nexthop is connected on EBGP sessions

**[no] 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) (1-20000) (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

[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)

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:


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:


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)

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.
```

(continues on next page)

(continued from previous page)

```
<------------------------------------------------------------------------

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

<------------------------------------------------------------------------>
```

### 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 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 ifunctionality 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 ifunctionality 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.

### Administrative Shutdown

**[no] 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.

**no network A.B.C.D/M**

**[no] 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

**[no] neighbor X:X::X:X 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.

**[no] bgp default ipv4-unicast**
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
 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 annocuments.

### Route Aggregation

### Route Aggregation-IPv4 Address Family

**aggregate-address A.B.C.D/M**
This command specifies an aggregate address.

**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. Aggregated routes will not be announced.

**no aggregate-address A.B.C.D/M**
This command removes an aggregate address.

This configuration example setup the aggregate-address under 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
  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. Aggregated routes will not be announced.

**no aggregate-address X:X::X:X/M**
This command removes an aggregate address.

This configuration example setup the aggregate-address under 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

**redistribute kernel**
Redistribute kernel route to BGP process.

**redistribute static**
Redistribute static route to BGP process.

**redistribute connected**
Redistribute connected route to BGP process.

**redistribute rip**
Redistribute RIP route to BGP process.

**redistribute ospf**
Redistribute OSPF route to BGP process.

**redistribute vnc**

**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 feaure 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.

**[no] 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.

---

**[no] 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

**[no] 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.

**[no] neighbor PEER disable-connected-check**
Allow peerings between directly connected eBGP peers using loopback addresses.

**[no] 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.

**[no] neighbor PEER description ...**
Set description of the peer.

**[no] neighbor PEER version VERSION**
Set up the neighbor's BGP version. *version* can be *4*, *4+* or *4-*. BGP version *4* is the default value used for BGP peering. BGP version *4+* means that the neighbor supports Multiprotocol Extensions for BGP-4. BGP version *4-* is similar but the neighbor speaks the old Internet-Draft revision 00's Multiprotocol Extensions for BGP-4. Some routing software is still using this version.

**[no] 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.

This command is deprecated and may be removed in a future release. Its use should be avoided.

**[no] neighbor PEER next-hop-self [all]**
This command specifies an announced route's nexthop as being equivalent to the address of the bgp router if it is learned via eBGP. If the optional keyword *all* 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.

**[no] 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
```

**[no] neighbor PEER default-originate**
*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.

**neighbor PEER port PORT**

**[no] 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**

**[no] neighbor PEER weight WEIGHT**
This command specifies a default *weight* value for the neighbor's routes.

**[no] 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.

**[no] 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.

**[no] 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.

**[no] 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.

**[no] 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 occurences 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.

**[no] 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.

**[no] 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.

**[no] 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*.

**[no] 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.

**[no] bgp fast-external-failover**
This command causes bgp to not 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.

**[no] bgp default ipv4-unicast**
This command allows the user to specify that v4 peering 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.

**[no] 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.

**[no] 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.

**[no] 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.

### Displaying Information about Peers

**show bgp <afi> <safi> neighbors WORD bestpath-routes [json] [wide]**
For the given neighbor, WORD, that is specified list the routes selected by BGP as having the best path.

### 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.

**[no] 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.

### Capability Negotiation

**neighbor PEER strict-capability-match**

**no 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.

**[no] 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**

**no neighbor PEER override-capability**
Override the result of Capability Negotiation with local configuration. Ignore remote peer's capability value.

### AS Path Access Lists

AS path access list is user defined AS path.

**bgp as-path access-list WORD permit|deny LINE**
This command defines a new AS path access list.

**no bgp as-path access-list WORD**

**no bgp as-path access-list WORD permit|deny LINE**

### 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]_
```

### Using AS Path in Route Map

**[no] 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.

**[no] 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.

**[no] 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.

### 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.

**internet** `internet` represents well-known communities value 0.

**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** documents 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. It is recommended upon receiving prefixes tagged with this community to add `NO_EXPORT` and `NO_ADVERTISE`.

**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.

**no bgp community-list [standard|expanded] NAME**
Deletes the community list specified by `NAME`. 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 and 99 is standard community list. Community list number in the range from 100 to 199 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 (100-199) permit|deny COMMUNITY**
This command defines a new expanded community list. The argument to (100-199) defines the list identifier.

### 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 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
 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 70 deny
bgp community-list 80 permit 7675:80
bgp community-list 80 deny
bgp community-list 90 permit 7675:90
bgp community-list 90 deny
!
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
```

(continues on next page)

(continued from previous page)

```
 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` or `0:90`. 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
!
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 communities value keyword `internet` has special meanings in standard community lists. In the below example `internet` matches all BGP routes even if the route does not have communities attribute at all. So community list `INTERNET` is the same as `FILTER` in the previous example.

```
bgp community-list standard INTERNET deny 1:1
bgp community-list standard INTERNET permit internet
```

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
```

(continues on next page)

(continued from previous page)

```
!
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.

**no bgp extcommunity-list NAME**

**no bgp extcommunity-list standard NAME**

**no bgp extcommunity-list expanded NAME**
These commands delete extended community lists specified by *name*. All of extended community lists shares a single name space. So extended community lists can be removed simply specifying the name.

**show bgp extcommunity-list**

**show bgp extcommunity-list NAME detail**
This command displays current extcommunity-list information. When *name* is specified the community list's information is shown.:

```
# show bgp extcommunity-list
```

### BGP Extended Communities in Route Map

**match extcommunity WORD**

**set extcommunity rt EXTCOMMUNITY**
This command set Route Target value.

**set extcommunity soo EXTCOMMUNITY**
This command set 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.

**no bgp large-community-list NAME**

**no bgp large-community-list standard NAME**

**no bgp large-community-list expanded NAME**
These commands delete Large Community lists specified by *name*. All Large Community lists share a single namespace. This means Large Community lists can be removed by simply specifying the name.

**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.

### 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.

**[no] 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.

**[no] 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.

**[no] 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.

**[no] debug bgp bestpath <A.B.C.D/M|X:X::X:X/M>**
Enable or disable debugging for bestpath selection on the specified prefix.

**[no] debug bgp nht**
Enable or disable debugging of BGP nexthop tracking.

**[no] debug bgp update-groups**
Enable or disable debugging of dynamic update groups. This provides general information on group creation, deletion, join and prune events.

**[no] debug bgp zebra**
Enable or disable debugging of communications between *bgpd* and *zebra*.

### Dumping Messages and Routing Tables

**dump bgp all PATH [INTERVAL]**

**dump bgp all-et PATH [INTERVAL]**

**no dump bgp all [PATH] [INTERVAL]**
Dump all BGP packet and events to *path* file. If *interval* is set, a new file will be created for echo *interval* of seconds. The path *path* can be set with date and time formatting (strftime). The type 'all-et' enables support for Extended Timestamp Header (packet-binary-dump-format).

**dump bgp updates PATH [INTERVAL]**

**dump bgp updates-et PATH [INTERVAL]**

**no dump bgp updates [PATH] [INTERVAL]**
Dump only BGP updates messages to *path* file. If *interval* is set, a new file will be created for echo *interval* of seconds. The path *path* can be set with date and time formatting (strftime). The type 'updates-et' enables support for Extended Timestamp Header (packet-binary-dump-format).

**dump bgp routes-mrt PATH**

**dump bgp routes-mrt PATH INTERVAL**

**no dump bgp route-mrt [PATH] [INTERVAL]**
Dump whole BGP routing table to *path*. This is heavy process. The path *path* can be set with date and time formatting (strftime). If *interval* is set, a new file will be created for echo *interval* of seconds.

Note: the interval variable can also be set using hours and minutes: 04h20m00.

### 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.

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.

### 3.2.3 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 [wide]**

**show ip bgp A.B.C.D [wide]**

**show bgp [wide]**

**show bgp X:X::X:X [wide]**
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.

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 summary**
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]`.

**show bgp [afi] [safi]**

**show bgp <ipv4|ipv6> <unicast|multicast|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] summary**
Show a bgp peer summary for the specified address family, and subsequent address-family.

**show bgp [afi] [safi] summary failed [json]**
Show a bgp peer summary for peers that are not succesfully exchanging routes for the specified address family, and subsequent address-family.

**show bgp [afi] [safi] summary established [json]**
Show a bgp peer summary for peers that are succesfully exchanging routes for the specified address family, and subsequent address-family.

**show bgp [afi] [safi] neighbor [PEER]**
This command shows information on a specific BGP peer of the relevant afi and safi selected.

**show bgp [afi] [safi] dampening dampened-paths**
Display paths suppressed due to dampening of the selected afi and safi selected.

**show bgp [afi] [safi] dampening flap-statistics**
Display flap statistics of routes of the selected afi and safi selected.

**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.

### Displaying Routes by Community Attribute

The following commands allow displaying routes based on their community attribute.

**show [ip] bgp <ipv4|ipv6> community**

**show [ip] bgp <ipv4|ipv6> community COMMUNITY**

**show [ip] bgp <ipv4|ipv6> community COMMUNITY exact-match**
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**

**show [ip] bgp <ipv4|ipv6> community-list WORD exact-match**
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.

#### 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 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.

### 3.2.4 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**

**no 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**

### 3.2.5 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.

### 3.2.6 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 `(^|[,{}()]|$)`.

### 3.2.7 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
 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
```

(continues on next page)

(continued from previous page)

```
  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 asp-own-as permit ^$
bgp as-path access-list 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
! 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
```

(continues on next page)

(continued from previous page)

```
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
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
```

(continues on next page)

(continued from previous page)

```
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
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.
!
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
!
```

### 3.2.8 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 (*Full Mesh*), we can convert it into a centralized scenario where each of the four establishes a single BGP peering against the Route Server (*Route server and clients*).

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 *Announcement processing inside a 'normal' BGP speaker*, 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.

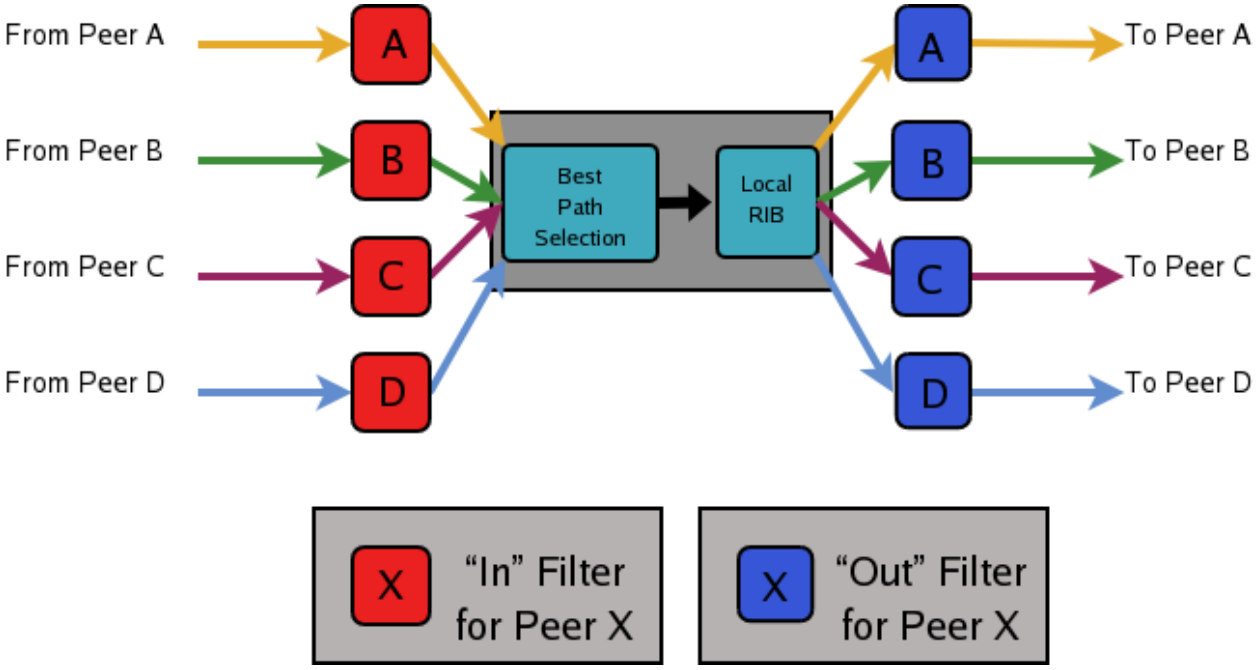



Fig. 1: Announcement processing inside a 'normal' BGP speaker

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.
  - 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.

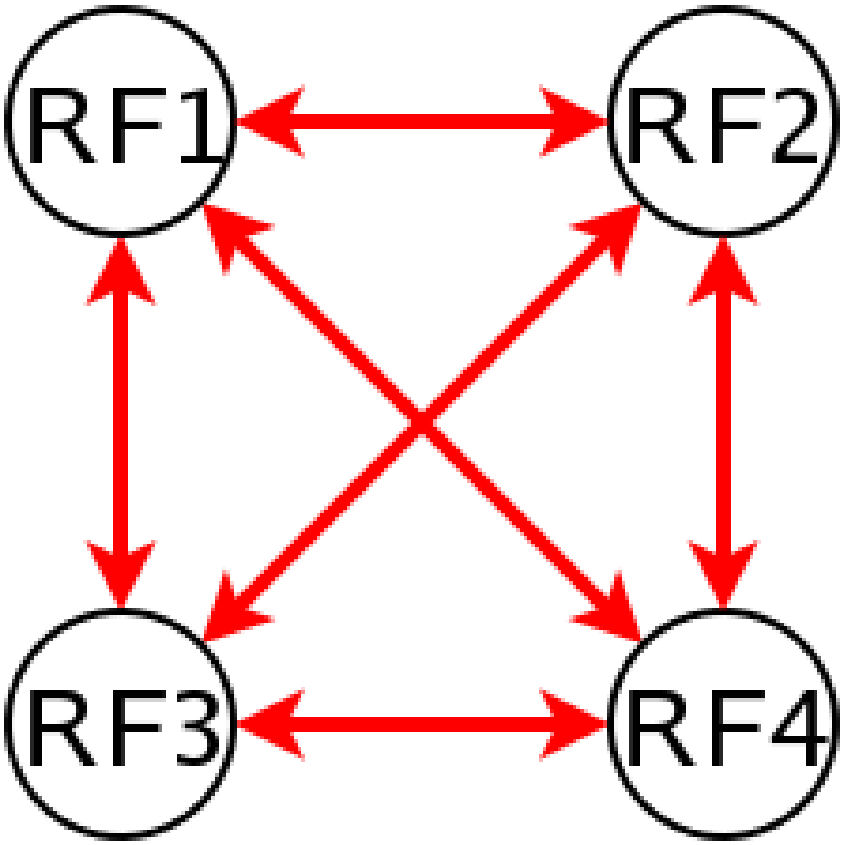



Fig. 2: Full Mesh

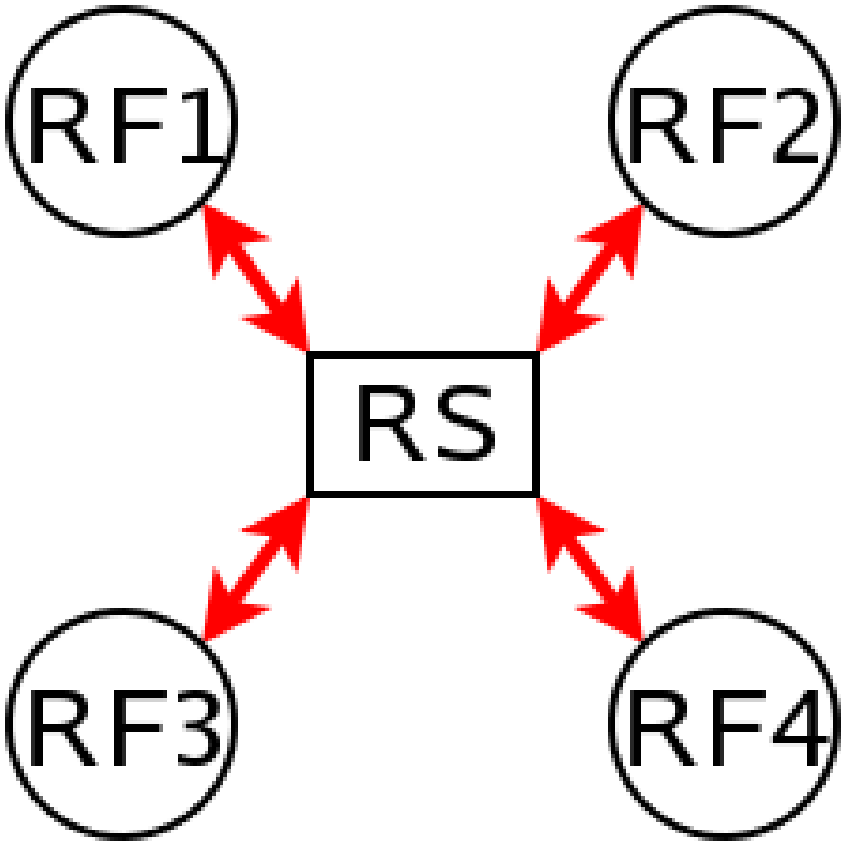



Fig. 3: Route server and clients

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 *Announcement processing model implemented by the Route Server*. 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 *Announcement processing model implemented by the Route Server*.). 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 import|export**
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.

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

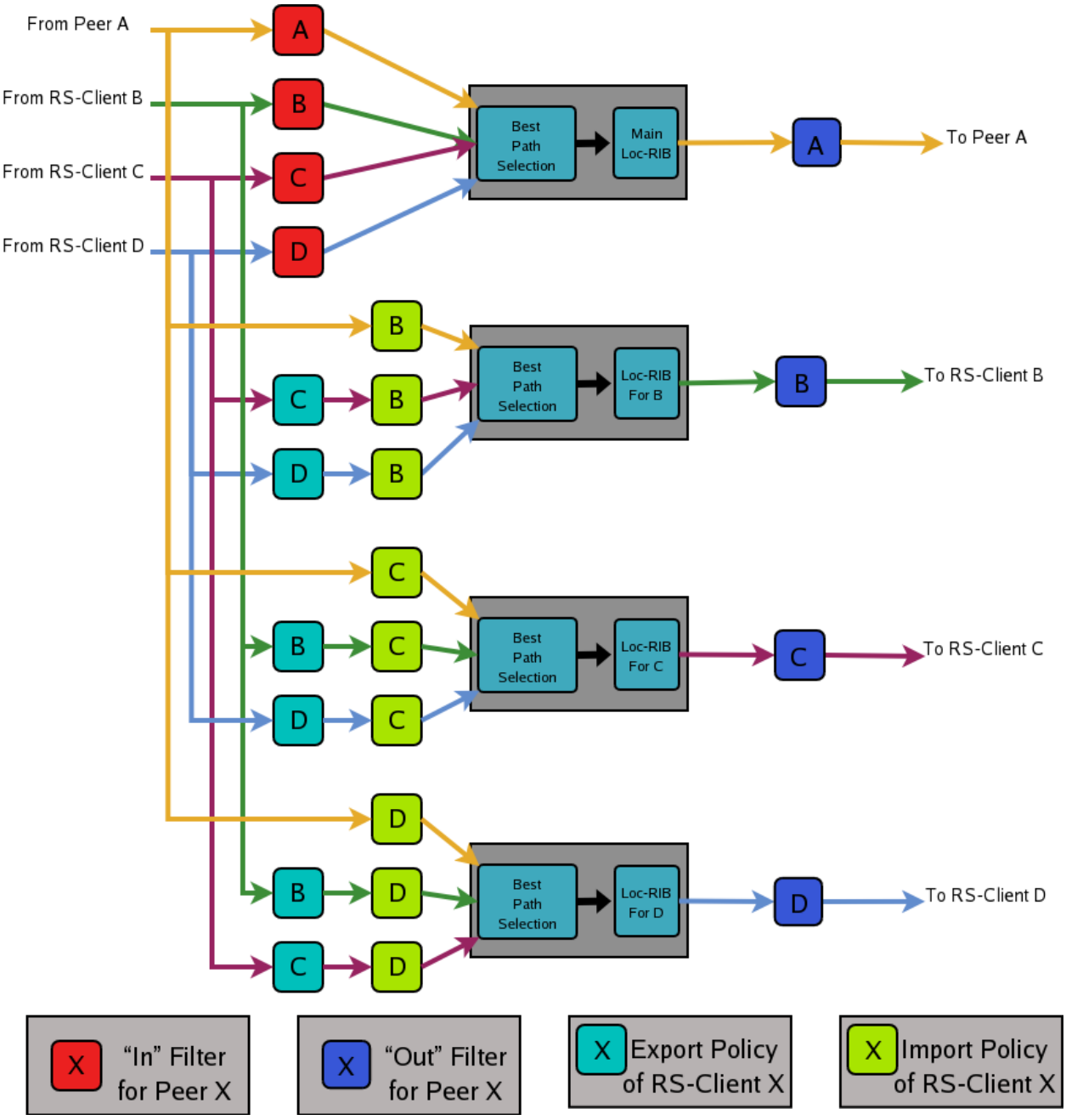



Fig. 4: Announcement processing model implemented by the Route Server

is applied before different Loc-RIBs is shown in *Announcement processing model implemented by the Route Server*.).

**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
!
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
```

(continues on next page)

(continued from previous page)

```
!
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
!
  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
!
```

(continues on next page)

(continued from previous page)

```
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 import
    neighbor 2001:0DB8::A route-map RSCLIENT-A-EXPORT export
    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 import
    neighbor 2001:0DB8::B route-map RSCLIENT-B-EXPORT export
    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 import
    neighbor 2001:0DB8::C route-map RSCLIENT-C-EXPORT export
    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
```

(continues on next page)

(continued from previous page)

```
!
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 import
...
!
...
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
  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.

### 3.2.9 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.

##### 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)
```

(continues on next page)

(continued from previous page)

```
  Advertised to non peer-group peers:
  l1(swp1) l2(swp2) l3(swp3) l4(swp4)
  65002
    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

## 3.3 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.

### 3.3.1 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.

### 3.3.2 LDP Configuration

**[no] mpls ldp**
Enable or disable LDP daemon

**[no] router-id A.B.C.D**
The following command located under MPLS router node configures the MPLS router-id of the local device.

**[no] ordered-control**
Configure LDP Ordered Label Distribution Control.

**[no] address-family [ipv4 | ipv6]**
Configure LDP for IPv4 or IPv6 address-family. Located under MPLS route node, this subnode permits configuring the LDP neighbors.

**[no] 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.

**[no] 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.

**[no] 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.

**[no] 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.

**[no] discovery hello holdtime HOLDTIME**

**[no] 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.

**[no] 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.

### 3.3.3 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#
```

### 3.3.4 LDP debugging commands

**[no] debug mpls ldp KIND**
Enable or disable debugging messages of a given kind. `KIND` can be one of:

- `discovery`
- `errors`
- `event`
- `labels`
- `messages`
- `zebra`

### 3.3.5 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#
```

## 3.4 EIGRP

**DUAL** The *Diffusing Update ALgorithm*, a Bellman-Ford based routing algorithm used by EIGRP.

EIGRP – Routing Information Protocol is widely deployed interior gateway routing protocol. EIGRP was developed in the 1990's. EIGRP is a distance-vector protocol and is based on the *DUAL* algorithms. As a distance-vector protocol, the EIGRP router send updates to its neighbors as networks change, thus allowing the convergence to a known topology.

*eigrpd* supports EIGRP as described in RFC7868

*eigrpd* invocation options. Common options that can be specified (common-invocation-options).

### 3.4.1 EIGRP Configuration

**router eigrp (1-65535) [vrf NAME]**
The *router eigrp* command is necessary to enable EIGRP. To disable EIGRP, use the *no router eigrp (1-65535)* command. EIGRP must be enabled before carrying out any of the EIGRP commands. Specify vrf NAME if you want eigrp to work within the specified vrf.

**no router eigrp (1-65535) [vrf NAME]**
Disable EIGRP.

**network NETWORK**

**no network NETWORK**
Set the EIGRP enable interface by *network*. The interfaces which have addresses matching with *network* are enabled.

This group of commands either enables or disables EIGRP interfaces between certain numbers of a specified network address. For example, if the network for 10.0.0.0/24 is EIGRP enabled, this would result in all the addresses from 10.0.0.0 to 10.0.0.255 being enabled for EIGRP. The *no network* command will disable EIGRP for the specified network.

Below is very simple EIGRP configuration. Interface *eth0* and interface which address match to *10.0.0.0/8* are EIGRP enabled.

```
!
router eigrp 1
 network 10.0.0.0/8
!
```

**passive-interface (IFNAME|default)**

**no passive-interface IFNAME**
This command sets the specified interface to passive mode. On passive mode interface, all receiving packets are ignored and eigrpd does not send either multicast or unicast EIGRP packets except to EIGRP neighbors specified with *neighbor* command. The interface may be specified as *default* to make eigrpd default to passive on all interfaces.

The default is to be passive on all interfaces.

### 3.4.2 How to Announce EIGRP route

**redistribute kernel**

**redistribute kernel metric (1-4294967295) (0-4294967295) (0-255) (1-255) (1-65535)**

**no redistribute kernel**
*redistribute kernel* redistributes routing information from kernel route entries into the EIGRP tables. *no redistribute kernel* disables the routes.

**redistribute static**

**redistribute static metric (1-4294967295) (0-4294967295) (0-255) (1-255) (1-65535)**

**no redistribute static**
*redistribute static* redistributes routing information from static route entries into the EIGRP tables. *no redistribute static* disables the routes.

**redistribute connected**

**redistribute connected metric (1-4294967295) (0-4294967295) (0-255) (1-255) (1-65535)**

**no redistribute connected**
Redistribute connected routes into the EIGRP tables. *no redistribute connected* disables the connected routes in the EIGRP tables. This command redistribute connected of the interface which EIGRP disabled. The connected route on EIGRP enabled interface is announced by default.

**redistribute ospf**

**redistribute ospf metric (1-4294967295) (0-4294967295) (0-255) (1-255) (1-65535)**

**no redistribute ospf**
*redistribute ospf* redistributes routing information from ospf route entries into the EIGRP tables. *no redistribute ospf* disables the routes.

**redistribute bgp**

**redistribute bgp metric  (1-4294967295) (0-4294967295) (0-255) (1-255) (1-65535)**

**no redistribute bgp**
*redistribute bgp* redistributes routing information from bgp route entries into the EIGRP tables. *no redistribute bgp* disables the routes.

### 3.4.3 Show EIGRP Information

**show ip eigrp [vrf NAME] topology**
Display current EIGRP status.

```
eigrpd> **show ip eigrp topology**
# show ip eigrp topo

EIGRP Topology Table for AS(4)/ID(0.0.0.0)

Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply
       r - reply Status, s - sia Status

P  10.0.2.0/24, 1 successors, FD is 256256, serno: 0
       via Connected, enp0s3
```

**show ip eigrp [vrf NAME] interface**
Display the list of interfaces associated with a particular eigrp instance.

..index:: show ip eigrp [vrf NAME] neighbor ..clicmd:: show ip eigrp [vrf NAME] neighbor

Display the list of neighbors that have been established within a particular eigrp instance.

### 3.4.4 EIGRP Debug Commands

Debug for EIGRP protocol.

**debug eigrp packets**
Debug eigrp packets

`debug eigrp` will show EIGRP packets that are sent and received.

**debug eigrp transmit**
Debug eigrp transmit events

`debug eigrp transmit` will display detailed information about the EIGRP transmit events.

**show debugging eigrp**
Display *eigrpd*'s debugging option.

`show debugging eigrp` will show all information currently set for eigrpd debug.

## 3.5 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.

### 3.5.1 Configuring isisd

There are no *isisd* specific options. Common options can be specified (common-invocation-options) to *isisd*. *isisd* needs to acquire interface information from *zebra* in order to function. Therefore *zebra* must be running before invoking *isisd*. Also, if *zebra* is restarted then *isisd* must be too.

### 3.5.2 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.

**[no] 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**

**no net XX.XXXX. ... .XXX.XX**
Set/Unset network entity title (NET) provided in ISO format.

**hostname dynamic**

**no hostname dynamic**
Enable support for dynamic hostname.

**area-password [clear | md5] <password>**

**domain-password [clear | md5] <password>**

**no area-password**

**no domain-password**
Configure the authentication password for an area, respectively a domain, as clear text or md5 one.

**log-adjacency-changes**

**no log-adjacency-changes**
Log changes in adjacency state.

**metric-style [narrow | transition | wide]**

**no metric-style**
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

**set-overload-bit**

**no set-overload-bit**
Set overload bit to avoid any transit traffic.

**purge-originator**

**no purge-originator**
Enable or disable **RFC 6232** purge originator identification.

**[no] lsp-mtu (128-4352)**
Configure the maximum size of generated LSPs, in bytes.

### 3.5.3 ISIS Timer

**lsp-gen-interval (1-120)**

**lsp-gen-interval [level-1 | level-2] (1-120)**

**no lsp-gen-interval**

**no lsp-gen-interval [level-1 | level-2]**
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)**

**no lsp-refresh-interval [level-1 | level-2]**
Set LSP refresh interval in seconds, globally, for an area (level-1) or a domain (level-2).

**max-lsp-lifetime (360-65535)**

**max-lsp-lifetime [level-1 | level-2] (360-65535)**

**no max-lsp-lifetime**

**no max-lsp-lifetime [level-1 | level-2]**
Set LSP maximum LSP lifetime in seconds, globally, for an area (level-1) or a domain (level-2).

**spf-interval (1-120)**

**spf-interval [level-1 | level-2] (1-120)**

**no spf-interval**

**no spf-interval [level-1 | level-2]**
Set minimum interval between consecutive SPF calculations in seconds.

### 3.5.4 ISIS region

**is-type [level-1 | level-1-2 | level-2-only]**

**no is-type**
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

### 3.5.5 ISIS interface

**[no] <ip|ipv6> router isis WORD [vrf NAME]**
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]**

**no isis circuit-type**
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)**

**isis csnp-interval (1-600) [level-1 | level-2]**

**no isis csnp-interval**

**no isis csnp-interval [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-interval (1-600)**

**isis hello-interval (1-600) [level-1 | level-2]**

**no isis hello-interval**

**no isis hello-interval [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)**

**isis hello-multiplier (2-100) [level-1 | level-2]**

**no isis hello-multiplier**

**no isis hello-multiplier [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)]**

**isis metric [(0-255) | (0-16777215)] [level-1 | level-2]**

**no isis metric**

**no isis metric [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**

**no isis network point-to-point**
Set network type to 'Point-to-Point' (broadcast by default).

**isis passive**

**no isis passive**
Configure the passive mode for this interface.

**isis password [clear | md5] <password>**

**no isis password**
Configure the authentication password (clear or encoded text) for the interface.

**isis priority (0-127)**

**isis priority (0-127) [level-1 | level-2]**

**no isis priority**

**no isis priority [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)**

**isis psnp-interval (1-120) [level-1 | level-2]**

**no isis psnp-interval**

**no isis psnp-interval [level-1 | level-2]**
Set PSNP interval in seconds globally, for an area (level-1) or a domain (level-2).

**isis three-way-handshake**

**no isis three-way-handshake**
Enable or disable **RFC 5303** Three-Way Handshake for P2P adjacencies. Three-Way Handshake is enabled by default.

### 3.5.6 Showing ISIS information

**show isis summary**
Show summary information about ISIS.

**show isis hostname**
Show information about ISIS node.

**show isis interface**

**show isis interface detail**

**show isis interface <interface name>**
Show state and configuration of ISIS specified interface, or all interfaces if no interface is given with or without details.

**show isis neighbor**

**show isis neighbor <System Id>**

**show isis neighbor detail**
Show state and information of ISIS specified neighbor, or all neighbors if no system id is given with or without details.

**show isis database**

**show isis database [detail]**

**show isis database <LSP id> [detail]**

**show isis database detail <LSP id>**
Show the ISIS database globally, for a specific LSP id without or with details.

**show isis topology**

**show isis topology [level-1|level-2]**
Show topology IS-IS paths to Intermediate Systems, globally, in area (level-1) or domain (level-2).

**show isis route [level-1|level-2]**
Show the ISIS routing table, as determined by the most recent SPF calculation.

### 3.5.7 Debugging ISIS

**debug isis adj-packets**

**no debug isis adj-packets**
IS-IS Adjacency related packets.

**debug isis checksum-errors**

**no debug isis checksum-errors**
IS-IS LSP checksum errors.

**debug isis events**

**no debug isis events**
IS-IS Events.

**debug isis local-updates**

**no debug isis local-updates**
IS-IS local update packets.

**debug isis packet-dump**

**no debug isis packet-dump**
IS-IS packet dump.

**debug isis protocol-errors**

**no debug isis protocol-errors**
IS-IS LSP protocol errors.

**debug isis route-events**

**no debug isis route-events**
IS-IS Route related events.

**debug isis snp-packets**

**no debug isis snp-packets**
IS-IS CSNP/PSNP packets.

**debug isis spf-events**

**debug isis spf-statistics**

**debug isis spf-triggers**

**no debug isis spf-events**

**no debug isis spf-statistics**

**no debug isis spf-triggers**
IS-IS Shortest Path First Events, Timing and Statistic Data and triggering events.

**debug isis update-packets**

**no debug isis update-packets**
Update related packets.

**show debugging isis**
Print which ISIS debug level is activate.

### 3.5.8 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
```

(continues on next page)

(continued from previous page)

```
!
router isis FOO
net 47.0023.0000.0000.0000.0000.0000.1900.0004.00
 metric-style wide
 is-type level-2-only
```

## 3.6 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.

### 3.6.1 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 `passive-interface INTERFACE`.

- 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
```

(continues on next page)

(continued from previous page)

```
  Options: 0x2  : *|-|-|-|-|-|E|*
  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
        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)
```

(continues on next page)

(continued from previous page)

```
    (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  vrf NAME**

**no router ospf  vrf NAME**
Enable or disable the OSPF process.

**ospf router-id A.B.C.D**

**no 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**

**no 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**

**no 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]**

**no 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 INTERFACE**

**no passive-interface INTERFACE**

Do not speak OSPF interface on the given 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.

**timers throttle spf (0-600000) (0-600000) (0-600000)**

**no timers throttle spf**

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|on-shutdown] (5-86400)**

**max-metric router-lsa administrative**

**no max-metric router-lsa [on-startup|on-shutdown|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)**

**no auto-cost reference-bandwidth**

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)**

**no network A.B.C.D/M area A.B.C.D**

**no 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.

In some cases it may be more convenient to enable OSPF on a per interface/subnet basis (`ip ospf area AREA [ADDR]`).

**proactive-arp**

**no 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.

### Areas

**area A.B.C.D range A.B.C.D/M**

**area (0-4294967295) range A.B.C.D/M**

**no area A.B.C.D range A.B.C.D/M**

**no area (0-4294967295) range A.B.C.D/M**
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. Summarizing Type-7 AS-external-LSAs isn't supported yet by FRR.

```
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 IPV4_PREFIX not-advertise**

**no area A.B.C.D range IPV4_PREFIX 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 IPV4_PREFIX substitute IPV4_PREFIX**

**no area A.B.C.D range IPV4_PREFIX substitute IPV4_PREFIX**
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. 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**

**no area A.B.C.D virtual-link A.B.C.D**

**no area (0-4294967295) virtual-link A.B.C.D**

**area A.B.C.D shortcut**

**area (0-4294967295) shortcut**

**no area A.B.C.D shortcut**

**no 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**

**no area A.B.C.D stub**

**no 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**

**no area A.B.C.D stub no-summary**

**no 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 default-cost (0-16777215)**

**no 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**

**no area A.B.C.D export-list NAME**

**no 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**

**no area A.B.C.D import-list NAME**

**no 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**

**no area A.B.C.D filter-list prefix NAME in**

**no area A.B.C.D filter-list prefix NAME out**

**no area (0-4294967295) filter-list prefix NAME in**

**no 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**

**no area A.B.C.D authentication**

**no 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]**

**no ip ospf area [ADDR]**
Enable OSPF on the interface, optionally restricted to just the IP address given by *ADDR*, putting it in the *AREA* area. Per interface area settings take precedence to network commands (`network A.B.C.D/M area A.B.C.D`).

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

**ip ospf authentication-key AUTH_KEY**

**no ip ospf authentication-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**

**no ip ospf message-digest-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 cost (1-65535)**

**no ip ospf cost**
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)**

**no ip ospf dead-interval**
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)**

**no ip ospf hello-interval**
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 network (broadcast|non-broadcast|point-to-multipoint|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*.

**no ip ospf network**
Set explicitly network type for specified interface.

**ip ospf priority (0-255)**

**no ip ospf priority**
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)**

**no ip ospf retransmit interval**
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]**

**no 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 area (A.B.C.D|(0-4294967295))**

**no ip ospf area**
Enable ospf on an interface and set associated area.

### 3.6.2 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 (kernel|connected|static|rip|bgp)**

**redistribute (kernel|connected|static|rip|bgp) ROUTE-MAP**

**redistribute (kernel|connected|static|rip|bgp) metric-type (1|2)**

**redistribute (kernel|connected|static|rip|bgp) metric-type (1|2) route-map WORD**

**redistribute (kernel|connected|static|rip|bgp) metric (0-16777214)**

**redistribute (kernel|connected|static|rip|bgp) metric (0-16777214) route-map WORD**

**redistribute (kernel|connected|static|rip|bgp) metric-type (1|2) metric (0-16777214)**

**redistribute (kernel|connected|static|rip|bgp) metric-type (1|2) metric (0-16777214) route-**

**no redistribute (kernel|connected|static|rip|bgp)**

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 *passive-interface* configuration.

**See also:**

clicmd:*passive-interface INTERFACE*.

**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**

**no default-information originate**
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|ospf**

**no distribute-list NAME out (kernel|connected|static|rip|ospf**

Apply the access-list filter, NAME, to redistributed routes of the given type before allowing the routes to redistributed into OSPF (*ospf redistribution*).

**default-metric (0-16777214)**

**no default-metric**

**distance (1-255)**

**no distance (1-255)**

**distance ospf (intra-area|inter-area|external) (1-255)**

**no distance ospf**

**router zebra**

**no router zebra**

### 3.6.3 Showing Information

**show ip ospf**
Show information on a variety of general OSPF and area state and configuration information.

**show ip ospf interface [INTERFACE]**
Show state and configuration of OSPF the specified interface, or all interfaces if no interface is given.

**show ip ospf neighbor**

**show ip ospf neighbor INTERFACE**

**show ip ospf neighbor detail**

**show ip ospf neighbor INTERFACE detail**

**show ip ospf database**

**show ip ospf database (asbr-summary|external|network|router|summary)**

**show ip ospf database (asbr-summary|external|network|router|summary) LINK-STATE-ID**

**show ip ospf database (asbr-summary|external|network|router|summary) LINK-STATE-ID adv-rout**

**show ip ospf database (asbr-summary|external|network|router|summary) adv-router ADV-ROUTER**

**show ip ospf database (asbr-summary|external|network|router|summary) LINK-STATE-ID self-ori**

**show ip ospf database (asbr-summary|external|network|router|summary) self-originate**

**show ip ospf database max-age**

**show ip ospf database self-originate**

**show ip ospf route**
Show the OSPF routing table, as determined by the most recent SPF calculation.

### 3.6.4 Opaque LSA

**ospf opaque-lsa**

**capability opaque**

**no ospf opaque-lsa**

**no capability opaque**

**show ip ospf database (opaque-link|opaque-area|opaque-external)**

**show ip ospf database (opaque-link|opaque-area|opaque-external) LINK-STATE-ID**

**show ip ospf database (opaque-link|opaque-area|opaque-external) LINK-STATE-ID adv-router AI**

**show ip ospf database (opaque-link|opaque-area|opaque-external) adv-router ADV-ROUTER**

**show ip ospf database (opaque-link|opaque-area|opaque-external) LINK-STATE-ID self-originat**

**show ip ospf database (opaque-link|opaque-area|opaque-external) self-originate**
Show Opaque LSA from the database.

### 3.6.5 Router Information

**router-info [as | area]**

**no router-info**
Enable Router Information (**RFC 4970**) LSA advertisement with AS scope (default) or Area scope flooding when area is specified. Old syntax *router-info area <A.B.C.D>* is always supported but mark as deprecated as the area ID is no more necessary. Indeed, router information support multi-area and detect automatically the areas.

**pce address <A.B.C.D>**

**no pce address**

**pce domain as (0-65535)**

**no pce domain as (0-65535)**

**pce neighbor as (0-65535)**

**no pce neighbor as (0-65535)**

**pce flag BITPATTERN**

**no pce flag**

**pce scope BITPATTERN**

**no pce scope**
The commands are conform to **RFC 5088** and allow OSPF router announce Path Computation Element (PCE) capabilities through the Router Information (RI) LSA. Router Information must be enable prior to this. The command set/unset respectively the PCE IP address, Autonomous System (AS) numbers of controlled domains, neighbor ASs, flag and scope. For flag and scope, please refer to :rfc‘5088‘ for the BITPATTERN recognition. Multiple ’pce neighbor’ command could be specified in order to specify all PCE neighbours.

**show ip ospf router-info**
Show Router Capabilities flag.

**show ip ospf router-info pce**
Show Router Capabilities PCE parameters.

### 3.6.6 Debugging OSPF

**debug ospf packet (hello|dd|ls-request|ls-update|ls-ack|all) (send|recv) [detail]**

**no debug ospf packet (hello|dd|ls-request|ls-update|ls-ack|all) (send|recv) [detail]**
Dump Packet for debugging

**debug ospf ism**

**debug ospf ism (status|events|timers)**

**no debug ospf ism**

**no debug ospf ism (status|events|timers)**
Show debug information of Interface State Machine

**debug ospf nsm**

**debug ospf nsm (status|events|timers)**

**no debug ospf nsm**

**no debug ospf nsm (status|events|timers)**
Show debug information of Network State Machine

**debug ospf event**

**no debug ospf event**
Show debug information of OSPF event

**debug ospf nssa**

**no debug ospf nssa**
Show debug information about Not So Stub Area

**debug ospf lsa**

**debug ospf lsa (generate|flooding|refresh)**

**no debug ospf lsa**

**no debug ospf lsa (generate|flooding|refresh)**
Show debug detail of Link State messages

**debug ospf zebra**

**debug ospf zebra (interface|redistribute)**

**no debug ospf zebra**

**no debug ospf zebra (interface|redistribute)**
Show debug information of ZEBRA API

**show debugging ospf**

### 3.6.7 OSPF Configuration Examples

A simple example, with MD5 authentication enabled:

```
!
interface bge0
 ip ospf authentication message-digest
 ip ospf message-digest-key 1 md5 ABCDEFGHIJK
!
```

(continues on next page)

(continued from previous page)

```
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 eth0
 ip ospf authentication message-digest
 ip ospf message-digest-key 1 md5 ABCDEFGHIJK
!
interface ppp0
!
interface br0
 ip ospf authentication message-digest
 ip ospf message-digest-key 2 md5 XYZ12345
!
router ospf
 ospf router-id 192.168.0.1
 redistribute connected
 passive interface ppp0
 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
!
```

Then:

```
hostname HOSTNAME
password PASSWORD
log file
!
!
interface eth0
 ip ospf hello-interval 60
 ip ospf dead-interval 240
!
interface eth1
 ip ospf hello-interval 60
 ip ospf dead-interval 240
!
!
router ospf
 ospf router-id 192.168.1.1
 network 192.168.0.0/16 area 1

!
line vty
```

A router information example with PCE advertisement:

```
!
router ospf
 ospf router-id 192.168.1.1
 network 192.168.0.0/16 area 1
 router-info area 0.0.0.1
 pce address 192.168.1.1
 pce flag 0x80
 pce domain as 65400
 pce neighbor as 65500
 pce neighbor as 65200
 pce scope 0x80
!
```

## 3.7 OSPFv3

*ospf6d* is a daemon support OSPF version 3 for IPv6 network. OSPF for IPv6 is described in **RFC 2740**.

### 3.7.1 OSPF6 router

**router ospf6**

**ospf6 router-id A.B.C.D**
Set router's Router-ID.

**interface IFNAME area (0-4294967295)**

**interface IFNAME area A.B.C.D**
Bind interface to specified area, and start sending OSPF packets. *area* can be specified as 0.

**timers throttle spf DELAY INITIAL-HOLDTIME MAX-HOLDTIME**

**no timers throttle spf**
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**

**no auto-cost reference-bandwidth**
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.

### 3.7.2 OSPF6 area

Area support for OSPFv3 is not yet implemented.

### 3.7.3 OSPF6 interface

**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 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.

### 3.7.4 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.

### 3.7.5 Redistribute routes to OSPF6

**redistribute static**

**redistribute connected**

**redistribute ripng**

### 3.7.6 Showing OSPF6 information

**show ipv6 ospf6**

**show ipv6 ospf6 database**
This command shows LSA database summary. You can specify the type of LSA.

**show ipv6 ospf6 interface**
To see OSPF interface configuration like costs.

**show ipv6 ospf6 neighbor**
Shows state and chosen (Backup) DR of neighbor.

**show ipv6 ospf6 request-list A.B.C.D**
Shows requestlist of neighbor.

**show ipv6 route ospf6**
This command shows internal routing table.

**show ipv6 ospf6 zebra**
Shows state about what is being redistributed between zebra and OSPF6

### 3.7.7 OSPF6 Configuration Examples

Example of ospf6d configured on one interface and area:

```
router ospf6
 ospf6 router-id 212.17.55.53
 area 0.0.0.0 range 2001:770:105:2::/64
 interface eth0 area 0.0.0.0
!
```

## 3.8 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.

### 3.8.1 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**
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.

**no router rip**
Disable RIP.

**network NETWORK**

**no 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**

**no 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**

**no neighbor A.B.C.D**
Specify RIP neighbor. When a neighbor doesn't understand multicast, this command is used to specify neighbors. In some cases, not all routers will be able to understand multicasting, where packets are sent to a network or a group of addresses. In a situation where a neighbor cannot process multicast packets, it is necessary to establish a direct link between routers. The neighbor command allows the network administrator to specify a router as a RIP neighbor. 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)**

**no passive-interface IFNAME**
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**

**no ip split-horizon**
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*.

### 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 1.

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.

**no version**
Reset the global version setting back to the default.

**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 kernel**

**redistribute kernel metric (0-16)**

**redistribute kernel route-map ROUTE-MAP**

**no redistribute kernel**
*redistribute kernel* redistributes routing information from kernel route entries into the RIP tables. *no redistribute kernel* disables the routes.

**redistribute static**

**redistribute static metric (0-16)**

**redistribute static route-map ROUTE-MAP**

**no redistribute static**
*redistribute static* redistributes routing information from static route entries into the RIP tables. *no redistribute static* disables the routes.

**redistribute connected**

**redistribute connected metric (0-16)**

**redistribute connected route-map ROUTE-MAP**

**no redistribute connected**
Redistribute connected routes into the RIP tables. *no redistribute connected* disables the connected routes in the RIP tables. This command redistribute connected of the interface which RIP disabled. The connected route on RIP enabled interface is announced by default.

**redistribute ospf**

**redistribute ospf metric (0-16)**

**redistribute ospf route-map ROUTE-MAP**

**no redistribute ospf**
*redistribute ospf* redistributes routing information from ospf route entries into the RIP tables. *no redistribute ospf* disables the routes.

**redistribute bgp**

**redistribute bgp metric (0-16)**

**redistribute bgp route-map ROUTE-MAP**

**no redistribute bgp**
*redistribute bgp* redistributes routing information from bgp route entries into the RIP tables. *no redistribute bgp* disables the routes.

If you want to specify RIP only static routes:

**default-information originate**

**route A.B.C.D/M**

**no 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 ACCESS_LIST DIRECT IFNAME**
You can apply access lists to the interface with a *distribute-list* command. ACCESS_LIST is the access list name. DIRECT is `in` or `out`. If DIRECT is `in` the access list is applied to input packets.

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.

**distribute-list prefix PREFIX_LIST (in|out) IFNAME**
You can apply prefix lists to the interface with a *distribute-list* command. PREFIX_LIST is the prefix list name. Next is the direction of `in` or `out`. If DIRECT is `in` the access list is applied to input packets.

### 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)**

**no 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)**

**no distance (1-255)**
Set default RIP distance to specified value.

**distance (1-255) A.B.C.D/M**

**no 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**

**no 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**

**no ip rip authentication mode md5**
Set the interface with RIPv2 MD5 authentication.

**ip rip authentication mode text**

**no ip rip authentication mode text**
Set the interface with RIPv2 simple password authentication.

**ip rip authentication string STRING**

**no 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**

**no 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.

**no timers basic**
The *no timers basic* command will reset the timers to the default settings listed above.

### Show RIP Information

To display RIP routes.

**show ip rip**
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 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.

## 3.9 RIPng

*ripngd* supports the RIPng protocol as described in **RFC 2080**. It's an IPv6 reincarnation of the RIP protocol.

### 3.9.1 Invoking ripngd

There are no *ripngd* specific invocation options. Common options can be specified (common-invocation-options).

### 3.9.2 ripngd Configuration

Currently ripngd supports the following commands:

**router ripng**
Enable RIPng.

**flush_timer TIME**
Set flush timer.

**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.

**router zebra**
This command is the default and does not appear in the configuration. With this statement, RIPng routes go to the *zebra* daemon.

### 3.9.3 ripngd Terminal Mode Commands

**show ip ripng**

**show debugging ripng**

**debug ripng events**

**debug ripng packet**

**debug ripng zebra**

### 3.9.4 ripngd Filtering Commands

**distribute-list ACCESS_LIST (in|out) IFNAME**
You can apply an access-list to the interface using the *distribute-list* command. ACCESS_LIST is an access-list name. *direct* is `in` or `out`. If *direct* is `in`, the access-list is applied only to incoming packets.:

```
distribute-list local-only out sit1
```

## 3.10 STATIC

STATIC is a daemon that handles the installation and deletion of static routes.

### 3.10.1 Static Route Commands

Static routing is a very fundamental feature of routing technology. It defines a static prefix and gateway.

**ip route NETWORK GATEWAY table TABLENO nexthop-vrf VRFNAME DISTANCE vrf VRFNAME**

**ipv6 route NETWORK from SRCPREFIX GATEWAY table TABLENO nexthop-vrf VRFNAME DISTANCE vrf VI**
NETWORK is destination prefix with a valid v4 or v6 network based upon initial form of the command. GATEWAY is gateway for the prefix it currently must match the v4 or v6 route type specified at the start of the command. GATEWAY can also be treated as an interface name. If the interface name is `null0` then zebra installs a blackhole route. 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. `nexthop-vrf` allows you to create a leaked route with a nexthop in the specified VRFNAME vrf VRFNAME allows you to create the route in a specified vrf. `nexthop-vrf` cannot be currently used with namespace based vrfs currently as well. The v6 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.

### 3.10.2 Multiple nexthop static route

To create multiple nexthops to the same NETWORK, 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 multihop 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
```

CHAPTER

FOUR

# NAT

## 4.1 NAT

Network Address Translation is a method of changing packet IP address. SoodarOS uses PNAT variation which change Port and Address.

### 4.1.1 NAT Static Mapping

Define a static mapping for address translation.

#### Address only NAT

In this mode, only address is translated to given address. Depending on flow direction( whether it's in2out or out2in) the source or destination of packet is changed.

When a packet containing a *source* same as *local address* is passing through an *input* interface, its *source* is replaced with *global address*. When a packet containing a *destination* same as *global address* is passing 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 NAT static table. first IP is local address and second IP is 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 just translates a specified protocol on a defined port. So when defining an entry, we introduce the protocol and desired ports too. All other aspects of this entry( including behaviour) is like the simpe *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 NAT static table. first IP is local address and the number following is its port. second IP is 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*

### 4.1.2 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 IP pool.

For every *source translation*, a new session is created and it's state is keeped. so the packets coming from an *output* interface and having a matched session, it's destination is changed with respect to the session's information.

#### NAT IP pool

A pool of available IP addresses to use as a NAT IP.

**ip nat pool PNAT44 A.B.C.D [ A.B.C.D ]**

Add an IP( or an IP range if second IP is provided) to a nat pool named PNAT44

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
```

First command 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.

#### 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.

### 4.1.3 Logging

Debugging logs can be set in case of need.

**[no] debug nat44 event**
log data plane installation processes and results

### 4.1.4 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
```

CHAPTER

FIVE

# QOS

## 5.1 QoS

### 5.1.1 Class Map

A class map is a set of rules to classify traffic.

#### Define Class Map

Class maps operate in two modes:

- Match all mode. Requires all circumstances be fulfilled
- Just matching a criterion is enough to classify the traffic as defined class.

**class-map match-all CNAME**
Create a match all class map and enter in class map config mode

**class-map match-any CNAME**
Create a match any class map and enter in class map config mode

**no class-map CNAM**
Removes a class map

Example :

```
n1(config)# class-map match-all cmap1
n1(config-cmap)#
```

#### Define matching criteria

As said, a class map is a tool to classify the traffic. so we need to define criteria. The keyword `match` is used to define a rule for matching. A packet could be match against its source, destinaion, etc.

**match any**
Every packet is accpeted.

**no match any**
Negate match any command and remove it from criteria

**[no] match access-list ACL**

**Note:** Only `permit` rules in ACL is considered.

**[no] match source-address A.B.C.D/M**

**[no] match destination-address A.B.C.D/M**

**[no] match source-address X:X::X:X/M**

**[no] match destination-address X:X::X:X/M**

**[no] match dscp (0-63)**
Match against packet DSCP value

**[no] match protocol (0-63)**
Match against packet protocol number

Example :

```
n1(config-cmap)# match destination-address 200.1.2.2
n1(config-cmap)# match access-list acl1
```

### 5.1.2 Policy Map

A policy map, is set of traffic policies and is attached to an interface. Currently the only policy available, is *traffic policing*. Policy map uses class maps as a base and define policies based on class map.

#### Define Policy Map

**policy-map NAME**
Create a policy map

**no policy-map NAME**
Delete a policy map

#### Define a new policy

**class CNAME**
Enter class map policy config mode

**police BPS [NORMALBURST [MAXBURST]] conform-action ACTION exceed-action ACTION [violat**
Define a police policy for entered class map. in above command, `BPS` is average bitrate of this traffic class.

`NORMALBURST` and `MAXBURST` are token buckets size.

ACTIONs are : *transmit*, *drop* and *set-dscp-transmit*. and the conditions are *conform*, *exceed* and *violate* ( which is optional).

*transmit* and *drop* actions, work as their names suggest. the *set-dscp-transmit* action, allows the traffic to be passed. But the DSCP value of packet is changed.

*conform* condition, is when data burst, is below the NORMALBURST rate. the *exceed* condition, is when the data burst is between NORMALBURST and MAXBURST.

*violate* condition, is when data burst, is over *MAXBURST* rate. When a violate-action is not defined, the algorithm is just a single token bucket algorithm and only *conform* ( below NORMALBURST) and *exceed* ( over NORMALBURST) occurs.

**no class CNAME**
Remove all policies defined for a class map

**no police**
Remove police policy defined for a class map

Example :

```
n1(config)# policy-map pmap1
n1(config-pmap)# class cmap1
n1(config-pmap-c)# police 100K conform-action transmit exceed-action drop
n1(config-pmap)# class cmap2
n1(config-pmap-c)# police 10K conform-action transmit exceed-action set-dscp-
↪transmit 24 violate-action drop
```

### Apply to interface

Currently a policy map can be applied to just ingress traffic.

**service-policy PMAP in**

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
```

Debugging logs can be set in case of need.

**[no] debug qos event**
log data plane installation processes and results

CHAPTER

SIX

# ACCESS CONTROL LIST

## 6.1 IP Access List

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 result and whole process of evaluation is terminated.

### 6.1.1 Define an ACL

To define an *ipv4* access list, just issue the following command:

**ip access-list ACL4**

ACL4 is access list name.

Example :

```
soodar(config)# ip access-list ACL_TEST
soodar(config-nacl)#
```

### 6.1.2 Delete an ACL

Use no command to delete an ACL

**no ip access-list ACL4**

### 6.1.3 Remark

You can include comment or remark about IP access list

**remark LINE ..**

Adds a remark for the access-list. The remark indicates the purpose of the access-list.

### 6.1.4 Define an entry

Rules( or entries) can be defined in one:

1. In access-list configuration mode

Since access lists uses *first match* approach, entries have priorities. It is by default sequential( the first entry has the highest priority) and behaviour 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
```

**(1-4294967295) <deny|permit> <any|A.B.C.D/M> <any|A.B.C.D/M> [exact-match]**
Create a simple IPv4 entry. Matches against all IP packets. if `exact-match` is entered, the prefixes are also checked and should be the same( 192.168.1.1/24 is not a match in 192.168.1.1/16). The first input is entry sequence number. The next input is the action done when entry is matched. Then we have *source* and *destination* prefix. intead of each, user can input `any` to match every address.

Example :

```
n1(config)# ip access-list TEST_ACL1
n1(config-nacl)# deny any 10.1.16.68/32
n1(config-nacl)# permit any any
```

**Note:** It's the best practice to add a `permit any` rule as latest entry, because by default if a packet doesn't match against non of entries, it will be dropped.

**(1-4294967295) <deny|permit> tcp <any|A.B.C.D/M> SRC_PORT <any|A.B.C.D/M> DST_PORT [tcp-fla**

**(1-4294967295) <deny|permit> udp <any|A.B.C.D/M> SRC_PORT <any|A.B.C.D/M> DST_PORT [exact-m**

```
n1(config)# ip access-list DENY_DNS
n1(config-nacl)# deny udp any eq domain any

   Deny any DNS service
```

**(1-4294967295) <deny|permit> icmp <any|A.B.C.D/M> <any|A.B.C.D/M> ICMP_TYPE_CODES [exact-ma**
ICMP_TYPE_CODES is Well known ICMP message code types to match. They can be defined by their name( like `echo-reply`) or by their *code* and *type* values.

#### 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 it's sequence number, it's defition or both of them

### 6.1.5 Apply ACL

An ACL is applied to ingress or egress traffic of an interface.

**ip access-group ACL4 in**
Apply an IPv4 ACL to interface's input traffic

Example :

```
n1(config)# interface ge3
n1(config-if)# ip access-group IN_ACL in
```

**ipv6 access-group ACL6 in**

**ip access-group ACL4 out**
Apply an IPv4 ACL to interface's output traffic

**ipv6 access-group ACL6 out**

**ip access-group ACL4 in out**
Apply ACL to both ways of traffic

**ipv6 access-group ACL6 in out**

**no ip access-group ACL4 in**
Detach an IPv4 ACL from interface's input traffic

**no ipv6 access-group ACL6 in**

**no ip access-group ACL4 out**
Detach an IPv4 ACL from interface's output traffic

**no ipv6 access-group ACL6 out**

**no ip access-group ACL4 in out**
Detach ACL from both ways of traffic

**no ipv6 access-group ACL6 in out**

### 6.1.6 Debug

**show ip access-list [NAME]**
Example :

```
n1# show ip access-list
IP access list TESTACL1
   10 permit tcp 1.1.1.10/32 eq 200 2.1.1.0/24 ge 5060 tcp-flag-mask 0 tcp-flag-
↪value 0
```

Debugging logs can be set in case of need.

**[no] debug acl event**
log data plane installation processes and results

CHAPTER

SEVEN

# VRF

## 7.1 VRF

### 7.1.1 Define a VRF

**vrf (VRF_NAME)**
Define a new VRF

Example :

```
n1(config)# vrf vrf-green
```

### 7.1.2 Add an interface to VRF

**ip vrf forwarding NAME**
Add interface to VRF NAME

**no ip vrf forwarding [NAME]**
Add interface to VRF default

**Note:** When adding/removing interface to 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
```

### 7.1.3 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
```

(continues on next page)

(continued from previous page)

```
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
```

### 7.1.4 VRF FIB

Example:

```
soodar1# sh ip ospf vrf vrf-green route
soodar1# sh ip fib vrf vrf-green
soodar1# sh ip fib vrf all
```

### 7.1.5 Display VRF info

**show vrf**

Example:

```
n1# sh vrf
vrf vrf-blue id 5 table 300
```

### 7.1.6 Logging

Debugging logs can be set in case of need.

**[no] debug vrf event**
log data plane installation processes and results

CHAPTER

EIGHT

# MPLS

## 8.1 MPLS

### 8.1.1 Enable MPLS on interface

**[no] mpls ip**

**[no] 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.

CHAPTER

NINE

# SECURITY

## 9.1 Tunnels

SoodarOS support many Layer 2 and Layer 3 tunnels including: `GRE`, `IPIP`, `VXLAN` and `VPLS`

### 9.1.1 Layer 3 Tunnels

Layer 3 Tunnels Includes:

- GRE
- IPIP

#### Create L3 Tunnel

**`interface tunnel [vrf VRF] (0-1023)`**
Create a new tunnel with instance number input.

**`no interface tunnel`**
Delete a tunnel

**`tunnel source <A.B.C.D|X:X::X:X>`**

**`tunnel destination <A.B.C.D|X:X::X:X>`**

**`tunnel mode ipip`**
Set tunnel mode to IP-IP

**`tunnel mode ipip multipoint`**
Set tunnel mode to IP-IP multipoint

**`tunnel mode gre`**

**`tunnel mode gre multipoint`**
Set tunnel mode to GRE MP.

> **Note:** In `multipoint` mode, `destination` should not be set. use `nhrp` instead.

**`tunnel protection ipsec profile IPSECPROFILE`**
Protect tunnel with IPSec. see *Profile*

**Warning:** Currently protection mode is only supported in P2P tunnels.

**Note:** When tunnel is in protected mode, It will be put in shutdown mode untill the IPSec SAs are established.

**Note:** When IPSec SAs protecting a tunnel are gone, the tunnel will immediately shutdown. Reestablishing SAs make tunnel available again.

### 9.1.2 Logging

Debugging logs can be set in case of need.

**[no] debug tunnel event**
log data plane installation processes and results

### 9.1.3 GRE configuration example

In 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 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
```

### 9.1.4 GRE-MP configuration example

Currently only NHRP static mapping is available.

In 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 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
```

### 9.1.5 VXLAN

Virtual Extensible LAN (VXLAN) is a proposed encapsulation protocol for running an overlay network on existing Layer 3 infrastructure.

**interface nve (0-1023)**
Create a NVE interface with instance

Example :

```
soodar(config)# interface nve 40
```

**source-ip <A.B.C.D|X:X::X:X>**
Set NVE source IP

**ingress-replication A.B.C.D**
Set NVE destination IP

**member vni (1-16777214)**
Associate NVE to VNI number.

**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 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.

**[no] debug vxlan event**
log data plane installation processes and results

### 9.1.6 VPLS

Virtual Private LAN Service( VPLS) is a method to extend LANs on network. SoodarOS, Support VPLS on a MPLS core network. To achieve this, first we need to create a `mpls-tunnel` interface. 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 a L2VPN using VPLS technology

**member pseudowire PW**
Add a `mpls-tunnel` to this L2VPN and enters 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 route to target's LSR-ID to establish 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 same split-horizon group id. It's to prevent from loops in packets( since bridge flooding is enabled and our topology is full-mesh).

#### Logging

Debugging logs can be set in case of need.

**[no] debug vpls event**
log data plane installation processes and results

## 9.2 PKI

Public Key Infrastructure manages certificates for supporting IPSec protocol. This includes: - RSA key generation. - Import Certificate Authorities. - PKCS#10 Certificate Signing Request( CSR).

**Note:** All PKI actions are permanent jobs; It does not appear in `running config` and it will be preserved after Router reboot.

**Note:** Currently no certificate revocation method is supported.

### 9.2.1 Key Generation

A pair of private/public keys is used for issuing certificate request or used in other protocols.

**crypto key generate rsa label NAME modulus (360-4096)**
Generate a new RSA key pair and store it as NAME. The key modulus can be determined( default is 2048).

**crypto key generate x25519 label LABEL**
Generate a new X25519 key pair and store it as LABEL.

**crypto key generate rawraw label LABEL bytes (32-1024)**
Generate Raw bytes and store them as LABEL. Number of generated bytes can be set( default is 32).

### 9.2.2 Importing a CA

To import a CA first we need to define a `trustpoint`. A `trustpoint` is basically a CA( and optionally a general purpose certificate signed by that CA). After defining `trustpoint`, an authentication is needed to import the CA. This certificate could be self-signed and 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**
Create a new `trustpoint`

**crypto pki authenticate TP**
Authenticate the trustpoint TP and write its CA to 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"
  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, 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
```

### 9.2.3 Generate a certificate and CSR

User can request for a certificate signing and import that certificate. To do this, a `trustpoint` needs to have these 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 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 screen and SoodarOS administrator need to copy it and sign it by a CA. Now to import this signed certificate, an authentication for this `trustpoint` is needed.

**subject-name LINE...**
Certificate SN setting

**subject-alt-name LINE**
Certificate SAN setting

---

**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 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 cetificate, should be signed by the same CA that the `trustpoint` is authenticated or else, it will fail 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
```

(continues on next page)

(continued from previous page)

```
  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.

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-----
```

(continues on next page)

(continued from previous page)

```
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"
  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
```

### 9.2.4 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` generate 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
  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
```

### 9.2.5 Removing a private key

Admin can remove unused private keys. Removing is done in a secure way by shreding and zeroing the key file.

**`crypto key zeroize RSAKEY`**
Shred a key pair.

---

**Note:** Removing a key, makes the ‘‘trustpoint‘‘s using them invalid. It's admin duty to take care of this situation and remove unused keys, or remove all certificates depending on that key.

---

### 9.2.6 Removing a trustpoint

Admin can remove a trustpoint. This action removes both the CA and general purpose certificate( if available) and update system CA database.

**`no crypto pki trustpoint TPNAME`**

### 9.2.7 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 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"
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 [RSAKEY] [json]**
Show RSA key information. If key name is not provided, all keys on system are shown. Output can be json

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
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,
```

(continues on next page)

(continued from previous page)

```
      "subject_key":"FCC893035EDA7E736D0A612BAD1D000612C87724",
      "key_id":"E5611192FEAD3FDFA877A0BAC5F336480A8C2D97"
    },
    {
      "label":"x25519-key",
      "algorithm":"X25519",
      "public_key":
↪"DEE5089576AD02780EFEF6908034E6BD471C2C6DF7FE68FC77F12C5DFCDB9D59"
    },
    {
      "label":"raw-key",
      "algorithm":"RAW",
      "length":256
    }
  ]
}
```

## 9.3 Wireguard

WireGuard is a communication protocol and free and open-source software that implements encrypted virtual private networks, and was designed with the goals of ease of use, high speed performance, and low attack surface.

SoodarOS supports Wireguard both as a wireguard server and as a wireguard client.

### 9.3.1 Interface

To start a wireguard tunnel, first a wireguard interface should be created.

**interface wireguard (0-1023)**
Create a wireguard interface instance.

**wireguard source A.B.C.D**
Set wireguard tunnel source.

**wireguard private-key X25519KEY**
Use x25519 key with `X25519KEY` label as wireguard private key.

**wireguard port (1000-65535)**
Wireguard's UDP listen port.

### 9.3.2 Server

Each wireguard server instance can have multiple peers. Each peer consist 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:** multiple ranges can be added bu issuing allowed-ip command multiple times.

**no allowed-ip A.B.C.D/M**
Remove A.B.C.D/M from peer's allowed IP ranges.

### 9.3.3 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), server address and its listening port.

**port (1000-65535)**
Peer's listening port.

**endpoint A.B.C.D**
Peer's endpoint address.

### 9.3.4 Debug

Admin can see wireguard status by using `show wireguard` command.

**show wireguard [(1-1024) PEER] [json]**
Show wireguard status. If instance is not indicated, show all wireguards and their peers status. User can specify which instance and which peer of that instance is in his interest. By adding `json` option to command, output transforms to json.

```
server# show wireguard
Wireguard 10
  Source: 200.1.3.1
  Key: key1
  Public key: 7D61BA2FA556FD7B4AA0D54114575DF6FBC5AB9B96337C4A438E85CDFC77ED7C
  Port: 5100

  Peer n3:
    Public key: 950A6657CDE2193C786FF4771A46318AB86B9CB60BA071E344E8C094EBEEF662
    Persistent keepalive: 25
    Allowed IPs:
     - 3.1.1.0/24
Wireguard 20
  Source: 200.1.2.1
  Key: key2
  Public key: DF6DEA63D7F5E11F9115C1CAA08D78FAFE6BA003739952B8094BC7AE744D235A
  Port: 5100

  Peer n2:
    Public key: 1D819A6950BBC16F04D86FBF8AA660434AEE12D77888E2F534641E9E7C51EEE2
    Persistent keepalive: 25
    Allowed IPs:
     - 2.1.1.0/24
server# show wireguard 20
Wireguard 20
  Source: 200.1.2.1
  Key: key2
  Public key: DF6DEA63D7F5E11F9115C1CAA08D78FAFE6BA003739952B8094BC7AE744D235A
  Port: 5100
```

(continues on next page)

(continued from previous page)

```
  Peer n2:
    Public key: 1D819A6950BBC16F04D86FBF8AA660434AEE12D77888E2F534641E9E7C51EEE2
    Persistent keepalive: 25
    Allowed IPs:
     - 2.1.1.0/24
n3# show wireguard 10 server
Wireguard 10
  Peer server:
    Public key: 7D61BA2FA556FD7B4AA0D54114575DF6FBC5AB9B96337C4A438E85CDFC77ED7C
    Endpoint: 200.1.3.1
    Persistent keepalive: 25
    Port: 5100
    Allowed IPs:
     - 1.1.1.0/24
server# show wireguard json
{
  "wireguards":[
    {
      "instance":10,
      "source":"200.1.3.1",
      "key":"temp",
      "public_key":"7D61BA2FA556FD7B4AA0D54114575DF6FBC5AB9B96337C4A438E85CDFC77ED7C",
      "port":5100,
      "peers":[
        {
          "name":"n3",
          "public_key":
↪"950A6657CDE2193C786FF4771A46318AB86B9CB60BA071E344E8C094EBEEF662",
          "endpoint":"0.0.0.0",
          "keepalive":25,
          "port":0,
          "allowed_ips":[
            "3.1.1.0/24"
          ]
        }
      ]
    },
    {
      "instance":20,
      "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,
          "allowed_ips":[
            "2.1.1.0/24"
          ]
        }
      ]
```

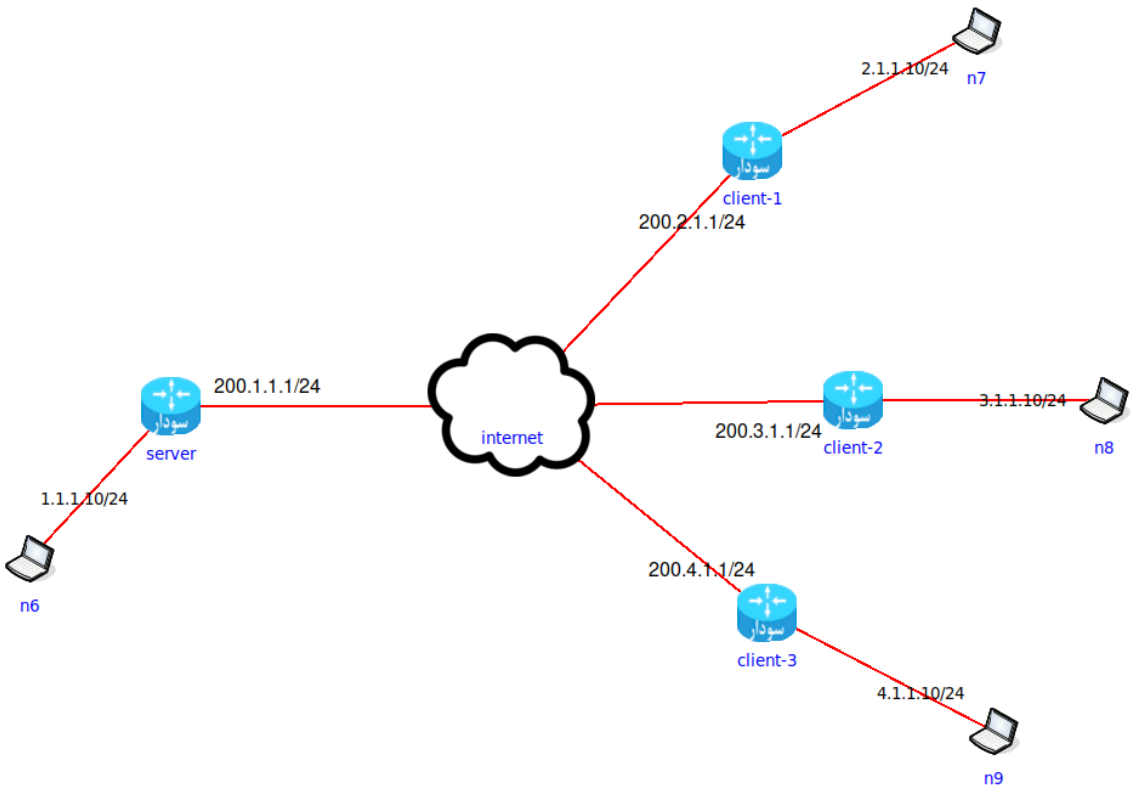
(continues on next page)

(continued from previous page)

```
    }
  ]
}
```

### 9.3.5 Example

In following scenario, we want to establish a wireguard server with 3 clients.




First we create a x25519 private key in 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 setup 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 setup 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)# port 51820
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
```

We do the same exact thing for remaining two clients.

Now that peers are set up, we add peer informations to 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 first packet from each client to `1.1.1.0/24` the tunnel establishes.

## 9.4 IKEv2

IKEv2( Internet Key Exchange version 2) is a protocol that establishes and handles SAs( Security Association). Usually these SAs are used with IPSec( in fact, IKEv2 is based on IPSec).

Like IPSec, IKEv2 uses a modular CLI for configuration.

**`crypto ikev2 dpd (1-3600)`**
: Set IKEv2 dead peer detection parameters. First parameter indicates how often liveness check is performed.

### 9.4.1 IKEv2 proposal

A proposal, consists of a suite of encryption/ HMAC algorithms for IKEv2 SA creation. Along with IPSec transform set, they define the IKEv2/IPSec algorithm suite and it's the first thing that is negotiated in IKEv2.

**`crypto ikev2 proposal IKEPOSAL`**
: Create a new IKEV2 proposal named IKEPOSAL

**`encryption ALGORITHM`**
: Use ALGORITHM as encryption algorithm for IKEv2 SA creation and negotiation.

**integrity ALGORITHM**
Use ALGORITHM as HMAC algorithm for IKEv2 SA creation and negotiation.

**group GROUP**
Use GROUP as Diffie-Hellman group.

Example :

```
soodar(config)# crypto ikev2 proposal sample-proposal
soodar(config-ikev2-proposal)# encryption aes-192
soodar(config-ikev2-proposal)# integrity sha1-96
soodar(config-ikev2-proposal)# group 28
```

### 9.4.2 IKEv2 keyring

As the name suggests, this struct, holds identity of peers, theirs address and authentication materials.

**crypto ikev2 keyring IKEKEYRING**
Create a new IKEV2 keyring named IKEKEYRING

**peer PEER**
Create a new peer in keyring as PEER

**pre-shared-key LINE**
Set peer's PSK value for authentication

**identity address <A.B.C.D|X:X::X:X>**
Use an address as peer's identity. This identity is used in negotiation and by other peer to identify the other one.

**identity fqdn FQDN**
Use a FQDN as peer's identity. This identity is used in negotiation and by other peer to identify the other one.

**identity email MAIL**
Use a mail address as peer's identity. This identity is used in negotiation and by other peer to identify the other one.

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!!
```

### 9.4.3 IKEv2 profile

Main struct of IKEv2.

**match address local A.B.C.D**
Send IKEv2 packets using this address. if it's not set, the packets can use any IP address available on router.

---

**Note:** It's a good practice to set local address explicitly. It can prevent from problems caused by changes in routes, which could lead to a change in source IP address of packets and being rejected by IKEv2 peer.

---

**identity local address <A.B.C.D|X:X::X:X>**
Define an address as local identity. During IKEv2 session, introduce him by this identity to its peer.

**identity local fqdn FQDN**
Define a FQDN as local identity. During IKEv2 session, introduce him by this identity to its peer.

**identity local email MAIL**
Define a mail address as local identity. During IKEv2 session, introduce him by this identity to its peer.

**authentication local rsa-sig**
Use RSA Digital Signature as authentication method during init.

---

**Note:** The chosen certificate to use, is the one that has the same `SAN` as `local identity`

---

**authentication local pre-share**
Use Shared Key Message Integrity Code as authentication method during init.

**authentication remote rsa-sig**
Expect RSA Digital Signature from peer as authentication method during init.

**authentication remote pre-share**
Expect Shared Key Message Integrity Code from peer as authentication method during init.

**match identity remote address <A.B.C.D|X:X::X:X>**
Expected remote identity. Other information about this peer is looked up in keyring.

**match identity remote fqdn FQDN**
Expected remote identity. Other information about this peer is looked up in keyring.

**match identity remote email EMAIL**
Expected remote identity. Other information about this peer is looked up in keyring.

**match certificate**
Match against DN fields and values as peer identity. using wildcards are 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 ther `identity` and this certificate is issued from `Iran` and is a sub-domain of `temp.ir`

**keyring local IKEKEYRING**
Use keyring IKEKEYRING for peer lookup.

**proposal IKEPOSAL**
Use IKEPOSAL for IKEv2 SA.

**lifetime <120-86400>**
Set IKEv2 profile lifetime. Cause a rekey action to take place when the life timer exceeds.

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
```

(continues on next page)

(continued from previous page)

```
soodar(config-ikev2-profile)# keyring local keyring-1
soodar(config-ikev2-profile)# proposal sample-proposal
```

## 9.5 IPSec

### 9.5.1 Transform set

Define authentication and encryption algorithms

**crypto ipsec transform-set IPSECTS ah hmac HMAC_ALG**
Use AH mode

**crypto ipsec transfor-set IPSECTS  esp {hmac HMAC_ALG |cipher CIPHER_ALG}**
Use ESP mode.

**mode transport**
In transport mode, Original IP header is not changed. only AH or ESP header is added.

### 9.5.2 Profile

Store encryption/decryption keys.

There are two ways of establishing SAs for IPSec: using static pre-defined keys and using IKEv2 profiles. One should choose just one method and using 2 methods combined, yields inconsistency.

**[no] crypto ipsec profile IPSECPROFILE**
Create a new profile IPSECPROFILE.

**[no] set transform-set IPSECTS**

> **Note:** An IPSec profile without Transform set is useless.

**[no] set ikev2 profile IKEPROFILE**
Set profile's IKEv2 profile to establish an IKEv2 session and SAs.

**Note:** An IKEv2 profile should be exclusive to one profile. and a profile using 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)**
When using IKEv2, set SA rekeying criterion as time.

**Note:** IPSec SAs are just installed when IPSec profile is protecting a tunnel.

### 9.5.3 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)# address 200.1.2.1
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
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 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
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
```

### 9.5.4 Example: Setup an IPSec profile using IKEv2 and RSA-Sig

We have 2 routers, `soodar1` and `soodar2`. On each of them we have a valid CA and a signed-certificate for authentication. `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
```

(continues on next page)

(continued from previous page)

```
soodar1(ipsec-profile)#  set transform-set ipsec-tunnel-TS
soodar1(ipsec-profile)#  set ikev2 profile profile-ike
```

and in 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
```

### 9.5.5 Logging

Debugging logs can be set in case of need.

**[no] debug ipsec event**
: log data plane installation processes and results

**[no] debug ipsec vici json**
: log all incoming VICI messages as json

**[no] debug ipsec vici detail**
: log all incoming VICI messages as json and raw

CHAPTER

TEN

# L2 FEATURES

## 10.1 L2 Abilities

### 10.1.1 VLAN

VLAN allows user to segment a LAN into different broadcast domains.

#### Subinterfaces

To create a VLAN, first we need a subinterface

**interface IFNAME.(0-4095)**
Creates a subinterface on interface IFNAME.

Example :

```
soodar(config)# interface ge1.100
```

**encapsulation dot1q (1-4094) [exact] [second-dot1q (1-4094)]**
Encapsulate packets with one( or two, if specified) VLAN tags. Drop the input packets that does not have the same tag. If `exact` is specified, input packets must have the same number of VLAN tags as the configuration.

---

**Note:** Although we can add two tags, but 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)**
Use Q-in-Q encapsulation mode to add two tags.

**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
```

(continues on next page)

(continued from previous page)

```
soodar(config-if)# interface ge1.200
soodar(config-if)# encapsulation default
```

#### Tag rewrite

An interface can be set up in a way to add or remove( and in future, translate) VLAN tags.

**[no] rewrite tag push <1|2> <dot1q|dot1ad> (0-4095) [(0-4095)]**
Push 1 or 2 tags to ingress traffic. The `no` form, negate all changes caused by this command.

> **Note:** if `dot1ad` is used, only the first tag is dot1ad and second tag is dot1q

**[no] rewrite tag pop <1|2>**
Pop 1 or 2 tags from ingress traffic. The `no` form, negate all changes caused by this command.

Example :

```
soodar(config)# int ge2
soodar(config-if)# rewrite tag push 1 dot1q 300
```

### 10.1.2 Bridge

Bridge is a tool for connecting two segment of LAN to each other.

#### 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 to interface, if destination MAC address is not available in bridge-group MAC table, the packet is flooded to all interfaces in 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 interface is not in any split-horizon group and is default value for SHG.

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
BD-ID   Index   BSN  Age(min)  Learning  U-Forwrd   UU-Flood   Flooding  ARP-
↪Term  arp-ufwd   BVI-Intf
   600      1      0     off        on        on       flood        on
↪off       off         N/A
```

(continues on next page)

(continued from previous page)

```
        Interface            If-idx ISN  SHG  BVI  TxFlood        VLAN-Tag-
↪Rewrite
           ge2                  3     1    2    -      *                 none
```

### 10.1.3 SPAN

Port monitoring tool. It mirrors source interfaces input, to destination interface.

#### Configuration

Each monitor session can have multiple source interfaces. But only one one destination interface.

**[no] monitor session (1-66) source interface INTERFACE [both|rx|tx]**
Add interface INTERFACE to session. By default, both traffics are mirrored. But one can change the behaviour by implying rx,tx or both. The `no` form, remove an interface from monitor session.

---

**Note:** A session is not established unless a valid destionation is available.

---

**monitor session (1-66) destination interface INTERFACE**
Add interface INTERFACE as session's destination port

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
```

---

**Note:** Since the exact packet is mirrored on port( without changing anything), it's important the interface in receiver side be in promiscuous mode

---

### 10.1.4 Logging

Debugging logs can be set in case of need.

**[no] debug vlan event**
log data plane installation processes and results

**[no] debug bridge event**
log data plane installation processes and results

**[no] debug span event**
log data plane installation processes and results

## 10.2 LACP

Link Aggregation Control Protocol is a L2-layer protocol to aggregate two or more interfaces traffic. It also introduces link redundancy.

### 10.2.1 Bundle interface

As the name suggests, it's an interface that is the result of bonding interfaces.

**interface bundle-ether (1-65535)**
Create a bundle-ether interface with given bundle id

**set mode <rr|xor|active-backup|broadcast|lacp> <l2|l23|l34>**
Set bundle-ether interface action mode. First input determines bonding algorithm and second one, is load-balancing algorithm. LB is available on LACP and XOR bonding algorithms.

`rr` is *round-robin*. Packets are sent through slave interfaces, in a round-robin manner.

in `active-backup` mode, all packets are sent through first available interface and the others are reserved.

`boradcast` mode sends all packets through all slave interfaces.

`xor` and `lacp` mode are the same. Packets are sent via different slaves based on determined flows.

`l2` load balancing, classify packet to flows by their source and destination MAC addresses.

`l23` load balancing, classify packet to flows by their source and destination MAC addresses and source and destinaion IP addresses.

`l34` load balancing, classify packet to flows by their ssource and destinaion IP addresses, protocol and if available, TCP/UDP source and destination port number.

Default values are LACP using Layer3-Layer4 load balancing.

### 10.2.2 Enslave an interface

**[no] bundle id (1-65535)**
Enslave an interface to the bundle interface with given ID. The `no` form, detach an interface from 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
```

(continues on next page)

(continued from previous page)

```
soodar(config-if)# no bridge-group 100
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
```

### 10.2.3 Logging

Debugging logs can be set in case of need.

**[no] debug bond event**
log data plane installation processes and results

# 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

# INDEX

## Symbols

## A

## B



## C

## D

## K



## L

## M



## N

## O

## P



## R

## S

## T



## U



## V



## W