FRR User Manual

Release latest

FRR

Sep 23, 2023

CONTENTS

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

Clear counter of ip prefix-list

`clear ip prefix-list [NAME [A.B.C.D/M]]`
Clears the counters of all IP prefix lists. Clear IP Prefix List can be used with a specified NAME or NAME and prefix.

2.6 Route Maps

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

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

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

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

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

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

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

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

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

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

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

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

To summarise the above:

	Match	No Match
Permit	action	cont
Deny	deny	cont

action

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

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

cont goto next route-map entry

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

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

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

2.6.1 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*.

2.6.2 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 ACCESS_LIST
Match the next-hop according to the given access-list.

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

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

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

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

match ipv6 next-hop prefix-list PREFIX_LIST
Match the next-hop according to the given prefix-list.

match as-path AS_PATH
Matches the specified *as_path*.

match metric METRIC
Matches the specified *metric*.

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

match local-preference METRIC
Matches the specified *local-preference*.

match community COMMUNITY_LIST
Matches the specified *community_list*

match peer IPV4_ADDR
This is a BGP specific match command. Matches the peer ip address if the neighbor was specified in this manner.

match peer IPV6_ADDR
This is a BGP specific match command. Matches the peer ipv6 address if the neighbor was specified in this manner.

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

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

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

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

match evpn route-type ROUTE_TYPE_NAME
This is a BGP EVPN specific match command. It matches to EVPN route-type from type-1 (EAD route-type) to type-5 (Prefix route-type). User can provide in an integral form (1-5) or string form of route-type (i.e ead, macip, multicast, es, prefix).

match evpn vni NUMBER
This is a BGP EVPN specific match command which matches to EVPN VNI id. The number is a range from (1-6777215).

2.6.3 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. As a note realms cannot currently be used with the installation of nexthops as nexthop groups in the linux kernel.

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*.

set distance (1-255)
Set the Administrative distance to use for the route. This is only locally significant and will not be dispersed to peers.

set weight WEIGHT
Set the route's weight.

set metric <[+|-](1-4294967295)|rtt|+rtt|-rtt>
Set the route metric. When used with BGP, set the BGP attribute MED to a specific value. Use +/- to add or subtract the specified value to/from the existing/MED. Use *rtt* to set the MED to the round trip time or +*rtt*/-*rtt* to add/subtract the round trip time to/from the MED.

set min-metric <(0-4294967295)>
Set the minimum meric for the route.

set max-metric <(0-4294967295)>
Set the maximum meric for the route.

set aigp-metric <igp-metric|(1-4294967295)>
Set the BGP attribute AIGP to a specific value. If `igp-metric` is specified, then the value is taken from the IGP protocol, otherwise an arbitrary value.

set as-path prepend AS_PATH
Set the BGP AS path to prepend.

set as-path exclude AS-NUMBER...
Drop AS-NUMBER from the BGP AS path.

set community COMMUNITY
Set the BGP community attribute.

set ipv6 next-hop local IPV6_ADDRESS
Set the BGP-4+ link local IPv6 nexthop address.

set origin ORIGIN <egp|igp|incomplete>
Set BGP route origin.

set table (1-4294967295)
Set the BGP table to a given table identifier

set sr-te color (1-4294967295)
Set the color of a SR-TE Policy to be applied to a learned route. The SR-TE Policy is uniquely determined by the color and the BGP nexthop.

set l3vpn next-hop encapsulation gre
Accept L3VPN traffic over GRE encapsulation.

2.6.4 Route Map Call Command

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

2.6.5 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

2.6.6 Route Map Optimization Command

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

2.6.7 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.

2.7 Affinity Maps

Affinity maps provide a means of configuring Standard Admininistrative-Group (RFC3630, RFC5305 and RFC5329) and Extended Admininistrative-Group (RFC7308). An affinity-map maps a specific bit position to a human readable-name.

An affinity refers to a color or a ressource class in the Traffic Engineering terminology. The bit position means the position of the bit set starting from the least significant bit. For example, if the affinity 'blue' has bit position 0 the extended Admin-Group value will be 0x01. If the affinity 'red' bit position 2 was added to a link in combination with the 'blue' affinity, the Admin-Group value would be 0x05.

2.7.1 Command

affinity-map NAME bit-position (0-1023)
: Map the affinity name NAME to the bit-position. The bit-position is the key so that only one name can be mapped to particular bit-position.

no affinity-map NAME
: Remove the affinity-map mapping.

Affinity-maps with a bit-position value higher than 31 are not compatible with Standard Admininistrative-Group. The CLI disallow the usage of such affinity-maps when Standard Admininistrative-Groups are required.

2.8 IPv6 Support

FRR fully supports IPv6 routing. As described so far, FRR supports RIPng, OSPFv3, and BGP-4+. You can give IPv6 addresses to an interface and configure static IPv6 routing information. FRR 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 FRR could be built without IPv6 support. This is no longer possible.

2.8.1 Router Advertisement

show ipv6 nd ra-interfaces [vrf <VRFNAME|all>]
Show configured route advertisement interfaces. VRF subcommand only applicable for netns-based vrfs.

ipv6 nd suppress-ra
Don't send router advertisement messages. The no form of this command enables sending RA messages.

ipv6 nd prefix ipv6prefix [valid-lifetime] [preferred-lifetime] [off-link] [no-autoconfig] [router-addr
Configuring the IPv6 prefix to include in router advertisements. Several prefix specific optional parameters and flags may follow:

- valid-lifetime: the length of time in seconds during what the prefix is valid for the purpose of on-link determination. Value infinite represents infinity (i.e. a value of all one bits (0xffffffff)). Range: (0-4294967295) Default: 2592000
- preferred-lifetime: the length of time in seconds during what addresses generated from the prefix remain preferred. Value infinite represents infinity. Range: (0-4294967295) Default: 604800
- off-link: indicates that advertisement makes no statement about on-link or off-link properties of the prefix. Default: not set, i.e. this prefix can be used for on-link determination.
- no-autoconfig: indicates to hosts on the local link that the specified prefix cannot be used for IPv6 autoconfiguration.

  Default: not set, i.e. prefix can be used for autoconfiguration.
- router-address: indicates to hosts on the local link that the specified prefix contains a complete IP address by setting R flag.

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

ipv6 nd ra-interval [(1-1800)]
The maximum time allowed between sending unsolicited multicast router advertisements from the interface, in seconds. Default: 600

ipv6 nd ra-interval [msec (70-1800000)]
The maximum time allowed between sending unsolicited multicast router advertisements from the interface, in milliseconds. Default: 600000

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

ipv6 nd ra-retrans-interval [(0-4294967295)]
The value to be placed in the retrans timer field of router advertisements sent from the interface, in msec. Indicates the interval between router advertisement retransmissions. Setting the value to zero indicates that the value is unspecified by this router. Must be between zero or 4294967295 msec. Default: 0

ipv6 nd ra-hop-limit [(0-255)]
The value to be placed in the hop count field of router advertisements sent from the interface, in hops. Indicates the maximum diameter of the network. Setting the value to zero indicates that the value is unspecified by this router. Must be between zero or 255 hops. Default: 64

ipv6 nd ra-lifetime [(0-9000)]
The value to be placed in the Router Lifetime field of router advertisements sent from the interface, in seconds. Indicates the usefulness of the router as a default router on this interface. Setting the value to zero indicates that the router should not be considered a default router on this interface. Must be either zero or between value specified with ipv6 nd ra-interval (or default) and 9000 seconds. Default: 1800

ipv6 nd reachable-time [(1-3600000)]
The value to be placed in the Reachable Time field in the Router Advertisement messages sent by the router, in milliseconds. The configured time enables the router to detect unavailable neighbors. The value zero means unspecified (by this router). Default: `0`

ipv6 nd managed-config-flag
Set/unset flag in IPv6 router advertisements which indicates to hosts that they should use managed (stateful) protocol for addresses autoconfiguration in addition to any addresses autoconfigured using stateless address autoconfiguration. Default: not set

ipv6 nd other-config-flag
Set/unset flag in IPv6 router advertisements which indicates to hosts that they should use administered (stateful) protocol to obtain autoconfiguration information other than addresses. Default: not set

ipv6 nd home-agent-config-flag
Set/unset flag in IPv6 router advertisements which indicates to hosts that the router acts as a Home Agent and includes a Home Agent Option. Default: not set

ipv6 nd home-agent-preference [(0-65535)]
The value to be placed in Home Agent Option, when Home Agent config flag is set, which indicates to hosts Home Agent preference. The default value of 0 stands for the lowest preference possible. Default: `0`

ipv6 nd home-agent-lifetime [(0-65520)]
The value to be placed in Home Agent Option, when Home Agent config flag is set, which indicates to hosts Home Agent Lifetime. The default value of 0 means to place the current Router Lifetime value.

Default: `0`

ipv6 nd adv-interval-option
Include an Advertisement Interval option which indicates to hosts the maximum time, in milliseconds, between successive unsolicited Router Advertisements. Default: not set

ipv6 nd router-preference [(high|medium|low)]
Set default router preference in IPv6 router advertisements per RFC4191. Default: medium

ipv6 nd mtu [(1-65535)]
Include an MTU (type 5) option in each RA packet to assist the attached hosts in proper interface configuration. The announced value is not verified to be consistent with router interface MTU.

Default: don't advertise any MTU option.

ipv6 nd rdnss ipv6address [lifetime]
Recursive DNS server address to advertise using the RDNSS (type 25) option described in RFC8106. Can be specified more than once to advertise multiple addresses. Note that hosts may choose to limit the number of RDNSS addresses to track.

Optional parameter:

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

Default: do not emit RDNSS option

ipv6 nd dnssl domain-name-suffix [lifetime]
Advertise DNS search list using the DNSSL (type 31) option described in RFC8106. Specify more than once to advertise multiple domain name suffixes. Host implementations may limit the number of honored search list entries.

Optional parameter:

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

Default: do not emit DNSSL option

2.8.2 Router Advertisement Configuration Example

A small example:

```
interface eth0
 no ipv6 nd suppress-ra
 ipv6 nd prefix 2001:0DB8:5009::/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)

2.9 Kernel Interface

There are several different methods for reading kernel routing table information, updating kernel routing tables, and for looking up interfaces. FRR relies heavily on the Netlink (`man 7 netlink`) interface to communicate with the Kernel. However, other interfaces are still used in some parts of the code.

- **ioctl** This method is a very traditional way for reading or writing kernel information. *ioctl* can be used for looking up interfaces and for modifying interface addresses, flags, mtu settings and other types of information. Also, *ioctl* can insert and delete kernel routing table entries. It will soon be available on almost any platform which zebra supports, but it is a little bit ugly thus far, so if a better method is supported by the kernel, zebra will use that.
- **sysctl** This is a program that can lookup kernel information using MIB (Management Information Base) syntax. Normally, it only provides a way of getting information from the kernel. So one would usually want to change kernel information using another method such as *ioctl*.
- **proc filesystem** This is a special filesystem mount that provides an easy way of getting kernel information.
- **routing socket / Netlink** Netlink first appeard in Linux kernel 2.0. It makes asynchronous communication between the kernel and FRR possible, similar to a routing socket on BSD systems. Netlink communication is done by reading/writing over Netlink socket.

const struct nh_grp

```
* integer id
* integer weight
```

Zebra Hook calls

on_rib_process_dplane_results

Called when RIB processes dataplane events. Set script location with the `zebra on-rib-process script SCRIPT` command.

Arguments

- *const struct zebra_dplane_ctx* ctx

```
function on_rib_process_dplane_results(ctx)
   log.info(ctx.rinfo.zd_dest.network)
   return {}
```

3.2 Bidirectional Forwarding Detection

BFD (Bidirectional Forwarding Detection) stands for Bidirectional Forwarding Detection and it is described and extended by the following RFCs:

- **RFC 5880**
- **RFC 5881**
- **RFC 5882**
- **RFC 5883**

Currently, there are two implementations of the BFD commands in FRR:

- PTM (Prescriptive Topology Manager): an external daemon which implements BFD;
- `bfdd`: a BFD implementation that is able to talk with remote peers;

This document will focus on the later implementation: *bfdd*.

3.2.1 Starting BFD

bfdd default configuration file is `bfdd.conf`. *bfdd* searches the current directory first then /etc/frr/bfdd.conf. All of *bfdd*'s command must be configured in `bfdd.conf`.

bfdd specific invocation options are described below. Common options may also be specified (*Common Invocation Options*).

--bfdctl <unix-socket>
: Set the BFD daemon control socket location. If using a non-default socket location:

  ```
  /usr/lib/frr/bfdd --bfdctl /tmp/bfdd.sock
  ```

  The default UNIX socket location is:

#define BFDD_CONTROL_SOCKET "/var/run/frr/bfdd.sock"

This option overrides the location addition that the -N option provides to the bfdd.sock

--dplaneaddr <type>:<address>[<:port>]
Configure the distributed BFD data plane listening socket bind address.

One would expect the data plane to run in the same machine as FRR, so the suggested configuration would be:

–dplaneaddr unix:/var/run/frr/bfdd_dplane.sock

Or using IPv4:

–dplaneaddr ipv4:127.0.0.1

Or using IPv6:

–dplaneaddr ipv6:[::1]

It is also possible to specify a port (for IPv4/IPv6 only):

–dplaneaddr ipv6:[::1]:50701

(if ommited the default port is **50700**).

It is also possible to operate in client mode (instead of listening for connections). To connect to a data plane server append the letter 'c' to the protocol, example:

–dplaneaddr ipv4c:127.0.0.1

Note: When using UNIX sockets don't forget to check the file permissions before attempting to use it.

3.2.2 BFDd Commands

bfd
Opens the BFD daemon configuration node.

peer <A.B.C.D|X:X::X:X> [{multihop|local-address <A.B.C.D|X:X::X:X>|interface IFNAME|vrf NAME}]
Creates and configures a new BFD peer to listen and talk to.

multihop tells the BFD daemon that we should expect packets with TTL less than 254 (because it will take more than one hop) and to listen on the multihop port (4784). When using multi-hop mode *echo-mode* will not work (see **RFC 5883** section 3).

local-address provides a local address that we should bind our peer listener to and the address we should use to send the packets. This option is mandatory for IPv6.

interface selects which interface we should use.

vrf selects which domain we want to use.

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

Deleting the profile will cause all peers using it to reset to the default values.

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

show bfd [vrf NAME] peer <WORD|<A.B.C.D|X:X::X:X> [{multihop|local-address <A.B.C.D|X:X::X:X>|interface IFNAME}]> [json]
Show status for a specific BFD peer.

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

show bfd distributed
Show the BFD data plane (distributed BFD) statistics.

Peer / Profile Configuration

BFD peers and profiles share the same BFD session configuration commands.

detect-multiplier (2-255)
Configures the detection multiplier to determine packet loss. The remote transmission interval will be multiplied by this value to determine the connection loss detection timer. The default value is 3.

Example: when the local system has *detect-multiplier 3* and the remote system has *transmission interval 300*, the local system will detect failures only after 900 milliseconds without receiving packets.

receive-interval (10-60000)
Configures the minimum interval that this system is capable of receiving control packets. The default value is 300 milliseconds.

transmit-interval (10-60000)
The minimum transmission interval (less jitter) that this system wants to use to send BFD control packets. Defaults to 300ms.

echo receive-interval <disabled|(10-60000)>
Configures the minimum interval that this system is capable of receiving echo packets. Disabled means that this system doesn't want to receive echo packets. The default value is 50 milliseconds.

echo transmit-interval (10-60000)
The minimum transmission interval (less jitter) that this system wants to use to send BFD echo packets. Defaults to 50ms.

echo-mode
Enables or disables the echo transmission mode. This mode is disabled by default. If you are not using distributed BFD then echo mode works only when the peer is also FRR.

It is recommended that the transmission interval of control packets to be increased after enabling echo-mode to reduce bandwidth usage. For example: *transmit-interval 2000*.

Echo mode is not supported on multi-hop setups (see **RFC 5883** section 3).

shutdown
Enables or disables the peer. When the peer is disabled an 'administrative down' message is sent to the remote peer.

passive-mode
Mark session as passive: a passive session will not attempt to start the connection and will wait for control packets from peer before it begins replying.

This feature is useful when you have a router that acts as the central node of a star network and you want to avoid sending BFD control packets you don't need to.

The default is active-mode (or `no passive-mode`).

minimum-ttl (1-254)
For multi hop sessions only: configure the minimum expected TTL for an incoming BFD control packet.

This feature serves the purpose of thightening the packet validation requirements to avoid receiving BFD control packets from other sessions.

The default value is 254 (which means we only expect one hop between this system and the peer).

BFD Peer Specific Commands

label WORD
Labels a peer with the provided word. This word can be referenced later on other daemons to refer to a specific peer.

profile BFDPROF
Configure peer to use the profile configurations.

Notes:

- Profile configurations can be overridden on a peer basis by specifying non-default parameters in peer configuration node.
- Non existing profiles can be configured and they will only be applied once they start to exist.
- If the profile gets updated the new configuration will be applied to all peers with the profile without interruptions.

BGP BFD Configuration

The following commands are available inside the BGP configuration node.

neighbor <A.B.C.D|X:X::X:X|WORD> bfd
Listen for BFD events registered on the same target as this BGP neighbor. When BFD peer goes down it immediately asks BGP to shutdown the connection with its neighbor and, when it goes back up, notify BGP to try to connect to it.

neighbor <A.B.C.D|X:X::X:X|WORD> bfd check-control-plane-failure
Allow to write CBIT independence in BFD outgoing packets. Also allow to read both C-BIT value of BFD and lookup BGP peer status. This command is useful when a BFD down event is caught, while the BGP peer requested that local BGP keeps the remote BGP entries as staled if such issue is detected. This is the case when graceful restart is enabled, and it is wished to ignore the BD event while waiting for the remote router to restart.

Disabling this disables presence of CBIT independence in BFD outgoing packets and pays attention to BFD down notifications. This is the default.

neighbor <A.B.C.D|X:X::X:X|WORD> bfd profile BFDPROF
Same as command `neighbor <A.B.C.D|X:X::X:X|WORD> bfd`, but applies the BFD profile to the sessions it creates or that already exist.

IS-IS BFD Configuration

The following commands are available inside the interface configuration node.

isis bfd
Listen for BFD events on peers created on the interface. Every time a new neighbor is found a BFD peer is created to monitor the link status for fast convergence.

Note that there will be just one BFD session per interface. In case both IPv4 and IPv6 support are configured then just a IPv6 based session is created.

isis bfd profile BFDPROF
Use a BFD profile BFDPROF as provided in the BFD configuration.

OSPF BFD Configuration

The following commands are available inside the interface configuration node.

ip ospf bfd
: Listen for BFD events on peers created on the interface. Every time a new neighbor is found a BFD peer is created to monitor the link status for fast convergence.

ip ospf bfd profile BFDPROF
: Same as command `ip ospf bfd`, but applies the BFD profile to the sessions it creates or that already exist.

OSPF6 BFD Configuration

The following commands are available inside the interface configuration node.

ipv6 ospf6 bfd [profile BFDPROF]
: Listen for BFD events on peers created on the interface. Every time a new neighbor is found a BFD peer is created to monitor the link status for fast convergence.

  Optionally uses the BFD profile `BFDPROF` in the created sessions under that interface.

PIM BFD Configuration

The following commands are available inside the interface configuration node.

ip pim bfd [profile BFDPROF]
: Listen for BFD events on peers created on the interface. Every time a new neighbor is found a BFD peer is created to monitor the link status for fast convergence.

  Optionally uses the BFD profile `BFDPROF` in the created sessions under that interface.

RIP BFD configuration

The following commands are available inside the interface configuration node:

ip rip bfd
: Automatically create BFD session for each RIP peer discovered in this interface. When the BFD session monitor signalize that the link is down the RIP peer is removed and all the learned routes associated with that peer are removed.

ip rip bfd profile BFD_PROFILE_NAME
: Selects a BFD profile for the BFD sessions created in this interface.

The following command is available in the RIP router configuration node:

bfd default-profile BFD_PROFILE_NAME
: Selects a default BFD profile for all sessions without a profile specified.

BFD Static Route Monitoring Configuration

A monitored static route conditions the installation to the RIB on the BFD session running state: when BFD session is up the route is installed to RIB, but when the BFD session is down it is removed from the RIB.

The following commands are available inside the configuration node:

ip route A.B.C.D/M A.B.C.D bfd [{multi-hop|source A.B.C.D|profile BFDPROF}]
Configure a static route for `A.B.C.D/M` using gateway `A.B.C.D` and use the gateway address as BFD peer destination address.

ipv6 route X:X::X:X/M [from X:X::X:X/M] X:X::X:X bfd [{multi-hop|source X:X::X:X|profile BFDPROF}]
Configure a static route for `X:X::X:X/M` using gateway `X:X::X:X` and use the gateway address as BFD peer destination address.

The static routes when uninstalled will no longer show up in the output of the command `show ip route` or `show ipv6 route`, instead we must use the BFD static route show command to see these monitored route status.

show bfd static route [json]
Show all monitored static routes and their status.

Example output:

```
Showing BFD monitored static routes:

  Route groups:
    rtg1 peer 172.16.0.1 (status: uninstalled):
        2001:db8::100/128

Next hops:
  VRF default IPv4 Unicast:
      192.168.100.0/24 peer 172.16.0.1 (status: uninstalled)

  VRF default IPv4 Multicast:

  VRF default IPv6 Unicast:
```

3.2.3 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
```

(continues on next page)

(continued from previous page)

```
 neighbor 192.168.0.3 remote-as 65532
 neighbor 192.168.0.3 bfd
!
```

Peers can be identified by its address (use `multihop` when you need to specify a multi hop peer) or can be specified manually by a label.

Here are the available peer configurations:

```
bfd
 ! Configure a fast profile
 profile fast
  receive-interval 150
  transmit-interval 150
 !

 ! Configure peer with fast profile
 peer 192.168.0.6
  profile fast
  no shutdown
 !

! Configure peer with fast profile and override receive speed.
 peer 192.168.0.7
  profile fast
  receive-interval 500
  no shutdown
 !

 ! configure a peer on an specific interface
 peer 192.168.0.1 interface eth0
  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
 !
```

(continues on next page)

Activates the following debug messages:

- Data plane received / send messages
- Connection events

debug bfd network
Toggle network events: show messages about socket failures and unexpected BFD messages that may not belong to registered peers.

debug bfd peer
Toggle peer event log messages: show messages about peer creation/removal and state changes.

debug bfd zebra
Toggle zebra message events: show messages about interfaces, local addresses, VRF and daemon peer registrations.

3.3 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.

3.3.1 Starting BGP

The default configuration file of *bgpd* is `bgpd.conf`. *bgpd* searches the current directory first, followed by /etc/frr/bgpd.conf. All of *bgpd*'s commands must be configured in `bgpd.conf` when the integrated config is not being used.

bgpd specific invocation options are described below. Common options may also be specified (*Common Invocation Options*).

-p, --bgp_port <port>
Set the bgp protocol's port number. When port number is 0, that means do not listen bgp port.

-l, --listenon
Specify specific IP addresses for bgpd to listen on, rather than its default of `0.0.0.0` / `::`. This can be useful to constrain bgpd to an internal address, or to run multiple bgpd processes on one host. Multiple addresses can be specified.

In the following example, bgpd is started listening for connections on the addresses 100.0.1.2 and fd00::2:2. The options -d (runs in daemon mode) and -f (uses specific configuration file) are also used in this example as we are likely to run multiple bgpd instances, each one with different configurations, when using -l option.

Note that this option implies the –no_kernel option, and no learned routes will be installed into the linux kernel.

```
# /usr/lib/frr/bgpd -d -f /some-folder/bgpd.conf -l 100.0.1.2 -l fd00::2:2
```

-n, --no_kernel
Do not install learned routes into the linux kernel. This option is useful for a route-reflector environment or if you are running multiple bgp processes in the same namespace. This option is different than the –no_zebra option in that a ZAPI connection is made.

This option can also be toggled during runtime by using the `[no] bgp no-rib` commands in VTY shell.

Note that this option will persist after saving the configuration during runtime, unless unset by the `no bgp no-rib` command in VTY shell prior to a configuration write operation.

-S, --skip_runas
Skip the normal process of checking capabilities and changing user and group information.

-e, --ecmp
Run BGP with a limited ecmp capability, that is different than what BGP was compiled with. The value specified must be greater than 0 and less than or equal to the MULTIPATH_NUM specified on compilation.

-Z, --no_zebra
Do not communicate with zebra at all. This is different than the –no_kernel option in that we do not even open a ZAPI connection to the zebra process.

-s, --socket_size
When opening tcp connections to our peers, set the socket send buffer size that the kernel will use for the peers socket. This option is only really useful at a very large scale. Experimentation should be done to see if this is helping or not at the scale you are running at.

LABEL MANAGER

-I, --int_num
Set zclient id. This is required when using Zebra label manager in proxy mode.

3.3.2 Basic Concepts

Autonomous Systems

From **RFC 1930**:

> An AS is a connected group of one or more IP prefixes run by one or more network operators which has a SINGLE and CLEARLY DEFINED routing policy.

Each AS has an identifying number associated with it called an ASN (Autonomous System Number). This is a two octet value ranging in value from 1 to 65535. The AS numbers 64512 through 65535 are defined as private AS numbers. Private AS numbers must not be advertised on the global Internet.

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

See also:

RFC 1930

Address Families

Multiprotocol extensions enable BGP to carry routing information for multiple network layer protocols. BGP supports an Address Family Identifier (AFI) for IPv4 and IPv6. Support is also provided for multiple sets of per-AFI information via the BGP Subsequent Address Family Identifier (SAFI). FRR supports SAFIs for unicast information, labeled information (**RFC 3107** and **RFC 8277**), and Layer 3 VPN information (**RFC 4364** and **RFC 4659**).

Route Selection

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

1. **Weight check**

   Prefer higher local weight routes to lower routes.

2. **Local preference check**

   Prefer higher local preference routes to lower.

   If `bgp bestpath aigp` is enabled, and both paths that are compared have AIGP attribute, BGP uses AIGP tie-breaking unless both of the paths have the AIGP metric attribute. This means that the AIGP attribute is not evaluated during the best path selection process between two paths when one path does not have the AIGP attribute.

3. **Local route check**

   Prefer local routes (statics, aggregates, redistributed) to received routes.

4. **AS path length check**

   Prefer shortest hop-count AS_PATHs.

5. **Origin check**

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

6. **MED check**

   Where routes with a MED were received from the same AS, prefer the route with the lowest MED. *Multi-Exit Discriminator*.

7. **External check**

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

8. **IGP cost check**

   Prefer the route with the lower IGP cost.

9. **Multi-path check**

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

10. **Already-selected external check**

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

11. **Router-ID check**

    Prefer the route with the lowest *router-ID*. If the route has an *ORIGINATOR_ID* attribute, through iBGP reflection, then that router ID is used, otherwise the *router-ID* of the peer the route was received from is used.

12. **Cluster-List length check**

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

13. **Peer address**

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

Capability Negotiation

When adding IPv6 routing information exchange feature to BGP. There were some proposals. IETF (Internet Engineering Task Force) IDR (Inter Domain Routing) adopted a proposal called Multiprotocol Extension for BGP. The specification is described in **RFC 2283**. The protocol does not define new protocols. It defines new attributes to existing BGP. When it is used exchanging IPv6 routing information it is called BGP-4+. When it is used for exchanging multicast routing information it is called MBGP.

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

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

By default, FRR will bring up peering with minimal common capability for the both sides. For example, if the local router has unicast and multicast capabilities and the remote router only has unicast capability the local router will establish the connection with unicast only capability. When there are no common capabilities, FRR sends Unsupported Capability error and then resets the connection.

3.3.3 BGP Router Configuration

ASN and Router ID

First of all you must configure BGP router with the *router bgp ASN* command. The AS number is an identifier for the autonomous system. The AS identifier can either be a number or two numbers separated by a period. The BGP protocol uses the AS identifier for detecting whether the BGP connection is internal or external.

router bgp ASN
: Enable a BGP protocol process with the specified ASN. After this statement you can input any *BGP Commands*.

bgp router-id A.B.C.D
: This command specifies the router-ID. If *bgpd* connects to *zebra* it gets interface and address information. In that case default router ID value is selected as the largest IP Address of the interfaces. When *router zebra* is not enabled *bgpd* can't get interface information so *router-id* is set to 0.0.0.0. So please set router-id by hand.

Multiple Autonomous Systems

FRR's BGP implementation is capable of running multiple autonomous systems at once. Each configured AS corresponds to a *Virtual Routing and Forwarding*. In the past, to get the same functionality the network administrator had to run a new *bgpd* process; using VRFs allows multiple autonomous systems to be handled in a single process.

When using multiple autonomous systems, all router config blocks after the first one must specify a VRF to be the target of BGP's route selection. This VRF must be unique within respect to all other VRFs being used for the same purpose, i.e. two different autonomous systems cannot use the same VRF. However, the same AS can be used with different VRFs.

Note: The separated nature of VRFs makes it possible to peer a single *bgpd* process to itself, on one machine. Note that this can be done fully within BGP without a corresponding VRF in the kernel or Zebra, which enables some practical use cases such as *route reflectors* and route servers.

Configuration of additional autonomous systems, or of a router that targets a specific VRF, is accomplished with the following command:

router bgp ASN vrf VRFNAME
`VRFNAME` is matched against VRFs configured in the kernel. When `vrf VRFNAME` is not specified, the BGP protocol process belongs to the default VRF.

An example configuration with multiple autonomous systems might look like this:

```
router bgp 1
 neighbor 10.0.0.1 remote-as 20
 neighbor 10.0.0.2 remote-as 30
!
router bgp 2 vrf blue
 neighbor 10.0.0.3 remote-as 40
 neighbor 10.0.0.4 remote-as 50
!
router bgp 3 vrf red
 neighbor 10.0.0.5 remote-as 60
 neighbor 10.0.0.6 remote-as 70
...
```

See also:

VRF Route Leaking

See also:

Virtual Routing and Forwarding

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
```

(continues on next page)

(continued from previous page)

```
neighbor 10.0.0.3 remote-as 4
neighbor 10.0.0.4 remote-as 5
```

show [ip] bgp view NAME
Display the routing table of BGP view NAME.

Route Selection

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

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

bgp bestpath compare-routerid
Ensure that when comparing routes where both are equal on most metrics, including local-pref, AS_PATH length, IGP cost, MED, that the tie is broken based on router-ID.

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

If a route has an *ORIGINATOR_ID* attribute because it has been reflected, that *ORIGINATOR_ID* will be used. Otherwise, the router-ID of the peer the route was received from will be used.

The advantage of this is that the route-selection (at this point) will be more deterministic. The disadvantage is that a few or even one lowest-ID router may attract all traffic to otherwise-equal paths because of this check. It may increase the possibility of MED or IGP oscillation, unless other measures were taken to avoid these. The exact behaviour will be sensitive to the iBGP and reflection topology.

bgp bestpath peer-type multipath-relax
This command specifies that BGP decision process should consider paths from all peers for multipath computation. If this option is enabled, paths learned from any of eBGP, iBGP, or confederation neighbors will be multipath if they are otherwise considered equal cost.

bgp bestpath aigp
Use the bgp bestpath aigp command to evaluate the AIGP attribute during the best path selection process between two paths that have the AIGP attribute.

When bgp bestpath aigp is disabled, BGP does not use AIGP tie-breaking rules unless paths have the AIGP attribute.

Disabled by default.

maximum-paths (1-128)
Sets the maximum-paths value used for ecmp calculations for this bgp instance in EBGP. The maximum value listed, 128, can be limited by the ecmp cli for bgp or if the daemon was compiled with a lower ecmp value. This value can also be set in ipv4/ipv6 unicast/labeled unicast to only affect those particular afi/safi's.

maximum-paths ibgp (1-128) [equal-cluster-length]
Sets the maximum-paths value used for ecmp calculations for this bgp instance in IBGP. The maximum value listed, 128, can be limited by the ecmp cli for bgp or if the daemon was compiled with a lower ecmp value. This value can also be set in ipv4/ipv6 unicast/labeled unicast to only affect those particular afi/safi's.

Administrative Distance Metrics

distance bgp (1-255) (1-255) (1-255)
This command changes distance value of BGP. The arguments are the distance values for external routes, internal routes and local routes respectively.

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

distance (1-255) A.B.C.D/M WORD
Sets the administrative distance for a particular route.

Require policy on EBGP

bgp ebgp-requires-policy
This command requires incoming and outgoing filters to be applied for eBGP sessions as part of RFC-8212 compliance. Without the incoming filter, no routes will be accepted. Without the outgoing filter, no routes will be announced.

This is enabled by default for the traditional configuration and turned off by default for datacenter configuration.

When you enable/disable this option you MUST clear the session.

When the incoming or outgoing filter is missing you will see "(Policy)" sign under `show bgp summary`:

```
exit1# show bgp summary

IPv4 Unicast Summary (VRF default):
BGP router identifier 10.10.10.1, local AS number 65001 vrf-id 0
BGP table version 4
RIB entries 7, using 1344 bytes of memory
Peers 2, using 43 KiB of memory

Neighbor        V         AS   MsgRcvd   MsgSent   TblVer  InQ OutQ  Up/Down State/
↪PfxRcd   PfxSnt Desc
192.168.0.2     4      65002         8        10        0    0    0 00:03:09       ␣
↪     5 (Policy) N/A
fe80:1::2222    4      65002         9        11        0    0    0 00:03:09     ␣
↪(Policy) (Policy) N/A
```

Additionally a *show bgp neighbor* command would indicate in the *For address family:* block that:

```
exit1# show bgp neighbor
...
For address family: IPv4 Unicast
 Update group 1, subgroup 1
 Packet Queue length 0
 Inbound soft reconfiguration allowed
 Community attribute sent to this neighbor(all)
 Inbound updates discarded due to missing policy
 Outbound updates discarded due to missing policy
 0 accepted prefixes
```

Reject routes with AS_SET or AS_CONFED_SET types

`bgp reject-as-sets`
This command enables rejection of incoming and outgoing routes having AS_SET or AS_CONFED_SET type.

Suppress duplicate updates

`bgp suppress-duplicates`
For example, BGP routers can generate multiple identical announcements with empty community attributes if stripped at egress. This is an undesired behavior. Suppress duplicate updates if the route actually not changed. Default: enabled.

Send Hard Reset CEASE Notification for Administrative Reset

`bgp hard-administrative-reset`
Send Hard Reset CEASE Notification for 'Administrative Reset' events.

When disabled, and Graceful Restart Notification capability is exchanged between the peers, Graceful Restart procedures apply, and routes will be retained.

Enabled by default.

Disable checking if nexthop is connected on EBGP sessions

`bgp disable-ebgp-connected-route-check`
This command is used to disable the connection verification process for EBGP peering sessions that are reachable by a single hop but are configured on a loopback interface or otherwise configured with a non-directly connected IP address.

Route Flap Dampening

`bgp dampening (1-45) (1-20000) (1-50000) (1-255)`
This command enables BGP route-flap dampening and specifies dampening parameters.

half-life Half-life time for the penalty

reuse-threshold Value to start reusing a route

suppress-threshold Value to start suppressing a route

max-suppress Maximum duration to suppress a stable route

The route-flap damping algorithm is compatible with **RFC 2439**. The use of this command is not recommended nowadays.

At the moment, route-flap dampening is not working per VRF and is working only for IPv4 unicast and multicast.

See also:

https://www.ripe.net/publications/docs/ripe-378

Multi-Exit Discriminator

The BGP MED (Multi-Exit Discriminator) attribute has properties which can cause subtle convergence problems in BGP. These properties and problems have proven to be hard to understand, at least historically, and may still not be widely understood. The following attempts to collect together and present what is known about MED, to help operators and FRR users in designing and configuring their networks.

The BGP MED attribute is intended to allow one AS to indicate its preferences for its ingress points to another AS. The MED attribute will not be propagated on to another AS by the receiving AS - it is 'non-transitive' in the BGP sense.

E.g., if AS X and AS Y have 2 different BGP peering points, then AS X might set a MED of 100 on routes advertised at one and a MED of 200 at the other. When AS Y selects between otherwise equal routes to or via AS X, AS Y should prefer to take the path via the lower MED peering of 100 with AS X. Setting the MED allows an AS to influence the routing taken to it within another, neighbouring AS.

In this use of MED it is not really meaningful to compare the MED value on routes where the next AS on the paths differs. E.g., if AS Y also had a route for some destination via AS Z in addition to the routes from AS X, and AS Z had also set a MED, it wouldn't make sense for AS Y to compare AS Z's MED values to those of AS X. The MED values have been set by different administrators, with different frames of reference.

The default behaviour of BGP therefore is to not compare MED values across routes received from different neighbouring ASes. In FRR this is done by comparing the neighbouring, left-most AS in the received AS_PATHs of the routes and only comparing MED if those are the same.

Unfortunately, this behaviour of MED, of sometimes being compared across routes and sometimes not, depending on the properties of those other routes, means MED can cause the order of preference over all the routes to be undefined. That is, given routes A, B, and C, if A is preferred to B, and B is preferred to C, then a well-defined order should mean the preference is transitive (in the sense of orders[1]) and that A would be preferred to C.

However, when MED is involved this need not be the case. With MED it is possible that C is actually preferred over A. So A is preferred to B, B is preferred to C, but C is preferred to A. This can be true even where BGP defines a deterministic 'most preferred' route out of the full set of A,B,C. With MED, for any given set of routes there may be a deterministically preferred route, but there need not be any way to arrange them into any order of preference. With unmodified MED, the order of preference of routes literally becomes undefined.

That MED can induce non-transitive preferences over routes can cause issues. Firstly, it may be perceived to cause routing table churn locally at speakers; secondly, and more seriously, it may cause routing instability in iBGP topologies, where sets of speakers continually oscillate between different paths.

The first issue arises from how speakers often implement routing decisions. Though BGP defines a selection process that will deterministically select the same route as best at any given speaker, even with MED, that process requires evaluating all routes together. For performance and ease of implementation reasons, many implementations evaluate route preferences in a pair-wise fashion instead. Given there is no well-defined order when MED is involved, the best route that will be chosen becomes subject to implementation details, such as the order the routes are stored in. That may be (locally) non-deterministic, e.g.: it may be the order the routes were received in.

This indeterminism may be considered undesirable, though it need not cause problems. It may mean additional routing churn is perceived, as sometimes more updates may be produced than at other times in reaction to some event .

This first issue can be fixed with a more deterministic route selection that ensures routes are ordered by the neighbouring AS during selection. `bgp deterministic-med`. This may reduce the number of updates as routes are received, and may in some cases reduce routing churn. Though, it could equally deterministically produce the largest possible set of updates in response to the most common sequence of received updates.

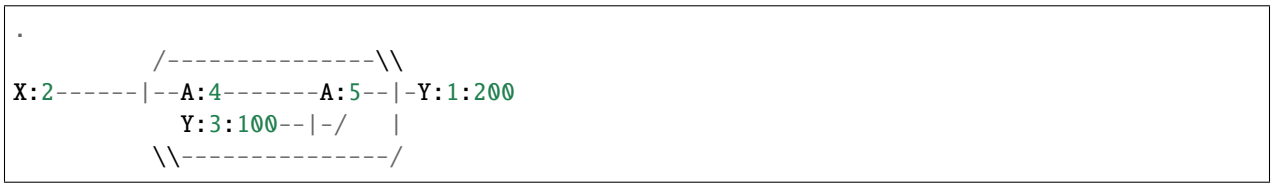
[1] For some set of objects to have an order, there *must* be some binary ordering relation that is defined for *every* combination of those objects, and that relation *must* be transitive. I.e.:, if the relation operator is <, and if a < b and b < c then that relation must carry over and it *must* be 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 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.

<---------------------------------------------------------------------

7. R1 was waiting for End-Of-Rib from R2 & which has been received
  now.

8. R1 now runs BGP Best-Path algorithm. Send Initial BGP  Update,
  followed by End-Of Rib

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

BGP-GR Preserve-Forwarding State

BGP OPEN message carrying optional capabilities for Graceful Restart has 8 bit "Flags for Address Family" for given AFI and SAFI. This field contains bit flags relating to routes that were advertised with the given AFI and SAFI.

```
0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
|F|   Reserved  |
+-+-+-+-+-+-+-+-+
```

The most significant bit is defined as the Forwarding State (F) bit, which can be used to indicate whether the forwarding state for routes that were advertised with the given AFI and SAFI has indeed been preserved during the previous BGP restart. When set (value 1), the bit indicates that the forwarding state has been preserved. The remaining bits are reserved and MUST be set to zero by the sender and ignored by the receiver.

bgp graceful-restart preserve-fw-state

FRR gives us the option to enable/disable the "F" flag using this specific vty command. However, it doesn't have the option to enable/disable this flag only for specific AFI/SAFI i.e. when this command is used, it applied to all the supported AFI/SAFI combinations for this peer.

End-of-RIB (EOR) message

An UPDATE message with no reachable Network Layer Reachability Information (NLRI) and empty withdrawn NLRI is specified as the End-of-RIB marker that can be used by a BGP speaker to indicate to its peer the completion of the initial routing update after the session is established.

For the IPv4 unicast address family, the End-of-RIB marker is an UPDATE message with the minimum length. For any other address family, it is an UPDATE message that contains only the MP_UNREACH_NLRI attribute with no withdrawn routes for that <AFI, SAFI>.

Although the End-of-RIB marker is specified for the purpose of BGP graceful restart, it is noted that the generation of such a marker upon completion of the initial update would be useful for routing convergence in general, and thus the practice is recommended.

Route Selection Deferral Timer

Specifies the time the restarting router defers the route selection process after restart.

Restarting Router : The usage of route election deferral timer is specified in https://tools.ietf.org/html/rfc4724#section-4.1

Once the session between the Restarting Speaker and the Receiving Speaker is re-established, the Restarting Speaker will receive and process BGP messages from its peers.

However, it MUST defer route selection for an address family until it either.

1. Receives the End-of-RIB marker from all its peers (excluding the ones with the "Restart State" bit set in the received capability and excluding the ones that do not advertise the graceful restart capability).
2. The Selection_Deferral_Timer timeout.

bgp graceful-restart select-defer-time (0-3600)
This is command, will set deferral time to value specified.

bgp graceful-restart rib-stale-time (1-3600)
This is command, will set the time for which stale routes are kept in RIB.

bgp graceful-restart restart-time (0-4095)
Set the time to wait to delete stale routes before a BGP open message is received.

Using with Long-lived Graceful Restart capability, this is recommended setting this timer to 0 and control stale routes with `bgp long-lived-graceful-restart stale-time`.

Default value is 120.

bgp graceful-restart stalepath-time (1-4095)
This is command, will set the max time (in seconds) to hold onto restarting peer's stale paths.

It also controls Enhanced Route-Refresh timer.

If this command is configured and the router does not receive a Route-Refresh EoRR message, the router removes the stale routes from the BGP table after the timer expires. The stale path timer is started when the router receives a Route-Refresh BoRR message.

bgp graceful-restart notification
Indicate Graceful Restart support for BGP NOTIFICATION messages.

After changing this parameter, you have to reset the peers in order to advertise N-bit in Graceful Restart capability.

Without Graceful-Restart Notification capability (N-bit not set), GR is not activated when receiving CEASE/HOLDTIME expire notifications.

When sending `CEASE/Administrative Reset` (`clear bgp`), the session is closed and routes are not retained. When N-bit is set and `bgp hard-administrative-reset` is turned off Graceful-Restart is activated and routes are retained.

Enabled by default.

BGP Per Peer Graceful Restart

Ability to enable and disable graceful restart, helper and no GR at all mode functionality at peer level.

So bgp graceful restart can be enabled at modes global BGP level or at per peer level. There are two FSM, one for BGP GR global mode and other for peer per GR.

Default global mode is helper and default peer per mode is inherit from global. If per peer mode is configured, the GR mode of this particular peer will override the global mode.

BGP GR Global Mode Commands

bgp graceful-restart
This command will enable BGP graceful restart functionality at the global level.

bgp graceful-restart disable
This command will disable both the functionality graceful restart and helper mode.

BGP GR Peer Mode Commands

neighbor A.B.C.D graceful-restart
This command will enable BGP graceful restart functionality at the peer level.

neighbor A.B.C.D graceful-restart-helper
This command will enable BGP graceful restart helper only functionality at the peer level.

neighbor A.B.C.D graceful-restart-disable
This command will disable the entire BGP graceful restart functionality at the peer level.

Long-lived Graceful Restart

Currently, only restarter mode is supported. This capability is advertised only if graceful restart capability is negotiated.

bgp long-lived-graceful-restart stale-time (1-16777215)
Specifies the maximum time to wait before purging long-lived stale routes for helper routers.

Default is 0, which means the feature is off by default. Only graceful restart takes into account.

Administrative Shutdown

bgp shutdown [message MSG...]
Administrative shutdown of all peers of a bgp instance. Drop all BGP peers, but preserve their configurations. The peers are notified in accordance with RFC 8203 by sending a `NOTIFICATION` message with error code `Cease` and subcode `Administrative Shutdown` prior to terminating connections. This global shutdown is independent of the neighbor shutdown, meaning that individually shut down peers will not be affected by lifting it.

An optional shutdown message *MSG* can be specified.

Networks

network A.B.C.D/M
This command adds the announcement network.

```
router bgp 1
 address-family ipv4 unicast
  network 10.0.0.0/8
 exit-address-family
```

This configuration example says that network 10.0.0.0/8 will be announced to all neighbors. Some vendors' routers don't advertise routes if they aren't present in their IGP routing tables; *bgpd* doesn't care about IGP routes when announcing its routes.

bgp network import-check
This configuration modifies the behavior of the network statement. If you have this configured the underlying network must exist in the rib. If you have the [no] form configured then BGP will not check for the networks existence in the rib. For versions 7.3 and before frr defaults for datacenter were the network must exist, traditional did not check for existence. For versions 7.4 and beyond both traditional and datacenter the network must exist.

IPv6 Support

neighbor A.B.C.D activate
This configuration modifies whether to enable an address family for a specific neighbor. By default only the IPv4 unicast address family is enabled.

```
router bgp 1
 address-family ipv6 unicast
  neighbor 2001:0DB8::1 activate
  network 2001:0DB8:5009::/64
 exit-address-family
```

This configuration example says that network 2001:0DB8:5009::/64 will be announced and enables the neighbor 2001:0DB8::1 to receive this announcement.

By default, only the IPv4 unicast address family is announced to all neighbors. Using the 'no bgp default ipv4-unicast' configuration overrides this default so that all address families need to be enabled explicitly.

```
router bgp 1
 no bgp default ipv4-unicast
 neighbor 10.10.10.1 remote-as 2
 neighbor 2001:0DB8::1 remote-as 3
 address-family ipv4 unicast
  neighbor 10.10.10.1 activate
  network 192.168.1.0/24
 exit-address-family
 address-family ipv6 unicast
  neighbor 2001:0DB8::1 activate
  network 2001:0DB8:5009::/64
 exit-address-family
```

This configuration demonstrates how the 'no bgp default ipv4-unicast' might be used in a setup with two upstreams where each of the upstreams should only receive either IPv4 or IPv6 announcements.

Using the `bgp default ipv6-unicast` configuration, IPv6 unicast address family is enabled by default for all new neighbors.

Route Aggregation

Route Aggregation-IPv4 Address Family

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

In order to advertise an aggregated prefix, a more specific (longer) prefix MUST exist in the BGP table. For example, if you want to create an `aggregate-address 10.0.0.0/24`, you should make sure you have something like `10.0.0.5/32` or `10.0.0.0/26`, or any other smaller prefix in the BGP table. The routing information table (RIB) is not enough, you have to redistribute them into the BGP table.

aggregate-address A.B.C.D/M route-map NAME
Apply a route-map for an aggregated prefix.

aggregate-address A.B.C.D/M origin <egp|igp|incomplete>
Override ORIGIN for an aggregated prefix.

aggregate-address A.B.C.D/M as-set
This command specifies an aggregate address. Resulting routes include AS set.

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

Longer prefixes advertisements of more specific routes to all neighbors are suppressed.

aggregate-address A.B.C.D/M matching-MED-only
Configure the aggregated address to only be created when the routes MED match, otherwise no aggregated route will be created.

aggregate-address A.B.C.D/M suppress-map NAME
Similar to *summary-only*, but will only suppress more specific routes that are matched by the selected route-map.

This configuration example sets up an `aggregate-address` under the ipv4 address-family.

```
router bgp 1
 address-family ipv4 unicast
  aggregate-address 10.0.0.0/8
  aggregate-address 20.0.0.0/8 as-set
  aggregate-address 40.0.0.0/8 summary-only
  aggregate-address 50.0.0.0/8 route-map aggr-rmap
 exit-address-family
```

Route Aggregation-IPv6 Address Family

aggregate-address X:X::X:X/M
This command specifies an aggregate address.

aggregate-address X:X::X:X/M route-map NAME
Apply a route-map for an aggregated prefix.

aggregate-address X:X::X:X/M origin <egp|igp|incomplete>
Override ORIGIN for an aggregated prefix.

aggregate-address X:X::X:X/M as-set
This command specifies an aggregate address. Resulting routes include AS set.

aggregate-address X:X::X:X/M summary-only
This command specifies an aggregate address.

Longer prefixes advertisements of more specific routes to all neighbors are suppressed

aggregate-address X:X::X:X/M matching-MED-only
Configure the aggregated address to only be created when the routes MED match, otherwise no aggregated route will be created.

aggregate-address X:X::X:X/M suppress-map NAME
Similar to *summary-only*, but will only suppress more specific routes that are matched by the selected route-map.

This configuration example sets up an `aggregate-address` under the ipv6 address-family.

```
router bgp 1
 address-family ipv6 unicast
  aggregate-address 10::0/64
  aggregate-address 20::0/64 as-set
  aggregate-address 40::0/64 summary-only
```

(continues on next page)

(continued from previous page)

```
 aggregate-address 50::0/64 route-map aggr-rmap
exit-address-family
```

Redistribution

Redistribution configuration should be placed under the `address-family` section for the specific AF to redistribute into. Protocol availability for redistribution is determined by BGP AF; for example, you cannot redistribute OSPFv3 into `address-family ipv4 unicast` as OSPFv3 supports IPv6.

redistribute <babel|connected|eigrp|isis|kernel|openfabric|ospf|ospf6|rip|ripng|sharp|static|table> [me

Redistribute routes from other protocols into BGP.

redistribute vnc-direct
: Redistribute VNC direct (not via zebra) routes to BGP process.

bgp update-delay MAX-DELAY

bgp update-delay MAX-DELAY ESTABLISH-WAIT
: This feature is used to enable read-only mode on BGP process restart or when a BGP process is cleared using 'clear ip bgp *'. Note that this command is configured at the global level and applies to all bgp instances/vrfs. It cannot be used at the same time as the "update-delay" command described below, which is entered in each bgp instance/vrf desired to delay update installation and advertisements. The global and per-vrf approaches to defining update-delay are mutually exclusive.

  When applicable, read-only mode would begin as soon as the first peer reaches Established status and a timer for max-delay seconds is started. During this mode BGP doesn't run any best-path or generate any updates to its peers. This mode continues until:

  1. All the configured peers, except the shutdown peers, have sent explicit EOR (End-Of-RIB) or an implicit-EOR. The first keep-alive after BGP has reached Established is considered an implicit-EOR. If the establish-wait optional value is given, then BGP will wait for peers to reach established from the beginning of the update-delay till the establish-wait period is over, i.e. the minimum set of established peers for which EOR is expected would be peers established during the establish-wait window, not necessarily all the configured neighbors.
  2. max-delay period is over.

  On hitting any of the above two conditions, BGP resumes the decision process and generates updates to its peers.

  Default max-delay is 0, i.e. the feature is off by default.

update-delay MAX-DELAY

update-delay MAX-DELAY ESTABLISH-WAIT
: This feature is used to enable read-only mode on BGP process restart or when a BGP process is cleared using 'clear ip bgp *'. Note that this command is configured under the specific bgp instance/vrf that the feature is enabled for. It cannot be used at the same time as the global "bgp update-delay" described above, which is entered at the global level and applies to all bgp instances. The global and per-vrf approaches to defining update-delay are mutually exclusive.

  When applicable, read-only mode would begin as soon as the first peer reaches Established status and a timer for max-delay seconds is started. During this mode BGP doesn't run any best-path or generate any updates to its peers. This mode continues until:

  1. All the configured peers, except the shutdown peers, have sent explicit EOR (End-Of-RIB) or an implicit-EOR. The first keep-alive after BGP has reached Established is considered an implicit-EOR. If the establish-wait optional value is given, then BGP will wait for peers to reach established from the beginning of the

update-delay till the establish-wait period is over, i.e. the minimum set of established peers for which EOR is expected would be peers established during the establish-wait window, not necessarily all the configured neighbors.

2. max-delay period is over.

On hitting any of the above two conditions, BGP resumes the decision process and generates updates to its peers.

Default max-delay is 0, i.e. the feature is off by default.

table-map ROUTE-MAP-NAME
This feature is used to apply a route-map on route updates from BGP to Zebra. All the applicable match operations are allowed, such as match on prefix, next-hop, communities, etc. Set operations for this attach-point are limited to metric and next-hop only. Any operation of this feature does not affect BGPs internal RIB.

Supported for ipv4 and ipv6 address families. It works on multi-paths as well, however, metric setting is based on the best-path only.

Peers

Defining Peers

neighbor PEER remote-as ASN
Creates a new neighbor whose remote-as is ASN. PEER can be an IPv4 address or an IPv6 address or an interface to use for the connection.

```
router bgp 1
 neighbor 10.0.0.1 remote-as 2
```

In this case my router, in AS-1, is trying to peer with AS-2 at 10.0.0.1.

This command must be the first command used when configuring a neighbor. If the remote-as is not specified, *bgpd* will complain like this:

```
can't find neighbor 10.0.0.1
```

neighbor PEER remote-as internal
Create a peer as you would when you specify an ASN, except that if the peers ASN is different than mine as specified under the *router bgp ASN* command the connection will be denied.

neighbor PEER remote-as external
Create a peer as you would when you specify an ASN, except that if the peers ASN is the same as mine as specified under the *router bgp ASN* command the connection will be denied.

bgp listen range <A.B.C.D/M|X:X::X:X/M> peer-group PGNAME
Accept connections from any peers in the specified prefix. Configuration from the specified peer-group is used to configure these peers.

Note: When using BGP listen ranges, if the associated peer group has TCP MD5 authentication configured, your kernel must support this on prefixes. On Linux, this support was added in kernel version 4.14. If your kernel does not support this feature you will get a warning in the log file, and the listen range will only accept connections from peers without MD5 configured.

Additionally, we have observed that when using this option at scale (several hundred peers) the kernel may hit its option memory limit. In this situation you will see error messages like:

```
bgpd: sockopt_tcp_signature: setsockopt(23): Cannot allocate memory
```

In this case you need to increase the value of the sysctl `net.core.optmem_max` to allow the kernel to allocate the necessary option memory.

---

bgp listen limit <1-65535>
Define the maximum number of peers accepted for one BGP instance. This limit is set to 100 by default. Increasing this value will really be possible if more file descriptors are available in the BGP process. This value is defined by the underlying system (ulimit value), and can be overridden by *–limit-fds*. More information is available in chapter (*Common Invocation Options*).

coalesce-time (0-4294967295)
The time in milliseconds that BGP will delay before deciding what peers can be put into an update-group together in order to generate a single update for them. The default time is 1000.

Configuring Peers

neighbor PEER shutdown [message MSG...] [rtt (1-65535) [count (1-255)]]
Shutdown the peer. We can delete the neighbor's configuration by `no neighbor PEER remote-as ASN` but all configuration of the neighbor will be deleted. When you want to preserve the configuration, but want to drop the BGP peer, use this syntax.

Optionally you can specify a shutdown message *MSG*.

Also, you can specify optionally `rtt` in milliseconds to automatically shutdown the peer if round-trip-time becomes higher than defined.

Additional `count` parameter is the number of keepalive messages to count before shutdown the peer if round-trip-time becomes higher than defined.

neighbor PEER disable-connected-check
Allow peerings between directly connected eBGP peers using loopback addresses.

neighbor PEER disable-link-bw-encoding-ieee
By default bandwidth in extended communities is carried encoded as IEEE floating-point format, which is according to the draft.

Older versions have the implementation where extended community bandwidth value is carried encoded as uint32. To enable backward compatibility we need to disable IEEE floating-point encoding option per-peer.

neighbor PEER extended-optional-parameters
Force Extended Optional Parameters Length format to be used for OPEN messages.

By default, it's disabled. If the standard optional parameters length is higher than one-octet (255), then extended format is enabled automatically.

For testing purposes, extended format can be enabled with this command.

neighbor PEER ebgp-multihop
Specifying `ebgp-multihop` allows sessions with eBGP neighbors to establish when they are multiple hops away. When the neighbor is not directly connected and this knob is not enabled, the session will not establish.

If the peer's IP address is not in the RIB and is reachable via the default route, then you have to enable `ip nht resolve-via-default`.

neighbor PEER description ...
Set description of the peer.

neighbor PEER interface IFNAME
When you connect to a BGP peer over an IPv6 link-local address, you have to specify the IFNAME of the

interface used for the connection. To specify IPv4 session addresses, see the `neighbor PEER update-source` command below.

neighbor PEER interface remote-as <internal|external|ASN>
Configure an unnumbered BGP peer. `PEER` should be an interface name. The session will be established via IPv6 link locals. Use `internal` for iBGP and `external` for eBGP sessions, or specify an ASN if you wish.

neighbor PEER next-hop-self [force]
This command specifies an announced route's nexthop as being equivalent to the address of the bgp router if it is learned via eBGP. This will also bypass third-party next-hops in favor of the local bgp address. If the optional keyword `force` is specified the modification is done also for routes learned via iBGP.

neighbor PEER attribute-unchanged [{as-path|next-hop|med}]
This command specifies attributes to be left unchanged for advertisements sent to a peer. Use this to leave the next-hop unchanged in ipv6 configurations, as the route-map directive to leave the next-hop unchanged is only available for ipv4.

neighbor PEER update-source <IFNAME|ADDRESS>
Specify the IPv4 source address to use for the BGP session to this neighbour, may be specified as either an IPv4 address directly or as an interface name (in which case the *zebra* daemon MUST be running in order for *bgpd* to be able to retrieve interface state).

```
router bgp 64555
 neighbor foo update-source 192.168.0.1
 neighbor bar update-source lo0
```

neighbor PEER default-originate [route-map WORD]
bgpd's default is to not announce the default route (0.0.0.0/0) even if it is in routing table. When you want to announce default routes to the peer, use this command.

If `route-map` keyword is specified, then the default route will be originated only if route-map conditions are met. For example, announce the default route only if `10.10.10.10/32` route exists and set an arbitrary community for a default route.

```
router bgp 64555
 address-family ipv4 unicast
  neighbor 192.168.255.1 default-originate route-map default
!
ip prefix-list p1 seq 5 permit 10.10.10.10/32
!
route-map default permit 10
 match ip address prefix-list p1
 set community 123:123
!
```

neighbor PEER port PORT

neighbor PEER password PASSWORD
Set a MD5 password to be used with the tcp socket that is being used to connect to the remote peer. Please note if you are using this command with a large number of peers on linux you should consider modifying the *net.core.optmem_max* sysctl to a larger value to avoid out of memory errors from the linux kernel.

neighbor PEER send-community

neighbor PEER weight WEIGHT
This command specifies a default *weight* value for the neighbor's routes.

neighbor PEER maximum-prefix NUMBER [force]
Sets a maximum number of prefixes we can receive from a given peer. If this number is exceeded, the BGP

session will be destroyed.

In practice, it is generally preferable to use a prefix-list to limit what prefixes are received from the peer instead of using this knob. Tearing down the BGP session when a limit is exceeded is far more destructive than merely rejecting undesired prefixes. The prefix-list method is also much more granular and offers much smarter matching criterion than number of received prefixes, making it more suited to implementing policy.

If `force` is set, then ALL prefixes are counted for maximum instead of accepted only. This is useful for cases where an inbound filter is applied, but you want maximum-prefix to act on ALL (including filtered) prefixes. This option requires *soft-reconfiguration inbound* to be enabled for the peer.

neighbor PEER maximum-prefix-out NUMBER
Sets a maximum number of prefixes we can send to a given peer.

Since sent prefix count is managed by update-groups, this option creates a separate update-group for outgoing updates.

neighbor PEER local-as AS-NUMBER [no-prepend] [replace-as]
Specify an alternate AS for this BGP process when interacting with the specified peer. With no modifiers, the specified local-as is prepended to the received AS_PATH when receiving routing updates from the peer, and prepended to the outgoing AS_PATH (after the process local AS) when transmitting local routes to the peer.

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

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

Note that replace-as can only be specified if no-prepend is.

This command is only allowed for eBGP peers.

neighbor <A.B.C.D|X:X::X:X|WORD> as-override
Override AS number of the originating router with the local AS number.

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

This command is only allowed for eBGP peers.

neighbor <A.B.C.D|X:X::X:X|WORD> allowas-in [<(1-10)|origin>]
Accept incoming routes with AS path containing AS number with the same value as the current system AS.

This is used when you want to use the same AS number in your sites, but you can't connect them directly. This is an alternative to *neighbor WORD as-override*.

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

The parameter *origin* configures BGP to only accept routes originated with the same AS number as the system.

This command is only allowed for eBGP peers.

neighbor <A.B.C.D|X:X::X:X|WORD> addpath-tx-all-paths
Configure BGP to send all known paths to neighbor in order to preserve multi path capabilities inside a network.

neighbor <A.B.C.D|X:X::X:X|WORD> addpath-tx-bestpath-per-AS
Configure BGP to send best known paths to neighbor in order to preserve multi path capabilities inside a network.

neighbor <A.B.C.D|X:X::X:X|WORD> disable-addpath-rx
Do not accept additional paths from this neighbor.

neighbor PEER ttl-security hops NUMBER
This command enforces Generalized TTL Security Mechanism (GTSM), as specified in RFC 5082. With this

command, only neighbors that are the specified number of hops away will be allowed to become neighbors. This command is mutually exclusive with *ebgp-multihop*.

neighbor PEER capability extended-nexthop
Allow bgp to negotiate the extended-nexthop capability with it's peer. If you are peering over a v6 LL address then this capability is turned on automatically. If you are peering over a v6 Global Address then turning on this command will allow BGP to install v4 routes with v6 nexthops if you do not have v4 configured on interfaces.

neighbor <A.B.C.D|X:X::X:X|WORD> accept-own
Enable handling of self-originated VPN routes containing `accept-own` community.

This feature allows you to handle self-originated VPN routes, which a BGP speaker receives from a route-reflector. A 'self-originated' route is one that was originally advertised by the speaker itself. As per **RFC 4271**, a BGP speaker rejects advertisements that originated the speaker itself. However, the BGP ACCEPT_OWN mechanism enables a router to accept the prefixes it has advertised, when reflected from a route-reflector that modifies certain attributes of the prefix.

A special community called `accept-own` is attached to the prefix by the route-reflector, which is a signal to the receiving router to bypass the ORIGINATOR_ID and NEXTHOP/MP_REACH_NLRI check.

Default: disabled.

neighbor <A.B.C.D|X:X::X:X|WORD> path-attribute discard (1-255)...
Drops specified path attributes from BGP UPDATE messages from the specified neighbor.

If you do not want specific attributes, you can drop them using this command, and let the BGP proceed by ignoring those attributes.

neighbor <A.B.C.D|X:X::X:X|WORD> path-attribute treat-as-withdraw (1-255)...
Received BGP UPDATES that contain specified path attributes are treat-as-withdraw. If there is an existing prefix in the BGP routing table, it will be removed.

neighbor <A.B.C.D|X:X::X:X|WORD> graceful-shutdown
Mark all routes from this neighbor as less preferred by setting `graceful-shutdown` community, and local-preference to 0.

bgp fast-external-failover
This command causes bgp to take down ebgp peers immediately when a link flaps. *bgp fast-external-failover* is the default and will not be displayed as part of a *show run*. The no form of the command turns off this ability.

bgp default ipv4-unicast
This command allows the user to specify that the IPv4 Unicast address family is turned on by default or not. This command defaults to on and is not displayed. The *no bgp default ipv4-unicast* form of the command is displayed.

bgp default ipv4-multicast
This command allows the user to specify that the IPv4 Multicast address family is turned on by default or not. This command defaults to off and is not displayed. The *bgp default ipv4-multicast* form of the command is displayed.

bgp default ipv4-vpn
This command allows the user to specify that the IPv4 MPLS VPN address family is turned on by default or not. This command defaults to off and is not displayed. The *bgp default ipv4-vpn* form of the command is displayed.

bgp default ipv4-flowspec
This command allows the user to specify that the IPv4 Flowspec address family is turned on by default or not. This command defaults to off and is not displayed. The *bgp default ipv4-flowspec* form of the command is displayed.

bgp default ipv6-unicast
This command allows the user to specify that the IPv6 Unicast address family is turned on by default or not. This command defaults to off and is not displayed. The *bgp default ipv6-unicast* form of the command is displayed.

bgp default ipv6-multicast
This command allows the user to specify that the IPv6 Multicast address family is turned on by default or not. This command defaults to off and is not displayed. The *bgp default ipv6-multicast* form of the command is displayed.

bgp default ipv6-vpn
This command allows the user to specify that the IPv6 MPLS VPN address family is turned on by default or not. This command defaults to off and is not displayed. The *bgp default ipv6-vpn* form of the command is displayed.

bgp default ipv6-flowspec
This command allows the user to specify that the IPv6 Flowspec address family is turned on by default or not. This command defaults to off and is not displayed. The *bgp default ipv6-flowspec* form of the command is displayed.

bgp default l2vpn-evpn
This command allows the user to specify that the L2VPN EVPN address family is turned on by default or not. This command defaults to off and is not displayed. The *bgp default l2vpn-evpn* form of the command is displayed.

bgp default show-hostname
This command shows the hostname of the peer in certain BGP commands outputs. It's easier to troubleshoot if you have a number of BGP peers.

bgp default show-nexthop-hostname
This command shows the hostname of the next-hop in certain BGP commands outputs. It's easier to troubleshoot if you have a number of BGP peers and a number of routes to check.

neighbor PEER advertisement-interval (0-600)
Setup the minimum route advertisement interval(mrai) for the peer in question. This number is between 0 and 600 seconds, with the default advertisement interval being 0.

neighbor PEER timers (0-65535) (0-65535)
Set keepalive and hold timers for a neighbor. The first value is keepalive and the second is hold time.

neighbor PEER timers connect (1-65535)
Set connect timer for a neighbor. The connect timer controls how long BGP waits between connection attempts to a neighbor.

neighbor PEER timers delayopen (1-240)
This command allows the user enable the *RFC 4271 <https://tools.ietf.org/html/rfc4271/>* DelayOpenTimer with the specified interval or disable it with the negating command for the peer. By default, the DelayOpenTimer is disabled. The timer interval may be set to a duration of 1 to 240 seconds.

bgp minimum-holdtime (1-65535)
This command allows user to prevent session establishment with BGP peers with lower holdtime less than configured minimum holdtime. When this command is not set, minimum holdtime does not work.

bgp tcp-keepalive (1-65535) (1-65535) (1-30)
This command allows user to configure TCP keepalive with new BGP peers. Each parameter respectively stands for TCP keepalive idle timer (seconds), interval (seconds), and maximum probes. By default, TCP keepalive is disabled.

Displaying Information about Peers

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

If `detail` option is specified, the detailed version of all routes will be displayed. The same format as `show [ip] bgp [afi] [safi] PREFIX` will be used, but for the whole table of received, advertised or filtered prefixes.

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

If `wide` option is specified, then the prefix table's width is increased to fully display the prefix and the nexthop.

Peer Filtering

neighbor PEER distribute-list NAME [in|out]
This command specifies a distribute-list for the peer. *direct* is `in` or `out`.

neighbor PEER prefix-list NAME [in|out]

neighbor PEER filter-list NAME [in|out]

neighbor PEER route-map NAME [in|out]
Apply a route-map on the neighbor. *direct* must be *in* or *out*.

bgp route-reflector allow-outbound-policy
By default, attribute modification via route-map policy out is not reflected on reflected routes. This option allows the modifications to be reflected as well. Once enabled, it affects all reflected routes.

neighbor PEER sender-as-path-loop-detection
Enable the detection of sender side AS path loops and filter the bad routes before they are sent.

This setting is disabled by default.

Peer Groups

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

neighbor WORD peer-group
This command defines a new peer group.

neighbor PEER peer-group PGNAME
This command bind specific peer to peer group WORD.

neighbor PEER solo
This command is used to indicate that routes advertised by the peer should not be reflected back to the peer. This command only is only meaningful when there is a single peer defined in the peer-group.

show [ip] bgp peer-group [json]
This command displays configured BGP peer-groups.

```
exit1-debian-9# show bgp peer-group

BGP peer-group test1, remote AS 65001
Peer-group type is external
```

(continues on next page)

(continued from previous page)

```
Configured address-families: IPv4 Unicast; IPv6 Unicast;
1 IPv4 listen range(s)
   192.168.100.0/24
2 IPv6 listen range(s)
   2001:db8:1::/64
   2001:db8:2::/64
Peer-group members:
   192.168.200.1  Active
   2001:db8::1  Active

BGP peer-group test2
Peer-group type is external
Configured address-families: IPv4 Unicast;
```

Optional `json` parameter is used to display JSON output.

```
{
  "test1":{
    "remoteAs":65001,
    "type":"external",
    "addressFamiliesConfigured":[
      "IPv4 Unicast",
      "IPv6 Unicast"
    ],
    "dynamicRanges":{
      "IPv4":{
        "count":1,
        "ranges":[
          "192.168.100.0\/24"
        ]
      },
      "IPv6":{
        "count":2,
        "ranges":[
          "2001:db8:1::\/64",
          "2001:db8:2::\/64"
        ]
      }
    },
    "members":{
      "192.168.200.1":{
        "status":"Active"
      },
      "2001:db8::1":{
        "status":"Active"
      }
    }
  },
  "test2":{
     "type":"external",
     "addressFamiliesConfigured":[
       "IPv4 Unicast"
```

(continues on next page)

(continued from previous page)

```
    ]
  }
}
```

Capability Negotiation

neighbor PEER strict-capability-match
Strictly compares remote capabilities and local capabilities. If capabilities are different, send Unsupported Capability error then reset connection.

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

neighbor PEER dont-capability-negotiate
Suppress sending Capability Negotiation as OPEN message optional parameter to the peer. This command only affects the peer is configured other than IPv4 unicast configuration.

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

You may prefer locally configured capabilities more than the negotiated capabilities even though remote peer sends capabilities. If the peer is configured by *override-capability*, *bgpd* ignores received capabilities then override negotiated capabilities with configured values.

Additionally the operator should be reminded that this feature fundamentally disables the ability to use widely deployed BGP features. BGP unnumbered, hostname support, AS4, Addpath, Route Refresh, ORF, Dynamic Capabilities, and graceful restart.

neighbor PEER override-capability
Override the result of Capability Negotiation with local configuration. Ignore remote peer's capability value.

neighbor PEER capability software-version
Send the software version in the BGP OPEN message to the neighbor. This is very useful in environments with a large amount of peers with different versions of FRR or any other vendor.

Disabled by default.

neighbor PEER aigp
Send and receive AIGP attribute for this neighbor. This is valid only for eBGP neighbors.

Disabled by default. iBGP neighbors have this option enabled implicitly.

AS Path Access Lists

AS path access list is user defined AS path.

bgp as-path access-list WORD [seq (0-4294967295)] permit|deny LINE
This command defines a new AS path access list.

show bgp as-path-access-list [json]
Display all BGP AS Path access lists.

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

show bgp as-path-access-list WORD [json]
Display the specified BGP AS Path access list.

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

Bogon ASN filter policy configuration example

```
bgp as-path access-list 99 permit _0_
bgp as-path access-list 99 permit _23456_
bgp as-path access-list 99 permit _1310[0-6][0-9]_|_13107[0-1]_
bgp as-path access-list 99 seq 20 permit ^65
```

Using AS Path in Route Map

match as-path WORD
For a given as-path, WORD, match it on the BGP as-path given for the prefix and if it matches do normal route-map actions. The no form of the command removes this match from the route-map.

set as-path prepend AS-PATH
Prepend the given string of AS numbers to the AS_PATH of the BGP path's NLRI. The no form of this command removes this set operation from the route-map.

set as-path prepend last-as NUM
Prepend the existing last AS number (the leftmost ASN) to the AS_PATH. The no form of this command removes this set operation from the route-map.

set as-path replace <any|ASN>
Replace a specific AS number to local AS number. `any` replaces each AS number in the AS-PATH with the local AS number.

Communities Attribute

The BGP communities attribute is widely used for implementing policy routing. Network operators can manipulate BGP communities attribute based on their network policy. BGP communities attribute is defined in **RFC 1997** and **RFC 1998**. It is an optional transitive attribute, therefore local policy can travel through different autonomous system.

The communities attribute is a set of communities values. Each community value is 4 octet long. The following format is used to define the community value.

AS:VAL This format represents 4 octet communities value. `AS` is high order 2 octet in digit format. `VAL` is low order 2 octet in digit format. This format is useful to define AS oriented policy value. For example, `7675:80` can be used when AS 7675 wants to pass local policy value 80 to neighboring peer.

graceful-shutdown `graceful-shutdown` represents well-known communities value `GRACEFUL_SHUTDOWN` `0xFFFF0000` `65535:0`. **RFC 8326** implements the purpose Graceful BGP Session Shutdown to reduce the amount of lost traffic when taking BGP sessions down for maintenance. The use of the community needs to be supported from your peers side to actually have any effect.

accept-own `accept-own` represents well-known communities value `ACCEPT_OWN` `0xFFFF0001` `65535:1`. **RFC 7611** implements a way to signal to a router to accept routes with a local nexthop address. This can be the case when doing policing and having traffic having a nexthop located in another VRF but still local interface to the router. It is recommended to read the RFC for full details.

route-filter-translated-v4 `route-filter-translated-v4` represents well-known communities value `ROUTE_FILTER_TRANSLATED_v4` `0xFFFF0002` `65535:2`.

route-filter-v4 `route-filter-v4` represents well-known communities value `ROUTE_FILTER_v4` `0xFFFF0003` `65535:3`.

route-filter-translated-v6 route-filter-translated-v6 represents well-known communities value ROUTE_FILTER_TRANSLATED_v6 0xFFFF0004 65535:4.

route-filter-v6 route-filter-v6 represents well-known communities value ROUTE_FILTER_v6 0xFFFF0005 65535:5.

llgr-stale llgr-stale represents well-known communities value LLGR_STALE 0xFFFF0006 65535:6. Assigned and intended only for use with routers supporting the Long-lived Graceful Restart Capability as described in [Draft-IETF-uttaro-idr-bgp-persistence]. Routers receiving routes with this community may (depending on implementation) choose allow to reject or modify routes on the presence or absence of this community.

no-llgr no-llgr represents well-known communities value NO_LLGR 0xFFFF0007 65535:7. Assigned and intended only for use with routers supporting the Long-lived Graceful Restart Capability as described in [Draft-IETF-uttaro-idr-bgp-persistence]. Routers receiving routes with this community may (depending on implementation) choose allow to reject or modify routes on the presence or absence of this community.

accept-own-nexthop accept-own-nexthop represents well-known communities value accept-own-nexthop 0xFFFF0008 65535:8. [Draft-IETF-agrewal-idr-accept-own-nexthop] describes how to tag and label VPN routes to be able to send traffic between VRFs via an internal layer 2 domain on the same PE device. Refer to [Draft-IETF-agrewal-idr-accept-own-nexthop] for full details.

blackhole blackhole represents well-known communities value BLACKHOLE 0xFFFF029A 65535:666. **RFC 7999** 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. Upon receiving BLACKHOLE community from a BGP speaker, NO_ADVERTISE community is added automatically.

no-export no-export represents well-known communities value NO_EXPORT 0xFFFFFF01. All routes carry this value must not be advertised to outside a BGP confederation boundary. If neighboring BGP peer is part of BGP confederation, the peer is considered as inside a BGP confederation boundary, so the route will be announced to the peer.

no-advertise no-advertise represents well-known communities value NO_ADVERTISE 0xFFFFFF02. All routes carry this value must not be advertise to other BGP peers.

local-AS local-AS represents well-known communities value NO_EXPORT_SUBCONFED 0xFFFFFF03. All routes carry this value must not be advertised to external BGP peers. Even if the neighboring router is part of confederation, it is considered as external BGP peer, so the route will not be announced to the peer.

no-peer no-peer represents well-known communities value NOPEER 0xFFFFFF04 65535:65284. **RFC 3765** is used to communicate to another network how the originating network want the prefix propagated.

When the communities attribute is received duplicate community values in the attribute are ignored and value is sorted in numerical order.

Community Lists

Community lists are user defined lists of community attribute values. These lists can be used for matching or manipulating the communities attribute in UPDATE messages.

There are two types of community list:

standard This type accepts an explicit value for the attribute.

expanded This type accepts a regular expression. Because the regex must be interpreted on each use expanded community lists are slower than standard lists.

bgp community-list standard NAME permit|deny COMMUNITY
This command defines a new standard community list. COMMUNITY is communities value. The COMMUNITY is compiled into community structure. We can define multiple community list under same name. In that case match will happen user defined order. Once the community list matches to communities attribute in BGP updates it

return permit or deny by the community list definition. When there is no matched entry, deny will be returned. When `COMMUNITY` is empty it matches to any routes.

bgp community-list expanded NAME permit|deny COMMUNITY
This command defines a new expanded community list. `COMMUNITY` is a string expression of communities attribute. `COMMUNITY` can be a regular expression (*BGP Regular Expressions*) to match the communities attribute in BGP updates. The expanded community is only used to filter, not *set* actions.

Deprecated since version 5.0: It is recommended to use the more explicit versions of this command.

bgp community-list NAME permit|deny COMMUNITY
When the community list type is not specified, the community list type is automatically detected. If `COMMUNITY` can be compiled into communities attribute, the community list is defined as a standard community list. Otherwise it is defined as an expanded community list. This feature is left for backward compatibility. Use of this feature is not recommended.

Note that all community lists share the same namespace, so it's not necessary to specify `standard` or `expanded`; these modifiers are purely aesthetic.

show bgp community-list [NAME detail]
Displays community list information. When `NAME` is specified the specified community list's information is shown.

```
# show bgp community-list
Named Community standard list CLIST
permit 7675:80 7675:100 no-export
deny internet
  Named Community expanded list EXPAND
permit :

  # show bgp community-list CLIST detail
  Named Community standard list CLIST
permit 7675:80 7675:100 no-export
deny internet
```

Numbered Community Lists

When number is used for BGP community list name, the number has special meanings. Community list number in the range from 1 to 99 is standard community list. Community list number in the range from 100 to 500 is expanded community list. These community lists are called as numbered community lists. On the other hand normal community lists is called as named community lists.

bgp community-list (1-99) permit|deny COMMUNITY
This command defines a new community list. The argument to (1-99) defines the list identifier.

bgp community-list (100-500) permit|deny COMMUNITY
This command defines a new expanded community list. The argument to (100-500) defines the list identifier.

Community alias

BGP community aliases are useful to quickly identify what communities are set for a specific prefix in a human-readable format. Especially handy for a huge amount of communities. Accurately defined aliases can help you faster spot things on the wire.

bgp community alias NAME ALIAS
This command creates an alias name for a community that will be used later in various CLI outputs in a human-readable format.

```
~# vtysh -c 'show run' | grep 'bgp community alias'
bgp community alias 65001:14 community-1
bgp community alias 65001:123:1 lcommunity-1

~# vtysh -c 'show ip bgp 172.16.16.1/32'
BGP routing table entry for 172.16.16.1/32, version 21
Paths: (2 available, best #2, table default)
  Advertised to non peer-group peers:
  65030
    192.168.0.2 from 192.168.0.2 (172.16.16.1)
      Origin incomplete, metric 0, valid, external, best (Neighbor IP)
      Community: 65001:12 65001:13 community-1 65001:65534
      Large Community: lcommunity-1 65001:123:2
      Last update: Fri Apr 16 12:51:27 2021
```

show bgp [afi] [safi] [all] alias WORD [wide|json]
Display prefixes with matching BGP community alias.

Using Communities in Route Maps

In *Route Maps* we can match on or set the BGP communities attribute. Using this feature network operator can implement their network policy based on BGP communities attribute.

The following commands can be used in route maps:

match alias WORD
This command performs match to BGP updates using community alias WORD. When the one of BGP communities value match to the one of community alias value in community alias, it is match.

match community WORD exact-match [exact-match]
This command perform match to BGP updates using community list WORD. When the one of BGP communities value match to the one of communities value in community list, it is match. When *exact-match* keyword is specified, match happen only when BGP updates have completely same communities value specified in the community list.

set community <none|COMMUNITY> additive
This command sets the community value in BGP updates. If the attribute is already configured, the newly provided value replaces the old one unless the `additive` keyword is specified, in which case the new value is appended to the existing value.

If `none` is specified as the community value, the communities attribute is not sent.

It is not possible to set an expanded community list.

set comm-list WORD delete
This command remove communities value from BGP communities attribute. The `word` is community list name. When BGP route's communities value matches to the community list `word`, the communities value is removed.

When all of communities value is removed eventually, the BGP update's communities attribute is completely removed.

Example Configuration

The following configuration is exemplary of the most typical usage of BGP communities attribute. In the example, AS 7675 provides an upstream Internet connection to AS 100. When the following configuration exists in AS 7675, the network operator of AS 100 can set local preference in AS 7675 network by setting BGP communities attribute to the updates.

```
router bgp 7675
 neighbor 192.168.0.1 remote-as 100
 address-family ipv4 unicast
  neighbor 192.168.0.1 route-map RMAP in
 exit-address-family
!
bgp community-list 70 permit 7675:70
bgp community-list 80 permit 7675:80
bgp community-list 90 permit 7675:90
!
route-map RMAP permit 10
 match community 70
 set local-preference 70
!
route-map RMAP permit 20
 match community 80
 set local-preference 80
!
route-map RMAP permit 30
 match community 90
 set local-preference 90
```

The following configuration announces `10.0.0.0/8` from AS 100 to AS 7675. The route has communities value `7675:80` so when above configuration exists in AS 7675, the announced routes' local preference value will be set to 80.

```
router bgp 100
 network 10.0.0.0/8
 neighbor 192.168.0.2 remote-as 7675
 address-family ipv4 unicast
  neighbor 192.168.0.2 route-map RMAP out
 exit-address-family
!
ip prefix-list PLIST permit 10.0.0.0/8
!
route-map RMAP permit 10
 match ip address prefix-list PLIST
 set community 7675:80
```

The following configuration is an example of BGP route filtering using communities attribute. This configuration only permit BGP routes which has BGP communities value (`0:80` and `0:90`) or `0:100`. The network operator can set special internal communities value at BGP border router, then limit the BGP route announcements into the internal network.

```
router bgp 7675
 neighbor 192.168.0.1 remote-as 100
 address-family ipv4 unicast
  neighbor 192.168.0.1 route-map RMAP in
 exit-address-family
!
bgp community-list 1 permit 0:80 0:90
bgp community-list 1 permit 0:100
!
route-map RMAP permit in
 match community 1
```

The following example filters BGP routes which have a community value of `1:1`. When there is no match community-list returns `deny`. To avoid filtering all routes, a `permit` line is set at the end of the community-list.

```
router bgp 7675
 neighbor 192.168.0.1 remote-as 100
 address-family ipv4 unicast
  neighbor 192.168.0.1 route-map RMAP in
 exit-address-family
!
bgp community-list standard FILTER deny 1:1
bgp community-list standard FILTER permit
!
route-map RMAP permit 10
 match community FILTER
```

The following configuration is an example of communities value deletion. With this configuration the community values `100:1` and `100:2` are removed from BGP updates. For communities value deletion, only `permit` community-list is used. `deny` community-list is ignored.

```
router bgp 7675
 neighbor 192.168.0.1 remote-as 100
 address-family ipv4 unicast
  neighbor 192.168.0.1 route-map RMAP in
 exit-address-family
!
bgp community-list standard DEL permit 100:1 100:2
!
route-map RMAP permit 10
 set comm-list DEL delete
```

Extended Communities Attribute

BGP extended communities attribute is introduced with MPLS VPN/BGP technology. MPLS VPN/BGP expands capability of network infrastructure to provide VPN functionality. At the same time it requires a new framework for policy routing. With BGP Extended Communities Attribute we can use Route Target or Site of Origin for implementing network policy for MPLS VPN/BGP.

BGP Extended Communities Attribute is similar to BGP Communities Attribute. It is an optional transitive attribute. BGP Extended Communities Attribute can carry multiple Extended Community value. Each Extended Community value is eight octet length.

BGP Extended Communities Attribute provides an extended range compared with BGP Communities Attribute. Adding to that there is a type field in each value to provides community space structure.

There are two format to define Extended Community value. One is AS based format the other is IP address based format.

AS:VAL This is a format to define AS based Extended Community value. `AS` part is 2 octets Global Administrator subfield in Extended Community value. `VAL` part is 4 octets Local Administrator subfield. `7675:100` represents AS 7675 policy value 100.

IP-Address:VAL This is a format to define IP address based Extended Community value. `IP-Address` part is 4 octets Global Administrator subfield. `VAL` part is 2 octets Local Administrator subfield.

Extended Community Lists

bgp extcommunity-list standard NAME permit|deny EXTCOMMUNITY
This command defines a new standard extcommunity-list. *extcommunity* is extended communities value. The *extcommunity* is compiled into extended community structure. We can define multiple extcommunity-list under same name. In that case match will happen user defined order. Once the extcommunity-list matches to extended communities attribute in BGP updates it return permit or deny based upon the extcommunity-list definition. When there is no matched entry, deny will be returned. When *extcommunity* is empty it matches to any routes.

bgp extcommunity-list expanded NAME permit|deny LINE
This command defines a new expanded extcommunity-list. *line* is a string expression of extended communities attribute. *line* can be a regular expression (*BGP Regular Expressions*) to match an extended communities attribute in BGP updates.

Note that all extended community lists shares a single name space, so it's not necessary to specify their type when creating or destroying them.

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

BGP Extended Communities in Route Map

match extcommunity WORD

set extcommunity none
This command resets the extended community value in BGP updates. If the attribute is already configured or received from the peer, the attribute is discarded and set to none. This is useful if you need to strip incoming extended communities.

set extcommunity rt EXTCOMMUNITY
This command sets Route Target value.

set extcommunity nt EXTCOMMUNITY
This command sets Node Target value.

If the receiving BGP router supports Node Target Extended Communities, it will install the route with the community that contains it's own local BGP Identifier. Otherwise, it's not installed.

set extcommunity soo EXTCOMMUNITY
This command sets Site of Origin value.

set extcommunity bandwidth <(1-25600) | cumulative | num-multipaths> [non-transitive]
This command sets the BGP link-bandwidth extended community for the prefix (best path) for which it is applied. The link-bandwidth can be specified as an `explicit value` (specified in Mbps), or the router can be told to

use the `cumulative bandwidth` of all multipaths for the prefix or to compute it based on the `number of multipaths`. The link bandwidth extended community is encoded as `transitive` unless the set command explicitly configures it as `non-transitive`.

See also:

Weighted ECMP using BGP link bandwidth

Note that the extended expanded community is only used for *match* rule, not for *set* actions.

Large Communities Attribute

The BGP Large Communities attribute was introduced in Feb 2017 with **RFC 8092**.

The BGP Large Communities Attribute is similar to the BGP Communities Attribute except that it has 3 components instead of two and each of which are 4 octets in length. Large Communities bring additional functionality and convenience over traditional communities, specifically the fact that the `GLOBAL` part below is now 4 octets wide allowing seamless use in networks using 4-byte ASNs.

`GLOBAL:LOCAL1:LOCAL2` This is the format to define Large Community values. Referencing **RFC 8195** the values are commonly referred to as follows:

- The `GLOBAL` part is a 4 octet Global Administrator field, commonly used as the operators AS number.
- The `LOCAL1` part is a 4 octet Local Data Part 1 subfield referred to as a function.
- The `LOCAL2` part is a 4 octet Local Data Part 2 field and referred to as the parameter subfield.

As an example, `65551:1:10` represents AS 65551 function 1 and parameter 10. The referenced RFC above gives some guidelines on recommended usage.

Large Community Lists

Two types of large community lists are supported, namely *standard* and *expanded*.

`bgp large-community-list standard NAME permit|deny LARGE-COMMUNITY`
This command defines a new standard large-community-list. *large-community* is the Large Community value. We can add multiple large communities under same name. In that case the match will happen in the user defined order. Once the large-community-list matches the Large Communities attribute in BGP updates it will return permit or deny based upon the large-community-list definition. When there is no matched entry, a deny will be returned. When *large-community* is empty it matches any routes.

`bgp large-community-list expanded NAME permit|deny LINE`
This command defines a new expanded large-community-list. Where *line* is a string matching expression, it will be compared to the entire Large Communities attribute as a string, with each large-community in order from lowest to highest. *line* can also be a regular expression which matches this Large Community attribute.

Note that all community lists share the same namespace, so it's not necessary to specify `standard` or `expanded`; these modifiers are purely aesthetic.

`show bgp large-community-list`

`show bgp large-community-list NAME detail`
This command display current large-community-list information. When *name* is specified the community list information is shown.

`show ip bgp large-community-info`
This command displays the current large communities in use.

Large Communities in Route Map

match large-community LINE [exact-match]
Where *line* can be a simple string to match, or a regular expression. It is very important to note that this match occurs on the entire large-community string as a whole, where each large-community is ordered from lowest to highest. When *exact-match* keyword is specified, match happen only when BGP updates have completely same large communities value specified in the large community list.

set large-community LARGE-COMMUNITY

set large-community LARGE-COMMUNITY LARGE-COMMUNITY

set large-community LARGE-COMMUNITY additive
These commands are used for setting large-community values. The first command will overwrite any large-communities currently present. The second specifies two large-communities, which overwrites the current large-community list. The third will add a large-community value without overwriting other values. Multiple large-community values can be specified.

Note that the large expanded community is only used for *match* rule, not for *set* actions.

BGP Roles and Only to Customers

BGP roles are defined in **RFC 9234** and provide an easy way to route leaks prevention, detection and mitigation.

To enable its mechanics, you must set your local role to reflect your type of peering relationship with your neighbor. Possible values of `LOCAL-ROLE` are:

- provider
- rs-server
- rs-client
- customer
- peer

The local Role value is negotiated with the new BGP Role capability with a built-in check of the corresponding value. In case of mismatch the new OPEN Roles Mismatch Notification <2, 11> would be sent.

The correct Role pairs are:

- Provider - Customer
- Peer - Peer
- RS-Server - RS-Client

```
~# vtysh -c 'show bgp neighbor' | grep 'Role'
  Local Role: customer
  Neighbor Role: provider
    Role: advertised and received
```

If strict-mode is set BGP session won't become established until BGP neighbor set local Role on its side. This configuration parameter is defined in **RFC 9234** and used to enforce corresponding configuration at your counter-part side. Default value - disabled.

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

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

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

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

neighbor PEER local-role LOCAL-ROLE [strict-mode]

This command set your local-role to LOCAL-ROLE: <provider|rs-server|rs-client|customer|peer>.

This role helps to detect and prevent route leaks.

If strict-mode is set, your neighbor must send you Capability with the value of his role (by setting local-role on his side). Otherwise, a Role Mismatch Notification will be sent.

Labeled unicast

bgpd supports labeled information, as per **RFC 3107**.

bgp labeled-unicast <explicit-null|ipv4-explicit-null|ipv6-explicit-null>

By default, locally advertised prefixes use the *implicit-null* label to encode in the outgoing NLRI. The following command uses the *explicit-null* label value for all the BGP instances.

L3VPN VRFs

bgpd supports L3VPN (Layer 3 Virtual Private Networks) VRFs (Virtual Routing and Forwarding) for IPv4 **RFC 4364** and IPv6 **RFC 4659**. L3VPN routes, and their associated VRF MPLS labels, can be distributed to VPN SAFI neighbors in the *default*, i.e., non VRF, BGP instance. VRF MPLS labels are reached using *core* MPLS labels which are distributed using LDP or BGP labeled unicast. *bgpd* also supports inter-VRF route leaking.

L3VPN over GRE interfaces

In MPLS-VPN or SRv6-VPN, an L3VPN next-hop entry requires that the path chosen respectively contains a labelled path or a valid SID IPv6 address. Otherwise the L3VPN entry will not be installed. It is possible to ignore that check when the path chosen by the next-hop uses a GRE interface, and there is a route-map configured at inbound side of ipv4-vpn or ipv6-vpn address family with following syntax:

set l3vpn next-hop encapsulation gre

The incoming BGP L3VPN entry is accepted, provided that the next hop of the L3VPN entry uses a path that takes the GRE tunnel as outgoing interface. The remote endpoint should be configured just behind the GRE tunnel; remote device configuration may vary depending whether it acts at edge endpoint or not: in any case, the expectation is that incoming MPLS traffic received at this endpoint should be considered as a valid path for L3VPN.

VRF Route Leaking

BGP routes may be leaked (i.e. copied) between a unicast VRF RIB and the VPN SAFI RIB of the default VRF for use in MPLS-based L3VPNs. Unicast routes may also be leaked between any VRFs (including the unicast RIB of the default BGP instanced). A shortcut syntax is also available for specifying leaking from one VRF to another VRF using the default instance's VPN RIB as the intermediary. A common application of the VRF-VRF feature is to connect a customer's private routing domain to a provider's VPN service. Leaking is configured from the point of view of an individual VRF: `import` refers to routes leaked from VPN to a unicast VRF, whereas `export` refers to routes leaked from a unicast VRF to VPN.

Required parameters

Routes exported from a unicast VRF to the VPN RIB must be augmented by two parameters:

- an RD (Route Distinguisher)
- an RTLIST (Route-target List)

Configuration for these exported routes must, at a minimum, specify these two parameters.

Routes imported from the VPN RIB to a unicast VRF are selected according to their RTLISTs. Routes whose RTLIST contains at least one route-target in common with the configured import RTLIST are leaked. Configuration for these imported routes must specify an RTLIST to be matched.

The RD, which carries no semantic value, is intended to make the route unique in the VPN RIB among all routes of its prefix that originate from all the customers and sites that are attached to the provider's VPN service. Accordingly, each site of each customer is typically assigned an RD that is unique across the entire provider network.

The RTLIST is a set of route-target extended community values whose purpose is to specify route-leaking policy. Typically, a customer is assigned a single route-target value for import and export to be used at all customer sites. This configuration specifies a simple topology wherein a customer has a single routing domain which is shared across all its sites. More complex routing topologies are possible through use of additional route-targets to augment the leaking of sets of routes in various ways.

When using the shortcut syntax for vrf-to-vrf leaking, the RD and RT are auto-derived.

General configuration

Configuration of route leaking between a unicast VRF RIB and the VPN SAFI RIB of the default VRF is accomplished via commands in the context of a VRF address-family:

rd vpn export AS:NN|IP:nn
: Specifies the route distinguisher to be added to a route exported from the current unicast VRF to VPN.

rt vpn import|export|both RTLIST...
: Specifies the route-target list to be attached to a route (export) or the route-target list to match against (import) when exporting/importing between the current unicast VRF and VPN.

  The RTLIST is a space-separated list of route-targets, which are BGP extended community values as described in *Extended Communities Attribute*.

label vpn export allocation-mode per-vrf|per-nexthop
: Select how labels are allocated in the given VRF. By default, the *per-vrf* mode is selected, and one label is used for all prefixes from the VRF. The *per-nexthop* will use a unique label for all prefixes that are reachable via the same nexthop.

label vpn export (0..1048575)|auto
: Enables an MPLS label to be attached to a route exported from the current unicast VRF to VPN. If the value specified is `auto`, the label value is automatically assigned from a pool maintained by the Zebra daemon. If Zebra is not running, or if this command is not configured, automatic label assignment will not complete, which will block corresponding route export.

nexthop vpn export A.B.C.D|X:X::X:X
: Specifies an optional nexthop value to be assigned to a route exported from the current unicast VRF to VPN. If left unspecified, the nexthop will be set to 0.0.0.0 or 0:0::0:0 (self).

route-map vpn import|export MAP
: Specifies an optional route-map to be applied to routes imported or exported between the current unicast VRF and VPN.

import|export vpn
Enables import or export of routes between the current unicast VRF and VPN.

import vrf VRFNAME
Shortcut syntax for specifying automatic leaking from vrf VRFNAME to the current VRF using the VPN RIB as intermediary. The RD and RT are auto derived and should not be specified explicitly for either the source or destination VRF's.

This shortcut syntax mode is not compatible with the explicit *import vpn* and *export vpn* statements for the two VRF's involved. The CLI will disallow attempts to configure incompatible leaking modes.

bgp retain route-target all

It is possible to retain or not VPN prefixes that are not imported by local VRF configuration. This can be done via the following command in the context of the global VPNv4/VPNv6 family. This command defaults to on and is not displayed. The *no bgp retain route-target all* form of the command is displayed.

neighbor <A.B.C.D|X:X::X:X|WORD> soo EXTCOMMUNITY

Without this command, SoO extended community attribute is configured using an inbound route map that sets the SoO value during the update process. With the introduction of the new BGP per-neighbor Site-of-Origin (SoO) feature, two new commands configured in sub-modes under router configuration mode simplify the SoO value configuration.

If we configure SoO per neighbor at PEs, the SoO community is automatically added for all routes from the CPEs. Routes are validated and prevented from being sent back to the same CPE (e.g.: multi-site). This is especially needed when using `as-override` or `allowas-in` to prevent routing loops.

mpls bgp forwarding

It is possible to permit BGP install VPN prefixes without transport labels, by issuing the following command under the interface configuration context. This configuration will install VPN prefixes originated from an e-bgp session, and with the next-hop directly connected.

L3VPN SRv6

segment-routing srv6
Use SRv6 backend with BGP L3VPN, and go to its configuration node.

locator NAME
Specify the SRv6 locator to be used for SRv6 L3VPN. The Locator name must be set in zebra, but user can set it in any order.

General configuration

Configuration of the SRv6 SID used to advertise a L3VPN for both IPv4 and IPv6 is accomplished via the following command in the context of a VRF:

sid vpn per-vrf export (1..1048575)|auto
Enables a SRv6 SID to be attached to a route exported from the current unicast VRF to VPN. A single SID is used for both IPv4 and IPv6 address families. If you want to set a SID for only IPv4 address family or IPv6 address family, you need to use the command `sid vpn export (1..1048575)|auto` in the context of an address-family. If the value specified is `auto`, the SID value is automatically assigned from a pool maintained by the Zebra daemon. If Zebra is not running, or if this command is not configured, automatic SID assignment will not complete, which will block corresponding route export.

Ethernet Virtual Network - EVPN

Note: When using EVPN features and if you have a large number of hosts, make sure to adjust the size of the arp neighbor cache to avoid neighbor table overflow and/or excessive garbage collection. On Linux, the size of the table and garbage collection frequency can be controlled via the following sysctl configurations:

```
net.ipv4.neigh.default.gc_thresh1
net.ipv4.neigh.default.gc_thresh2
net.ipv4.neigh.default.gc_thresh3

net.ipv6.neigh.default.gc_thresh1
net.ipv6.neigh.default.gc_thresh2
net.ipv6.neigh.default.gc_thresh3
```

For more information, see `man 7 arp`.

Enabling EVPN

EVPN should be enabled on the BGP instance corresponding to the VRF acting as the underlay for the VXLAN tunneling. In most circumstances this will be the default VRF. The command to enable EVPN for a BGP instance is `advertise-all-vni` which lives under `address-family l2vpn evpn`:

```
router bgp 65001
 !
 address-family l2vpn evpn
  advertise-all-vni
```

A more comprehensive configuration example can be found in the *EVPN* page.

EVPN L3 Route-Targets

`route-target <import|export|both> <RTLIST|auto>`

Modify the route-target set for EVPN advertised type-2/type-5 routes. RTLIST is a list of any of matching `(A.B.C.D:MN|EF:OPQR|GHJK:MN|*:OPQR|*:MN)` where `*` indicates wildcard matching for the AS number. It will be set to match any AS number. This is useful in datacenter deployments with Downstream VNI. `auto` is used to retain the autoconfigure that is default behavior for L3 RTs.

EVPN advertise-PIP

In a EVPN symmetric routing MLAG deployment, all EVPN routes advertised with anycast-IP as next-hop IP and anycast MAC as the Router MAC (RMAC - in BGP EVPN Extended-Community). EVPN picks up the next-hop IP from the VxLAN interface's local tunnel IP and the RMAC is obtained from the MAC of the L3VNI's SVI interface. Note: Next-hop IP is used for EVPN routes whether symmetric routing is deployed or not but the RMAC is only relevant for symmetric routing scenario.

Current behavior is not ideal for Prefix (type-5) and self (type-2) routes. This is because the traffic from remote VTEPs routed sub optimally if they land on the system where the route does not belong.

The advertise-pip feature advertises Prefix (type-5) and self (type-2) routes with system's individual (primary) IP as the next-hop and individual (system) MAC as Router-MAC (RMAC), while leaving the behavior unchanged for other EVPN routes.

To support this feature there needs to have ability to co-exist a (system-MAC, system-IP) pair with a (anycast-MAC, anycast-IP) pair with the ability to terminate VxLAN-encapsulated packets received for either pair on the same L3VNI (i.e associated VLAN). This capability is needed per tenant VRF instance.

To derive the system-MAC and the anycast MAC, there must be a separate/additional MAC-VLAN interface corresponding to L3VNI's SVI. The SVI interface's MAC address can be interpreted as system-MAC and MAC-VLAN interface's MAC as anycast MAC.

To derive system-IP and anycast-IP, the default BGP instance's router-id is used as system-IP and the VxLAN interface's local tunnel IP as the anycast-IP.

User has an option to configure the system-IP and/or system-MAC value if the auto derived value is not preferred.

Note: By default, advertise-pip feature is enabled and user has an option to disable the feature via configuration CLI. Once the feature is disabled under bgp vrf instance or MAC-VLAN interface is not configured, all the routes follow the same behavior of using same next-hop and RMAC values.

advertise-pip [ip <addr> [mac <addr>]]

Enables or disables advertise-pip feature, specify system-IP and/or system-MAC parameters.

EVPN advertise-svi-ip

Typically, the SVI IP address is reused on VTEPs across multiple racks. However, if you have unique SVI IP addresses that you want to be reachable you can use the advertise-svi-ip option. This option advertises the SVI IP/MAC address as a type-2 route and eliminates the need for any flooding over VXLAN to reach the IP from a remote VTEP.

advertise-svi-ip

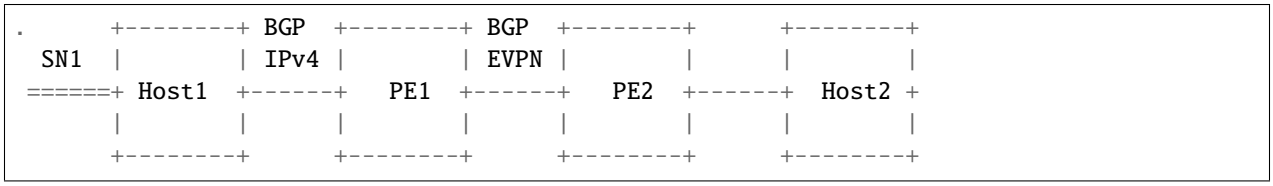
Note that you should not enable both the advertise-svi-ip and the advertise-default-gw at the same time.

EVPN Overlay Index Gateway IP

RFC https://datatracker.ietf.org/doc/html/rfc9136 explains the use of overlay indexes for recursive route resolution for EVPN type-5 route.

We support gateway IP overlay index. A gateway IP, advertised with EVPN prefix route, is used to find an EVPN MAC/IP route with its IP field same as the gateway IP. This MAC/IP entry provides the nexthop VTEP and the tunnel information required for the VxLAN encapsulation.

Functionality:



Consider above topology where prefix SN1 is connected behind host1. Host1 advertises SN1 to PE1 over BGP IPv4 session. PE1 advertises SN1 to PE2 using EVPN type-5 route with host1 IP as the gateway IP. PE1 also advertises Host1 MAC/IP as type-2 route which is used to resolve host1 gateway IP.

PE2 receives this type-5 route and imports it into the vrf based on route targets. BGP prefix imported into the vrf uses gateway IP as its BGP nexthop. This route is installed into zebra if following conditions are satisfied:

1. Gateway IP nexthop is L3 reachable.
2. PE2 has received EVPN type-2 route with IP field set to gateway IP.

Topology requirements:

1. This feature is supported for asymmetric routing model only. While sending packets to SN1, ingress PE (PE2) performs routing and egress PE (PE1) performs only bridging.
2. This feature supports only traditional(non vlan-aware) bridge model. Bridge interface associated with L2VNI is an L3 interface. i.e., this interface is configured with an address in the L2VNI subnet. Note that the gateway IP should also have an address in the same subnet.
3. As this feature works in asymmetric routing model, all L2VNIs and corresponding VxLAN and bridge interfaces should be present at all the PEs.
4. L3VNI configuration is required to generate and import EVPN type-5 routes. L3VNI VxLAN and bridge interfaces also should be present.

A PE can use one of the following two mechanisms to advertise an EVPN type-5 route with gateway IP.

1. CLI to add gateway IP while generating EVPN type-5 route from a BGP IPv4/IPv6 prefix:

advertise <ipv4|ipv6> unicast [gateway-ip]

When this CLI is configured for a BGP vrf under L2VPN EVPN address family, EVPN type-5 routes are generated for BGP prefixes in the vrf. Nexthop of the BGP prefix becomes the gateway IP of the corresponding type-5 route.

If the above command is configured without the "gateway-ip" keyword, type-5 routes are generated without overlay index.

2. Add gateway IP to EVPN type-5 route using a route-map:

set evpn gateway-ip <ipv4|ipv6> <addr>

When route-map with above set clause is applied as outbound policy in BGP, it will set the gateway-ip in EVPN type-5 NLRI.

Example configuration:

```
router bgp 100
 neighbor 192.168.0.1 remote-as 101
 !
 address-family ipv4 l2vpn evpn
  neighbor 192.168.0.1 route-map RMAP out
 exit-address-family
!
route-map RMAP permit 10
 set evpn gateway-ip 10.0.0.1
 set evpn gateway-ip 10::1
```

A PE that receives a type-5 route with gateway IP overlay index should have "enable-resolve-overlay-index" configuration enabled to recursively resolve the overlay index nexthop and install the prefix into zebra.

enable-resolve-overlay-index

Example configuration:

```
router bgp 65001
 bgp router-id 192.168.100.1
 no bgp ebgp-requires-policy
 neighbor 10.0.1.2 remote-as 65002
 !
 address-family l2vpn evpn
  neighbor 10.0.1.2 activate
```

(continues on next page)

(continued from previous page)

```
  advertise-all-vni
  enable-resolve-overlay-index
 exit-address-family
!
```

EVPN Multihoming

All-Active Multihoming is used for redundancy and load sharing. Servers are attached to two or more PEs and the links are bonded (link-aggregation). This group of server links is referred to as an Ethernet Segment.

Ethernet Segments

An Ethernet Segment can be configured by specifying a system-MAC and a local discriminator or a complete ESINAME against the bond interface on the PE (via zebra) -

evpn mh es-id <(1-16777215)|ESINAME>

evpn mh es-sys-mac X:X:X:X:X:X

The sys-mac and local discriminator are used for generating a 10-byte, Type-3 Ethernet Segment ID. ESINAME is a 10-byte, Type-0 Ethernet Segment ID - "00:AA:BB:CC:DD:EE:FF:GG:HH:II".

Type-1 (EAD-per-ES and EAD-per-EVI) routes are used to advertise the locally attached ESs and to learn off remote ESs in the network. Local Type-2/MAC-IP routes are also advertised with a destination ESI allowing for MAC-IP syncing between Ethernet Segment peers. Reference: RFC 7432, RFC 8365

EVPN-MH is intended as a replacement for MLAG or Anycast VTEPs. In multihoming each PE has an unique VTEP address which requires the introduction of a new dataplane construct, MAC-ECMP. Here a MAC/FDB entry can point to a list of remote PEs/VTEPs.

BUM handling

Type-4 (ESR) routes are used for Designated Forwarder (DF) election. DFs forward BUM traffic received via the overlay network. This implementation uses a preference based DF election specified by draft-ietf-bess-evpn-pref-df. The DF preference is configurable per-ES (via zebra) -

evpn mh es-df-pref (1-16777215)

BUM traffic is rxed via the overlay by all PEs attached to a server but only the DF can forward the de-capsulated traffic to the access port. To accommodate that non-DF filters are installed in the dataplane to drop the traffic.

Similarly traffic received from ES peers via the overlay cannot be forwarded to the server. This is split-horizon-filtering with local bias.

Knobs for interop

Some vendors do not send EAD-per-EVI routes. To interop with them we need to relax the dependency on EAD-per-EVI routes and activate a remote ES-PE based on just the EAD-per-ES route.

Note that by default we advertise and expect EAD-per-EVI routes.

disable-ead-evi-rx

disable-ead-evi-tx

Fast failover

As the primary purpose of EVPN-MH is redundancy keeping the failover efficient is a recurring theme in the implementation. Following sub-features have been introduced for the express purpose of efficient ES failovers.

- Layer-2 Nexthop Groups and MAC-ECMP via L2NHG.
- Host routes (for symmetric IRB) via L3NHG. On dataplanes that support layer3 nexthop groups the feature can be turned on via the following BGP config -

use-es-l3nhg

- Local ES (MAC/Neigh) failover via ES-redirect. On dataplanes that do not have support for ES-redirect the feature can be turned off via the following zebra config -

evpn mh redirect-off

Uplink/Core tracking

When all the underlay links go down the PE no longer has access to the VxLAN +overlay. To prevent blackholing of traffic the server/ES links are protodowned on the PE. A link can be setup for uplink tracking via the following zebra configuration -

evpn mh uplink

Proxy advertisements

To handle hitless upgrades support for proxy advertisement has been added as specified by draft-rbickhart-evpn-ip-mac-proxy-adv. This allows a PE (say PE1) to proxy advertise a MAC-IP rxed from an ES peer (say PE2). When the ES peer (PE2) goes down PE1 continues to advertise hosts learnt from PE2 for a holdtime during which it attempts to establish local reachability of the host. This holdtime is configurable via the following zebra commands -

evpn mh neigh-holdtime (0-86400)

evpn mh mac-holdtime (0-86400)

Startup delay

When a switch is rebooted we wait for a brief period to allow the underlay and EVPN network to converge before enabling the ESs. For this duration the ES bonds are held protodown. The startup delay is configurable via the following zebra command -

evpn mh startup-delay (0-3600)

EAD-per-ES fragmentation

The EAD-per-ES route carries the EVI route targets for all the broadcast domains associated with the ES. Depending on the EVI scale the EAD-per-ES route maybe fragmented.

The number of EVIs per-EAD route can be configured via the following BGP command -

[no] ead-es-frag evi-limit (1-1000)

Sample Configuration

```
!
router bgp 5556
 !
 address-family l2vpn evpn
  ead-es-frag evi-limit 200
 exit-address-family
 !
!
```

EAD-per-ES route-target

The EAD-per-ES route by default carries all the EVI route targets. Depending on EVI scale that can result in route fragmentation. In some cases it maybe necessary to avoid this fragmentation and that can be done via the following workaround - 1. Configure a single supplementary BD per-tenant VRF. This SBD needs to be provisioned on all EVPN PEs associated with the tenant-VRF. 2. Config the SBD's RT as the EAD-per-ES route's export RT.

Sample Configuration

```
!
router bgp 5556
 !
 address-family l2vpn evpn
  ead-es-route-target export 5556:1001
  ead-es-route-target export 5556:1004
  ead-es-route-target export 5556:1008
 exit-address-family
!
```

Support with VRF network namespace backend

It is possible to separate overlay networks contained in VXLAN interfaces from underlay networks by using VRFs. VRF-lite and VRF-netns backends can be used for that. In the latter case, it is necessary to set both bridge and vxlan interface in the same network namespace, as below example illustrates:

```
# linux shell
ip netns add vrf1
ip link add name vxlan101 type vxlan id 101 dstport 4789 dev eth0 local 10.1.1.1
ip link set dev vxlan101 netns vrf1
ip netns exec vrf1 ip link set dev lo up
ip netns exec vrf1 brctl addbr bridge101
ip netns exec vrf1 brctl addif bridge101 vxlan101
```

This makes it possible to separate not only layer 3 networks like VRF-lite networks. Also, VRF netns based make possible to separate layer 2 networks on separate VRF instances.

BGP Conditional Advertisement

The BGP conditional advertisement feature uses the `non-exist-map` or the `exist-map` and the `advertise-map` keywords of the neighbor advertise-map command in order to track routes by the route prefix.

non-exist-map

1. If a route prefix is not present in the output of non-exist-map command, then advertise the route specified by the advertise-map command.
2. If a route prefix is present in the output of non-exist-map command, then do not advertise the route specified by the addvertise-map command.

exist-map

1. If a route prefix is present in the output of exist-map command, then advertise the route specified by the advertise-map command.
2. If a route prefix is not present in the output of exist-map command, then do not advertise the route specified by the advertise-map command.

This feature is useful when some prefixes are advertised to one of its peers only if the information from the other peer is not present (due to failure in peering session or partial reachability etc).

The conditional BGP announcements are sent in addition to the normal announcements that a BGP router sends to its peer.

The conditional advertisement process is triggered by the BGP scanner process, which runs every 60 by default. This means that the maximum time for the conditional advertisement to take effect is the value of the process timer.

As an optimization, while the process always runs on each timer expiry, it determines whether or not the conditional advertisement policy or the routing table has changed; if neither have changed, no processing is necessary and the scanner exits early.

neighbor A.B.C.D advertise-map NAME [exist-map|non-exist-map] NAME
This command enables BGP scanner process to monitor routes specified by exist-map or non-exist-map command in BGP table and conditionally advertises the routes specified by advertise-map command.

bgp conditional-advertisement timer (5-240)
Set the period to rerun the conditional advertisement scanner process. The default is 60 seconds.

Sample Configuration

```
interface enp0s9
 ip address 10.10.10.2/24
!
interface enp0s10
 ip address 10.10.20.2/24
!
interface lo
 ip address 203.0.113.1/32
!
router bgp 2
 bgp log-neighbor-changes
 no bgp ebgp-requires-policy
 neighbor 10.10.10.1 remote-as 1
 neighbor 10.10.20.3 remote-as 3
 !
 address-family ipv4 unicast
  neighbor 10.10.10.1 soft-reconfiguration inbound
  neighbor 10.10.20.3 soft-reconfiguration inbound
  neighbor 10.10.20.3 advertise-map ADV-MAP non-exist-map EXIST-MAP
 exit-address-family
!
ip prefix-list DEFAULT seq 5 permit 192.0.2.5/32
ip prefix-list DEFAULT seq 10 permit 192.0.2.1/32
ip prefix-list EXIST seq 5 permit 10.10.10.10/32
ip prefix-list DEFAULT-ROUTE seq 5 permit 0.0.0.0/0
ip prefix-list IP1 seq 5 permit 10.139.224.0/20
!
bgp community-list standard DC-ROUTES seq 5 permit 64952:3008
bgp community-list standard DC-ROUTES seq 10 permit 64671:501
bgp community-list standard DC-ROUTES seq 15 permit 64950:3009
bgp community-list standard DEFAULT-ROUTE seq 5 permit 65013:200
!
route-map ADV-MAP permit 10
 match ip address prefix-list IP1
!
route-map ADV-MAP permit 20
 match community DC-ROUTES
!
route-map EXIST-MAP permit 10
 match community DEFAULT-ROUTE
 match ip address prefix-list DEFAULT-ROUTE
!
```

Sample Output

When default route is present in R2'2 BGP table, 10.139.224.0/20 and 192.0.2.1/32 are not advertised to R3.

```
Router2# show ip bgp
BGP table version is 20, local router ID is 203.0.113.1, vrf id 0
Default local pref 100, local AS 2
Status codes:  s suppressed, d damped, h history, * valid, > best, = multipath,
               i internal, r RIB-failure, S Stale, R Removed
Nexthop codes: @NNN nexthop's vrf id, < announce-nh-self
Origin codes:  i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

   Network          Next Hop            Metric LocPrf Weight Path
*> 0.0.0.0/0        10.10.10.1               0             0 1 i
*> 10.139.224.0/20  10.10.10.1               0             0 1 ?
*> 192.0.2.1/32     10.10.10.1               0             0 1 i
*> 192.0.2.5/32     10.10.10.1               0             0 1 i

Displayed  4 routes and 4 total paths
Router2# show ip bgp neighbors 10.10.20.3

!--- Output suppressed.

For address family: IPv4 Unicast
Update group 7, subgroup 7
Packet Queue length 0
Inbound soft reconfiguration allowed
Community attribute sent to this neighbor(all)
Condition NON_EXIST, Condition-map *EXIST-MAP, Advertise-map *ADV-MAP, status: Withdraw
0 accepted prefixes

!--- Output suppressed.

Router2# show ip bgp neighbors 10.10.20.3 advertised-routes
BGP table version is 20, local router ID is 203.0.113.1, vrf id 0
Default local pref 100, local AS 2
Status codes:  s suppressed, d damped, h history, * valid, > best, = multipath,
            i internal, r RIB-failure, S Stale, R Removed
Nexthop codes: @NNN nexthop's vrf id, < announce-nh-self
Origin codes:  i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

   Network          Next Hop            Metric LocPrf Weight Path
*> 0.0.0.0/0        0.0.0.0                                0 1 i
*> 192.0.2.5/32     0.0.0.0                                0 1 i

Total number of prefixes 2
```

When default route is not present in R2'2 BGP table, 10.139.224.0/20 and 192.0.2.1/32 are advertised to R3.

```
Router2# show ip bgp
BGP table version is 21, local router ID is 203.0.113.1, vrf id 0
```

(continues on next page)

(continued from previous page)

```
Default local pref 100, local AS 2
Status codes:  s suppressed, d damped, h history, * valid, > best, = multipath,
               i internal, r RIB-failure, S Stale, R Removed
Nexthop codes: @NNN nexthop's vrf id, < announce-nh-self
Origin codes:  i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

   Network          Next Hop            Metric LocPrf Weight Path
*> 10.139.224.0/20  10.10.10.1               0             0 1 ?
*> 192.0.2.1/32     10.10.10.1               0             0 1 i
*> 192.0.2.5/32     10.10.10.1               0             0 1 i

Displayed  3 routes and 3 total paths

Router2# show ip bgp neighbors 10.10.20.3

!--- Output suppressed.

For address family: IPv4 Unicast
Update group 7, subgroup 7
Packet Queue length 0
Inbound soft reconfiguration allowed
Community attribute sent to this neighbor(all)
Condition NON_EXIST, Condition-map *EXIST-MAP, Advertise-map *ADV-MAP, status: Advertise
0 accepted prefixes

!--- Output suppressed.

Router2# show ip bgp neighbors 10.10.20.3 advertised-routes
BGP table version is 21, local router ID is 203.0.113.1, vrf id 0
Default local pref 100, local AS 2
Status codes:  s suppressed, d damped, h history, * valid, > best, = multipath,
               i internal, r RIB-failure, S Stale, R Removed
Nexthop codes: @NNN nexthop's vrf id, < announce-nh-self
Origin codes:  i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

   Network          Next Hop            Metric LocPrf Weight Path
*> 10.139.224.0/20  0.0.0.0                              0 1 ?
*> 192.0.2.1/32     0.0.0.0                              0 1 i
*> 192.0.2.5/32     0.0.0.0                              0 1 i

Total number of prefixes 3
Router2#
```

Debugging

show debug
Show all enabled debugs.

show bgp listeners
Display Listen sockets and the vrf that created them. Useful for debugging of when listen is not working and this is considered a developer debug statement.

debug bgp allow-martian
Enable or disable BGP accepting martian nexthops from a peer. Please note this is not an actual debug command and this command is also being deprecated and will be removed soon. The new command is *bgp allow-martian-nexthop*

debug bgp bfd
Enable or disable debugging for BFD events. This will show BFD integration library messages and BGP BFD integration messages that are mostly state transitions and validation problems.

debug bgp conditional-advertisement
Enable or disable debugging of BGP conditional advertisement.

debug bgp neighbor-events
Enable or disable debugging for neighbor events. This provides general information on BGP events such as peer connection / disconnection, session establishment / teardown, and capability negotiation.

debug bgp updates
Enable or disable debugging for BGP updates. This provides information on BGP UPDATE messages transmitted and received between local and remote instances.

debug bgp keepalives
Enable or disable debugging for BGP keepalives. This provides information on BGP KEEPALIVE messages transmitted and received between local and remote instances.

debug bgp bestpath <A.B.C.D/M|X:X::X:X/M>
Enable or disable debugging for bestpath selection on the specified prefix.

debug bgp nht
Enable or disable debugging of BGP nexthop tracking.

debug bgp update-groups
Enable or disable debugging of dynamic update groups. This provides general information on group creation, deletion, join and prune events.

debug bgp zebra
Enable or disable debugging of communications between *bgpd* and *zebra*.

Dumping Messages and Routing Tables

dump bgp all PATH [INTERVAL]

dump bgp all-et 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]
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
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.

clear bgp [ipv4|ipv6] [unicast] PEER|* message-stats
Clear BGP message statistics for a specified peer or for all peers, optionally filtered by activated address-family and sub-address-family.

The following are available in the `router bgp` mode:

write-quanta (1-64)
BGP message Tx I/O is vectored. This means that multiple packets are written to the peer socket at the same time each I/O cycle, in order to minimize system call overhead. This value controls how many are written at a time. Under certain load conditions, reducing this value could make peer traffic less 'bursty'. In practice, leave this settings on the default (64) unless you truly know what you are doing.

read-quanta (1-10)
Unlike Tx, BGP Rx traffic is not vectored. Packets are read off the wire one at a time in a loop. This setting controls how many iterations the loop runs for. As with write-quanta, it is best to leave this setting on the default.

The following command is available in `config` mode as well as in the `router bgp` mode:

bgp graceful-shutdown
The purpose of this command is to initiate BGP Graceful Shutdown which is described in **RFC 8326**. The use case for this is to minimize or eliminate the amount of traffic loss in a network when a planned maintenance activity such as software upgrade or hardware replacement is to be performed on a router. The feature works by re-announcing routes to eBGP peers with the GRACEFUL_SHUTDOWN community included. Peers are then expected to treat such paths with the lowest preference. This happens automatically on a receiver running FRR;

with other routing protocol stacks, an inbound policy may have to be configured. In FRR, triggering graceful shutdown also results in announcing a LOCAL_PREF of 0 to iBGP peers.

Graceful shutdown can be configured per BGP instance or globally for all of BGP. These two options are mutually exclusive. The no form of the command causes graceful shutdown to be stopped, and routes will be re-announced without the GRACEFUL_SHUTDOWN community and/or with the usual LOCAL_PREF value. Note that if this option is saved to the startup configuration, graceful shutdown will remain in effect across restarts of *bgpd* and will need to be explicitly disabled.

bgp input-queue-limit (1-4294967295)
Set the BGP Input Queue limit for all peers when messaging parsing. Increase this only if you have the memory to handle large queues of messages at once.

bgp output-queue-limit (1-4294967295)
Set the BGP Output Queue limit for all peers when messaging parsing. Increase this only if you have the memory to handle large queues of messages at once.

3.3.4 Displaying BGP Information

The following four commands display the IPv6 and IPv4 routing tables, depending on whether or not the `ip` keyword is used. Actually, `show ip bgp` command was used on older *Quagga* routing daemon project, while `show bgp` command is the new format. The choice has been done to keep old format with IPv4 routing table, while new format displays IPv6 routing table.

show ip bgp [all] [wide|json [detail]]

show ip bgp A.B.C.D [json]

show bgp [all] [wide|json [detail]]

show bgp X:X::X:X [json]
These commands display BGP routes. When no route is specified, the default is to display all BGP routes.

```
BGP table version is 0, local router ID is 10.1.1.1
   Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
   Origin codes: i - IGP, e - EGP, ? - incomplete

Network    Next Hop      Metric LocPrf Weight Path
   \*> 1.1.1.1/32       0.0.0.0      0   32768 i

   Total number of prefixes 1
```

If `wide` option is specified, then the prefix table's width is increased to fully display the prefix and the nexthop.

This is especially handy dealing with IPv6 prefixes and if `[no] bgp default show-nexthop-hostname` is enabled.

If `all` option is specified, `ip` keyword is ignored, show bgp all and show ip bgp all commands display routes for all AFIs and SAFIs.

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

If `detail` option is specified after `json`, more verbose JSON output will be displayed.

Some other commands provide additional options for filtering the output.

show [ip] bgp regexp LINE
This command displays BGP routes using AS path regular expression (*BGP Regular Expressions*).

show [ip] bgp [all] summary [wide] [json]
Show a bgp peer summary for the specified address family.

The old command structure `show ip bgp` may be removed in the future and should no longer be used. In order to reach the other BGP routing tables other than the IPv6 routing table given by `show bgp`, the new command structure is extended with `show bgp [afi] [safi]`.

`wide` option gives more output like `LocalAS` and extended `Desc` to 64 characters.

```
exit1# show ip bgp summary wide

IPv4 Unicast Summary (VRF default):
BGP router identifier 192.168.100.1, local AS number 65534 vrf-id 0
BGP table version 3
RIB entries 5, using 920 bytes of memory
Peers 1, using 27 KiB of memory

Neighbor        V         AS    LocalAS   MsgRcvd   MsgSent   TblVer  InQ OutQ␣
↪ Up/Down State/PfxRcd   PfxSnt Desc
192.168.0.2     4      65030        123        15        22        0    0    0␣
↪00:07:00            0        1 us-east1-rs1.frrouting.org

Total number of neighbors 1
exit1#
```

If PfxRcd and/or PfxSnt is shown as `(Policy)`, that means that the EBGP default policy is turned on, but you don't have any filters applied for incoming/outgoing directions.

See also:

Require policy on EBGP

show bgp [afi] [safi] [all] [wide|json]

show bgp vrfs [<VRFNAME$vrf_name>] [json]
The command displays all bgp vrf instances basic info like router-id, configured and established neighbors, evpn related basic info like l3vni, router-mac, vxlan-interface. User can get that information as JSON format when `json` keyword at the end of cli is presented.

```
torc-11# show bgp vrfs
Type  Id     routerId          #PeersCfg  #PeersEstb  Name
             L3-VNI            RouterMAC              Interface
DFLT  0      17.0.0.6          3          3           default
             0                 00:00:00:00:00:00      unknown
 VRF  21     17.0.0.6          0          0           sym_1
             8888              34:11:12:22:22:01      vlan4034_l3
 VRF  32     17.0.0.6          0          0           sym_2
             8889              34:11:12:22:22:01      vlan4035_l3

Total number of VRFs (including default): 3
```

show bgp [<ipv4|ipv6> <unicast|multicast|vpn|labeled-unicast|flowspec> | l2vpn evpn]
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 l2vpn evpn route [type <macip|2|multicast|3|es|4|prefix|5>]
EVPN prefixes can also be filtered by EVPN route type.

show bgp l2vpn evpn route [detail] [type <ead|1|macip|2|multicast|3|es|4|prefix|5>] self-originate [jso
Display self-originated EVPN prefixes which can also be filtered by EVPN route type.

show bgp vni <all|VNI> [vtep VTEP] [type <ead|1|macip|2|multicast|3>] [<detail|json>]
Display per-VNI EVPN routing table in bgp. Filter route-type, vtep, or VNI.

show bgp [afi] [safi] [all] summary [json]
Show a bgp peer summary for the specified address family, and subsequent address-family.

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

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

show bgp [afi] [safi] [all] summary neighbor [PEER] [json]
Show a bgp summary for the specified peer, address family, and subsequent address-family. The neighbor filter can be used in combination with the failed, established filters.

show bgp [afi] [safi] [all] summary remote-as <internal|external|ASN> [json]
Show a bgp peer summary for the specified remote-as ASN or type (`internal` for iBGP and `external` for eBGP sessions), address family, and subsequent address-family. The remote-as filter can be used in combination with the failed, established filters.

show bgp [afi] [safi] [all] summary terse [json]
Shorten the output. Do not show the following information about the BGP instances: the number of RIB entries, the table version and the used memory. The `terse` option can be used in combination with the remote-as, neighbor, failed and established filters, and with the `wide` option as well.

show bgp [afi] [safi] [neighbor [PEER] [routes|advertised-routes|received-routes] [<A.B.C.D/M|X:X::X:X/M> | detail] [json]
This command shows information on a specific BGP peer of the relevant afi and safi selected.

The `routes` keyword displays only routes in this address-family's BGP table that were received by this peer and accepted by inbound policy.

The `advertised-routes` keyword displays only the routes in this address-family's BGP table that were permitted by outbound policy and advertised to to this peer.

The `received-routes` keyword displays all routes belonging to this address-family (prior to inbound policy) that were received by this peer.

If a specific prefix is specified, the detailed version of that prefix will be displayed.

If `detail` option is specified, the detailed version of all routes will be displayed. The same format as `show [ip] bgp [afi] [safi] PREFIX` will be used, but for the whole table of received, advertised or filtered prefixes.

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

show bgp [<view|vrf> VIEWVRFNAME] [afi] [safi] neighbors PEER received prefix-filter [json]
Display Address Prefix ORFs received from this peer.

show bgp [afi] [safi] [all] dampening dampened-paths [wide|json]
Display paths suppressed due to dampening of the selected afi and safi selected.

show bgp [afi] [safi] [all] dampening flap-statistics [wide|json]
Display flap statistics of routes of the selected afi and safi selected.

show bgp [afi] [safi] [all] dampening parameters [json]
Display details of configured dampening parameters of the selected afi and safi.

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

show bgp [afi] [safi] [all] version (1-4294967295) [wide|json]
Display prefixes with matching version numbers. The version number and above having prefixes will be listed here.

It helps to identify which prefixes were installed at some point.

Here is an example of how to check what prefixes were installed starting with an arbitrary version:

```
# vtysh -c 'show bgp ipv4 unicast json' | jq '.tableVersion'
9
# vtysh -c 'show ip bgp version 9 json' | jq -r '.routes | keys[]'
192.168.3.0/24
# vtysh -c 'show ip bgp version 8 json' | jq -r '.routes | keys[]'
192.168.2.0/24
192.168.3.0/24
```

show bgp [afi] [safi] statistics
Display statistics of routes of the selected afi and safi.

show bgp statistics-all
Display statistics of routes of all the afi and safi.

show [ip] bgp [afi] [safi] [all] cidr-only [wide|json]
Display routes with non-natural netmasks.

show [ip] bgp [afi] [safi] [all] prefix-list WORD [wide|json]
Display routes that match the specified prefix-list.

If `wide` option is specified, then the prefix table's width is increased to fully display the prefix and the nexthop.

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

show [ip] bgp [afi] [safi] [all] access-list WORD [wide|json]
Display routes that match the specified access-list.

show [ip] bgp [afi] [safi] [all] filter-list WORD [wide|json]
Display routes that match the specified AS-Path filter-list.

If `wide` option is specified, then the prefix table's width is increased to fully display the prefix and the nexthop.

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

show [ip] bgp [afi] [safi] [all] route-map WORD [wide|json]
Display routes that match the specified route-map.

If `wide` option is specified, then the prefix table's width is increased to fully display the prefix and the nexthop.

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

show [ip] bgp [afi] [safi] [all] <A.B.C.D/M|X:X::X:X/M> longer-prefixes [wide|json]
Displays the specified route and all more specific routes.

If `wide` option is specified, then the prefix table's width is increased to fully display the prefix and the nexthop.

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

show [ip] bgp [afi] [safi] [all] self-originate [wide|json]
Display self-originated routes.

If `wide` option is specified, then the prefix table's width is increased to fully display the prefix and the nexthop.

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

show [ip] bgp [afi] [safi] [all] neighbors A.B.C.D [advertised-routes|received-routes|filtered-routes] [<A.B.C.D/M|X:X::X:X/M> | detail] [json|wide]

Display the routes advertised to a BGP neighbor or received routes from neighbor or filtered routes received from neighbor based on the option specified.

If `wide` option is specified, then the prefix table's width is increased to fully display the prefix and the nexthop.

This is especially handy dealing with IPv6 prefixes and if `[no] bgp default show-nexthop-hostname` is enabled.

If `all` option is specified, `ip` keyword is ignored and, routes displayed for all AFIs and SAFIs. if afi is specified, with `all` option, routes will be displayed for each SAFI in the selcted AFI

If a specific prefix is specified, the detailed version of that prefix will be displayed.

If `detail` option is specified, the detailed version of all routes will be displayed. The same format as `show [ip] bgp [afi] [safi] PREFIX` will be used, but for the whole table of received, advertised or filtered prefixes.

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

show [ip] bgp [afi] [safi] [all] detail-routes

Display the detailed version of all routes. The same format as using `show [ip] bgp [afi] [safi] PREFIX`, but for the whole BGP table.

If `all` option is specified, `ip` keyword is ignored and, routes displayed for all AFIs and SAFIs.

If `afi` is specified, with `all` option, routes will be displayed for each SAFI in the selected AFI.

show [ip] bgp [<view|vrf> VIEWVRFNAME] [afi] [safi] detail [json]

Display the detailed version of all routes from the specified bgp vrf table for a given afi + safi.

If no vrf is specified, then it is assumed as a default vrf and routes are displayed from default vrf table.

If `all` option is specified as vrf name, then all bgp vrf tables routes from a given afi+safi are displayed in the detailed output of routes.

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

Following are sample output for few examples of how to use this command.

```
torm-23# sh bgp ipv4 unicast detail (OR) sh bgp vrf default ipv4 unicast detail

!--- Output suppressed.

BGP routing table entry for 172.16.16.1/32
Paths: (1 available, best #1, table default)
  Not advertised to any peer
  Local, (Received from a RR-client)
    172.16.16.1 (metric 20) from torm-22(172.16.16.1) (192.168.0.10)
      Origin IGP, metric 0, localpref 100, valid, internal
      Last update: Fri May  8 12:54:05 2023
BGP routing table entry for 172.16.16.2/32
Paths: (1 available, best #1, table default)
  Not advertised to any peer
  Local
    0.0.0.0 from 0.0.0.0 (172.16.16.2)
      Origin incomplete, metric 0, weight 32768, valid, sourced, bestpath-from-AS Local,␣
↪best (First path received)
      Last update: Wed May  8 12:54:41 2023
```

(continues on next page)

(continued from previous page)

```
Displayed  2 routes and 2 total paths
```

```
torm-23# sh bgp vrf all detail

Instance default:

!--- Output suppressed.

BGP routing table entry for 172.16.16.1/32
Paths: (1 available, best #1, table default)
  Not advertised to any peer
  Local, (Received from a RR-client)
    172.16.16.1 (metric 20) from torm-22(172.16.16.1) (192.168.0.10)
      Origin IGP, metric 0, localpref 100, valid, internal
      Last update: Fri May  8 12:44:05 2023
BGP routing table entry for 172.16.16.2/32
Paths: (1 available, best #1, table default)
  Not advertised to any peer
  Local
    0.0.0.0 from 0.0.0.0 (172.16.16.2)
      Origin incomplete, metric 0, weight 32768, valid, sourced, bestpath-from-AS Local,␣
↪best (First path received)
      Last update: Wed May  8 12:45:01 2023

Displayed  2 routes and 2 total paths

Instance vrf3:

!--- Output suppressed.

BGP routing table entry for 192.168.0.2/32
Paths: (1 available, best #1, vrf vrf3)
  Not advertised to any peer
  Imported from 172.16.16.1:12:[2]:[0]:[48]:[00:02:00:00:00:58]:[32]:[192.168.0.2], VNI␣
↪1008/4003
  Local
    172.16.16.1 from torm-22(172.16.16.1) (172.16.16.1) announce-nh-self
      Origin IGP, localpref 100, valid, internal, bestpath-from-AS Local, best (First␣
↪path received)
      Extended Community: RT:65000:1008 ET:8 Rmac:00:02:00:00:00:58
      Last update: Fri May  8 02:41:55 2023
BGP routing table entry for 192.168.1.2/32
Paths: (1 available, best #1, vrf vrf3)
  Not advertised to any peer
  Imported from 172.16.16.1:13:[2]:[0]:[48]:[00:02:00:00:00:58]:[32]:[192.168.1.2], VNI␣
↪1009/4003
  Local
    172.16.16.1 from torm-22(172.16.16.1) (172.16.16.1) announce-nh-self
      Origin IGP, localpref 100, valid, internal, bestpath-from-AS Local, best (First␣
↪path received)
      Extended Community: RT:65000:1009 ET:8 Rmac:00:02:00:00:00:58
```

(continues on next page)

(continued from previous page)

```
      Last update: Fri May  8 02:41:55 2023

Displayed  2 routes and 2 total paths
```

```
torm-23# sh bgp vrf vrf3 ipv4 unicast detail

!--- Output suppressed.

BGP routing table entry for 192.168.0.2/32
Paths: (1 available, best #1, vrf vrf3)
  Not advertised to any peer
  Imported from 172.16.16.1:12:[2]:[0]:[48]:[00:02:00:00:00:58]:[32]:[192.168.0.2], VNI␣
↪1008/4003
  Local
    172.16.16.1 from torm-22(172.16.16.1) (172.16.16.1) announce-nh-self
      Origin IGP, localpref 100, valid, internal, bestpath-from-AS Local, best (First␣
↪path received)
      Extended Community: RT:65000:1008 ET:8 Rmac:00:02:00:00:00:58
      Last update: Fri May  8 02:23:35 2023
BGP routing table entry for 192.168.1.2/32
Paths: (1 available, best #1, vrf vrf3)
  Not advertised to any peer
  Imported from 172.16.16.1:13:[2]:[0]:[48]:[00:02:00:00:00:58]:[32]:[192.168.1.2], VNI␣
↪1009/4003
  Local
    172.16.16.1 from torm-22(172.16.16.1) (172.16.16.1) announce-nh-self
      Origin IGP, localpref 100, valid, internal, bestpath-from-AS Local, best (First␣
↪path received)
      Extended Community: RT:65000:1009 ET:8 Rmac:00:02:00:00:00:58
      Last update: Fri May  8 02:23:55 2023

Displayed  2 routes and 2 total paths
```

Displaying Routes by Community Attribute

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

show [ip] bgp <ipv4|ipv6> [all] community [wide|json]

show [ip] bgp <ipv4|ipv6> [all] community COMMUNITY [wide|json]

show [ip] bgp <ipv4|ipv6> [all] community COMMUNITY exact-match [wide|json]
: These commands display BGP routes which have the community attribute. attribute. When `COMMUNITY` is specified, BGP routes that match that community are displayed. When *exact-match* is specified, it display only routes that have an exact match.

show [ip] bgp <ipv4|ipv6> community-list WORD [json]

show [ip] bgp <ipv4|ipv6> community-list WORD exact-match [json]
: These commands display BGP routes for the address family specified that match the specified community list. When *exact-match* is specified, it displays only routes that have an exact match.

  If `wide` option is specified, then the prefix table's width is increased to fully display the prefix and the nexthop.

This is especially handy dealing with IPv6 prefixes and if `[no] bgp default show-nexthop-hostname` is enabled.

If `all` option is specified, `ip` keyword is ignored and, routes displayed for all AFIs and SAFIs. if afi is specified, with `all` option, routes will be displayed for each SAFI in the selcted AFI

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

show bgp labelpool <chunks|inuse|ledger|requests|summary> [json]
These commands display information about the BGP labelpool used for the association of MPLS labels with routes for L3VPN and Labeled Unicast

If `chunks` option is specified, output shows the current list of label chunks granted to BGP by Zebra, indicating the start and end label in each chunk

If `inuse` option is specified, output shows the current inuse list of label to prefix mappings

If `ledger` option is specified, output shows ledger list of all label requests made per prefix

If `requests` option is specified, output shows current list of label requests which have not yet been fulfilled by the labelpool

If `summary` option is specified, output is a summary of the counts for the chunks, inuse, ledger and requests list along with the count of outstanding chunk requests to Zebra and the number of zebra reconnects that have happened

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

Displaying Routes by Large Community Attribute

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

show [ip] bgp <ipv4|ipv6> large-community

show [ip] bgp <ipv4|ipv6> large-community LARGE-COMMUNITY

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

show [ip] bgp <ipv4|ipv6> large-community LARGE-COMMUNITY json
These commands display BGP routes which have the large community attribute. attribute. When `LARGE-COMMUNITY` is specified, BGP routes that match that large community are displayed. When *exact-match* is specified, it display only routes that have an exact match. When *json* is specified, it display routes in json format.

show [ip] bgp <ipv4|ipv6> large-community-list WORD

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

show [ip] bgp <ipv4|ipv6> large-community-list WORD json
These commands display BGP routes for the address family specified that match the specified large community list. When *exact-match* is specified, it displays only routes that have an exact match. When *json* is specified, it display routes in json format.

Displaying Routes by AS Path

show bgp ipv4|ipv6 regexp LINE
This commands displays BGP routes that matches a regular expression *line* (*BGP Regular Expressions*).

show [ip] bgp ipv4 vpn

show [ip] bgp ipv6 vpn
Print active IPV4 or IPV6 routes advertised via the VPN SAFI.

show bgp ipv4 vpn summary

show bgp ipv6 vpn summary
Print a summary of neighbor connections for the specified AFI/SAFI combination.

Displaying Routes by Route Distinguisher

show bgp [<ipv4|ipv6> vpn | l2vpn evpn [route]] rd <all|RD>
For L3VPN and EVPN address-families, routes can be displayed on a per-RD (Route Distinguisher) basis or for all RD's.

show bgp l2vpn evpn rd <all|RD> [overlay | tags]
Use the `overlay` or `tags` keywords to display the overlay/tag information about the EVPN prefixes in the selected Route Distinguisher.

show bgp l2vpn evpn route rd <all|RD> mac <MAC> [ip <MAC>] [json]
For EVPN Type 2 (macip) routes, a MAC address (and optionally an IP address) can be supplied to the command to only display matching prefixes in the specified RD.

Displaying Update Group Information

show bgp update-groups [advertise-queue|advertised-routes|packet-queue]
Display Information about each individual update-group being used. If SUBGROUP-ID is specified only display about that particular group. If advertise-queue is specified the list of routes that need to be sent to the peers in the update-group is displayed, advertised-routes means the list of routes we have sent to the peers in the update-group and packet-queue specifies the list of packets in the queue to be sent.

show bgp update-groups statistics
Display Information about update-group events in FRR.

Displaying Nexthop Information

show [ip] bgp [<view|vrf> VIEWVRFNAME] nexthop ipv4 [A.B.C.D] [detail] [json]

show [ip] bgp [<view|vrf> VIEWVRFNAME] nexthop ipv6 [X:X::X:X] [detail] [json]

show [ip] bgp [<view|vrf> VIEWVRFNAME] nexthop [<A.B.C.D|X:X::X:X>] [detail] [json]

show [ip] bgp <view|vrf> all nexthop [json]
Display information about nexthops to bgp neighbors. If a certain nexthop is specified, also provides information about paths associated with the nexthop. With detail option provides information about gates of each nexthop.

show [ip] bgp [<view|vrf> VIEWVRFNAME] import-check-table [detail] [json]
Display information about nexthops from table that is used to check network's existence in the rib for network statements.

Segment-Routing IPv6

show bgp segment-routing srv6
This command displays information about SRv6 L3VPN in bgpd. Specifically, what kind of Locator is being used, and its Locator chunk information. And the SID of the SRv6 Function that is actually managed on bgpd. In the following example, bgpd is using a Locator named loc1, and two SRv6 Functions are managed to perform VPNv6 VRF redirect for vrf10 and vrf20.

```
router# show bgp segment-routing srv6
locator_name: loc1
locator_chunks:
- 2001:db8:1:1::/64
functions:
- sid: 2001:db8:1:1::100
  locator: loc1
- sid: 2001:db8:1:1::200
  locator: loc1
bgps:
- name: default
  vpn_policy[AFI_IP].tovpn_sid: none
  vpn_policy[AFI_IP6].tovpn_sid: none
- name: vrf10
  vpn_policy[AFI_IP].tovpn_sid: none
  vpn_policy[AFI_IP6].tovpn_sid: 2001:db8:1:1::100
- name: vrf20
  vpn_policy[AFI_IP].tovpn_sid: none
  vpn_policy[AFI_IP6].tovpn_sid: 2001:db8:1:1::200
```

AS-notation support

By default, the ASN value output follows how the BGP ASN instance is expressed in the configuration. Three as-notation outputs are available:

- plain output: both AS4B and AS2B use a single number. ` router bgp 65536`.
- dot output: AS4B values are using two numbers separated by a period. *router bgp 1.1* means that the AS number is 65536.
- dot+ output: AS2B and AS4B values are using two numbers separated by a period. *router bgp 0.5* means that the AS number is 5.

The below option permits forcing the as-notation output:

router bgp ASN as-notation dot|dot+|plain
The chosen as-notation format will override the BGP ASN output.

3.3.5 Route Reflector

BGP routers connected inside the same AS through BGP belong to an internal BGP session, or IBGP. In order to prevent routing table loops, IBGP does not advertise IBGP-learned routes to other routers in the same session. As such, IBGP requires a full mesh of all peers. For large networks, this quickly becomes unscalable. Introducing route reflectors removes the need for the full-mesh.

When route reflectors are configured, these will reflect the routes announced by the peers configured as clients. A route reflector client is configured with:

neighbor PEER route-reflector-client

To avoid single points of failure, multiple route reflectors can be configured.

A cluster is a collection of route reflectors and their clients, and is used by route reflectors to avoid looping.

bgp cluster-id A.B.C.D

bgp no-rib

To set and unset the BGP daemon `-n` / `--no_kernel` options during runtime to disable BGP route installation to the RIB (Zebra), the `[no] bgp no-rib` commands can be used;

Please note that setting the option during runtime will withdraw all routes in the daemons RIB from Zebra and unsetting it will announce all routes in the daemons RIB to Zebra. If the option is passed as a command line argument when starting the daemon and the configuration gets saved, the option will persist unless removed from the configuration with the negating command prior to the configuration write operation. At this point in time non SAFI_UNICAST BGP data is not properly withdrawn from zebra when this command is issued.

bgp allow-martian-nexthop

When a peer receives a martian nexthop as part of the NLRI for a route permit the nexthop to be used as such, instead of rejecting and resetting the connection.

bgp send-extra-data zebra

This command turns on the ability of BGP to send extra data to zebra. Currently, it's the AS-Path, communities, and the path selection reason. The default behavior in BGP is not to send this data. If the routes were sent to zebra and the option is changed, bgpd doesn't reinstall the routes to comply with the new setting.

bgp session-dscp (0-63)

This command allows bgp to control, at a global level, the TCP dscp values in the TCP header.

3.3.6 Suppressing routes not installed in FIB

The FRR implementation of BGP advertises prefixes learnt from a peer to other peers even if the routes do not get installed in the FIB. There can be scenarios where the hardware tables in some of the routers (along the path from the source to destination) is full which will result in all routes not getting installed in the FIB. If these routes are advertised to the downstream routers then traffic will start flowing and will be dropped at the intermediate router.

The solution is to provide a configurable option to check for the FIB install status of the prefixes and advertise to peers if the prefixes are successfully installed in the FIB. The advertisement of the prefixes are suppressed if it is not installed in FIB.

The following conditions apply will apply when checking for route installation status in FIB:

1. The advertisement or suppression of routes based on FIB install status applies only for newly learnt routes from peer (routes which are not in BGP local RIB).

2. If the route received from peer already exists in BGP local RIB and route attributes have changed (best path changed), the old path is deleted and new path is installed in FIB. The FIB install status will not have any effect. Therefore only when the route is received first time the checks apply.
3. The feature will not apply for routes learnt through other means like redistribution to bgp from other protocols. This is applicable only to peer learnt routes.
4. If a route is installed in FIB and then gets deleted from the dataplane, then routes will not be withdrawn from peers. This will be considered as dataplane issue.
5. The feature will slightly increase the time required to advertise the routes to peers since the route install status needs to be received from the FIB
6. If routes are received by the peer before the configuration is applied, then the bgp sessions need to be reset for the configuration to take effect.
7. If the route which is already installed in dataplane is removed for some reason, sending withdraw message to peers is not currently supported.

bgp suppress-fib-pending
: This command is applicable at the global level and at an individual bgp level. If applied at the global level all bgp instances will wait for fib installation before announcing routes and there is no way to turn it off for a particular bgp vrf.

3.3.7 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.

3.3.8 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 (^|[,{}()]|$).

3.3.9 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
```

(continues on next page)

(continued from previous page)

```
 address-family ipv4 unicast
  network 10.123.456.0/24
  network 10.123.456.128/25 route-map rm-no-export
  neighbor upstream route-map rm-upstream-out out
  neighbor cust route-map rm-cust-in in
  neighbor cust route-map rm-cust-out out
  neighbor cust send-community both
  neighbor peer route-map rm-peer-in in
  neighbor peer route-map rm-peer-out out
  neighbor peer send-community both
  neighbor 10.3.1.1 prefix-list pl-cust1-network in
  neighbor 10.4.1.1 prefix-list pl-cust2-network in
  neighbor 10.5.1.1 prefix-list pl-peer1-network in
  neighbor 10.6.1.1 prefix-list pl-peer2-network in
 exit-address-family
!
ip prefix-list pl-default permit 0.0.0.0/0
!
ip prefix-list pl-upstream-peers permit 10.1.1.1/32
ip prefix-list pl-upstream-peers permit 10.2.1.1/32
!
ip prefix-list pl-cust1-network permit 10.3.1.0/24
ip prefix-list pl-cust1-network permit 10.3.2.0/24
!
ip prefix-list pl-cust2-network permit 10.4.1.0/24
!
ip prefix-list pl-peer1-network permit 10.5.1.0/24
ip prefix-list pl-peer1-network permit 10.5.2.0/24
ip prefix-list pl-peer1-network permit 192.168.0.0/24
!
ip prefix-list pl-peer2-network permit 10.6.1.0/24
ip prefix-list pl-peer2-network permit 10.6.2.0/24
ip prefix-list pl-peer2-network permit 192.168.1.0/24
ip prefix-list pl-peer2-network permit 192.168.2.0/24
ip prefix-list pl-peer2-network permit 172.16.1/24
!
bgp as-path access-list seq 5 asp-own-as permit ^$
bgp as-path access-list seq 10 asp-own-as permit _64512_
!
! ###################################################################
! Match communities we provide actions for, on routes receives from
! customers. Communities values of <our-ASN>:X, with X, have actions:
!
! 100 - blackhole the prefix
! 200 - set no_export
! 300 - advertise only to other customers
! 400 - advertise only to upstreams
! 500 - set no_export when advertising to upstreams
! 2X00 - set local_preference to X00
!
! blackhole the prefix of the route
```

(continues on next page)

(continued from previous page)

```
bgp community-list standard cm-blackhole permit 64512:100
!
! set no-export community before advertising
bgp community-list standard cm-set-no-export permit 64512:200
!
! advertise only to other customers
bgp community-list standard cm-cust-only permit 64512:300
!
! advertise only to upstreams
bgp community-list standard cm-upstream-only permit 64512:400
!
! advertise to upstreams with no-export
bgp community-list standard cm-upstream-noexport permit 64512:500
!
! set local-pref to least significant 3 digits of the community
bgp community-list standard cm-prefmod-100 permit 64512:2100
bgp community-list standard cm-prefmod-200 permit 64512:2200
bgp community-list standard cm-prefmod-300 permit 64512:2300
bgp community-list standard cm-prefmod-400 permit 64512:2400
bgp community-list expanded cme-prefmod-range permit 64512:2...
!
! Informational communities
!
! 3000 - learned from upstream
! 3100 - learned from customer
! 3200 - learned from peer
!
bgp community-list standard cm-learnt-upstream permit 64512:3000
bgp community-list standard cm-learnt-cust permit 64512:3100
bgp community-list standard cm-learnt-peer permit 64512:3200
!
! ###################################################################
! Utility route-maps
!
! These utility route-maps generally should not used to permit/deny
! routes, i.e. they do not have meaning as filters, and hence probably
! should be used with 'on-match next'. These all finish with an empty
! permit entry so as not interfere with processing in the caller.
!
route-map rm-no-export permit 10
 set community additive no-export
route-map rm-no-export permit 20
!
route-map rm-blackhole permit 10
 description blackhole, up-pref and ensure it cannot escape this AS
 set ip next-hop 127.0.0.1
 set local-preference 10
 set community additive no-export
route-map rm-blackhole permit 20
!
! Set local-pref as requested
route-map rm-prefmod permit 10
```

(continues on next page)

(continued from previous page)

```
 match community cm-prefmod-100
 set local-preference 100
route-map rm-prefmod permit 20
 match community cm-prefmod-200
 set local-preference 200
route-map rm-prefmod permit 30
 match community cm-prefmod-300
 set local-preference 300
route-map rm-prefmod permit 40
 match community cm-prefmod-400
 set local-preference 400
route-map rm-prefmod permit 50
!
! Community actions to take on receipt of route.
route-map rm-community-in permit 10
 description check for blackholing, no point continuing if it matches.
 match community cm-blackhole
 call rm-blackhole
route-map rm-community-in permit 20
 match community cm-set-no-export
 call rm-no-export
 on-match next
route-map rm-community-in permit 30
 match community cme-prefmod-range
 call rm-prefmod
route-map rm-community-in permit 40
!
! ###################################################################
! Community actions to take when advertising a route.
! These are filtering route-maps,
!
! Deny customer routes to upstream with cust-only set.
route-map rm-community-filt-to-upstream deny 10
 match community cm-learnt-cust
 match community cm-cust-only
route-map rm-community-filt-to-upstream permit 20
!
! Deny customer routes to other customers with upstream-only set.
route-map rm-community-filt-to-cust deny 10
 match community cm-learnt-cust
 match community cm-upstream-only
route-map rm-community-filt-to-cust permit 20
!
! ###################################################################
! The top-level route-maps applied to sessions. Further entries could
! be added obviously..
!
! Customers
route-map rm-cust-in permit 10
 call rm-community-in
 on-match next
route-map rm-cust-in permit 20
```

(continues on next page)

(continued from previous page)

```
 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
```

(continues on next page)

(continued from previous page)

```
!
log file bgpd.log
!
```

3.3.10 BGP tcp-mss support

TCP provides a mechanism for the user to specify the max segment size. setsockopt API is used to set the max segment size for TCP session. We can configure this as part of BGP neighbor configuration.

This document explains how to avoid ICMP vulnerability issues by limiting TCP max segment size when you are using MTU discovery. Using MTU discovery on TCP paths is one method of avoiding BGP packet fragmentation.

TCP negotiates a maximum segment size (MSS) value during session connection establishment between two peers. The MSS value negotiated is primarily based on the maximum transmission unit (MTU) of the interfaces to which the communicating peers are directly connected. However, due to variations in link MTU on the path taken by the TCP packets, some packets in the network that are well within the MSS value might be fragmented when the packet size exceeds the link's MTU.

This feature is supported with TCP over IPv4 and TCP over IPv6.

CLI Configuration:

Below configuration can be done in router bgp mode and allows the user to configure the tcp-mss value per neighbor. The configuration gets applied only after hard reset is performed on that neighbor. If we configure tcp-mss on both the neighbors then both neighbors need to be reset.

The configuration takes effect based on below rules, so there is a configured tcp-mss and a synced tcp-mss value per TCP session.

By default if the configuration is not done then the TCP max segment size is set to the Maximum Transmission unit (MTU) – (IP/IP6 header size + TCP header size + ethernet header). For IPv4 its MTU – (20 bytes IP header + 20 bytes TCP header + 12 bytes ethernet header) and for IPv6 its MTU – (40 bytes IPv6 header + 20 bytes TCP header + 12 bytes ethernet header).

If the config is done then it reduces 12-14 bytes for the ether header and uses it after synchronizing in TCP handshake.

neighbor <A.B.C.D|X:X::X:X|WORD> tcp-mss (1-65535)

When tcp-mss is configured kernel reduces 12-14 bytes for ethernet header. E.g. if tcp-mss is configured as 150 the synced value will be 138.

Note: configured and synced value is different since TCP module will reduce 12 bytes for ethernet header.

Running config:

```
frr# show running-config
Building configuration...

Current configuration:
!
router bgp 100
 bgp router-id 192.0.2.1
```

(continues on next page)

(continued from previous page)

```
 neighbor 198.51.100.2 remote-as 100
 neighbor 198.51.100.2 tcp-mss 150       => new entry
 neighbor 2001:DB8::2 remote-as 100
 neighbor 2001:DB8::2 tcp-mss 400        => new entry
```

Show command:

```
frr# show bgp neighbors 198.51.100.2
BGP neighbor is 198.51.100.2, remote AS 100, local AS 100, internal link
Hostname: frr
  BGP version 4, remote router ID 192.0.2.2, local router ID 192.0.2.1
  BGP state = Established, up for 02:15:28
  Last read 00:00:28, Last write 00:00:28
  Hold time is 180, keepalive interval is 60 seconds
  Configured tcp-mss is 150, synced tcp-mss is 138     => new display
```

```
frr# show bgp neighbors 2001:DB8::2
BGP neighbor is 2001:DB8::2, remote AS 100, local AS 100, internal link
Hostname: frr
  BGP version 4, remote router ID 192.0.2.2, local router ID 192.0.2.1
  BGP state = Established, up for 02:16:34
  Last read 00:00:34, Last write 00:00:34
  Hold time is 180, keepalive interval is 60 seconds
  Configured tcp-mss is 400, synced tcp-mss is 388      => new display
```

Show command json output:

```
frr# show bgp neighbors 2001:DB8::2 json
{
  "2001:DB8::2":{
    "remoteAs":100,
    "localAs":100,
    "nbrInternalLink":true,
    "hostname":"frr",
    "bgpVersion":4,
    "remoteRouterId":"192.0.2.2",
    "localRouterId":"192.0.2.1",
    "bgpState":"Established",
    "bgpTimerUpMsec":8349000,
    "bgpTimerUpString":"02:19:09",
    "bgpTimerUpEstablishedEpoch":1613054251,
    "bgpTimerLastRead":9000,
    "bgpTimerLastWrite":9000,
    "bgpInUpdateElapsedTimeMsecs":8347000,
    "bgpTimerHoldTimeMsecs":180000,
    "bgpTimerKeepAliveIntervalMsecs":60000,
    "bgpTcpMssConfigured":400,                                  => new entry
    "bgpTcpMssSynced":388,                                   => new entry
```

```
frr# show bgp neighbors 198.51.100.2 json
{
  "198.51.100.2":{
    "remoteAs":100,
    "localAs":100,
    "nbrInternalLink":true,
    "hostname":"frr",
    "bgpVersion":4,
    "remoteRouterId":"192.0.2.2",
    "localRouterId":"192.0.2.1",
    "bgpState":"Established",
    "bgpTimerUpMsec":8370000,
    "bgpTimerUpString":"02:19:30",
    "bgpTimerUpEstablishedEpoch":1613054251,
    "bgpTimerLastRead":30000,
    "bgpTimerLastWrite":30000,
    "bgpInUpdateElapsedTimeMsecs":8368000,
    "bgpTimerHoldTimeMsecs":180000,
    "bgpTimerKeepAliveIntervalMsecs":60000,
    "bgpTcpMssConfigured":150,                                    => new entry
    "bgpTcpMssSynced":138,                                      => new entry
```

3.3.11 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.

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.

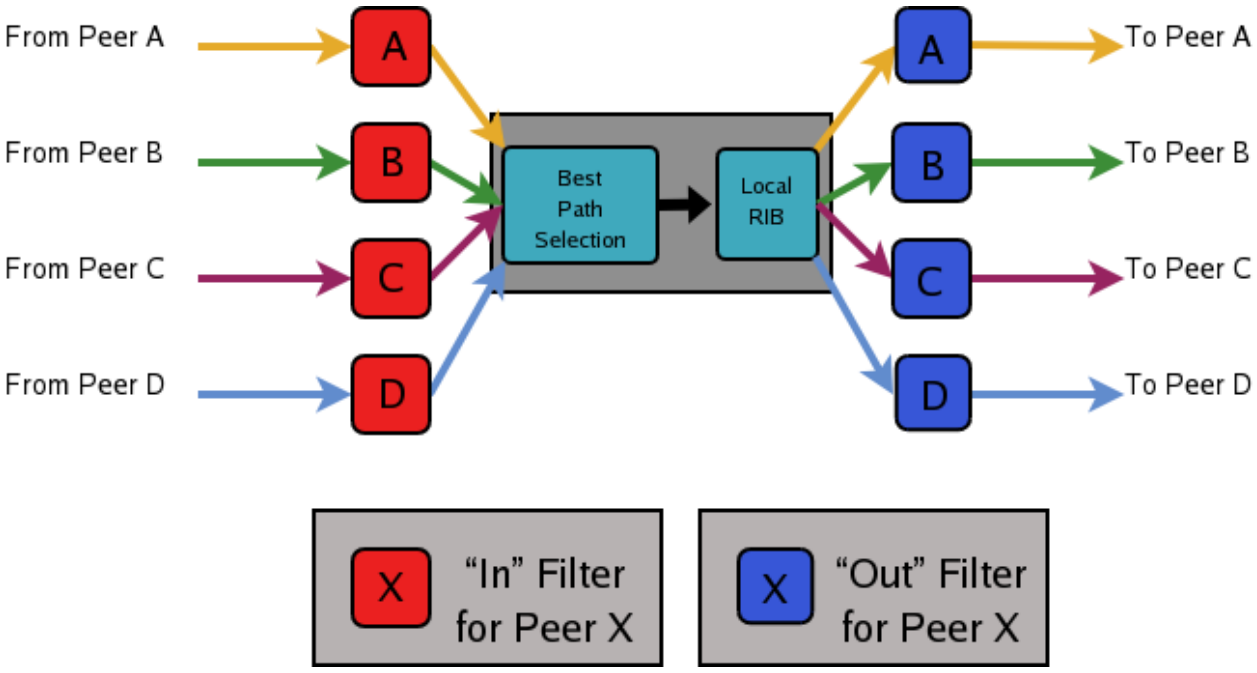


Fig. 1: Announcement processing inside a 'normal' BGP speaker

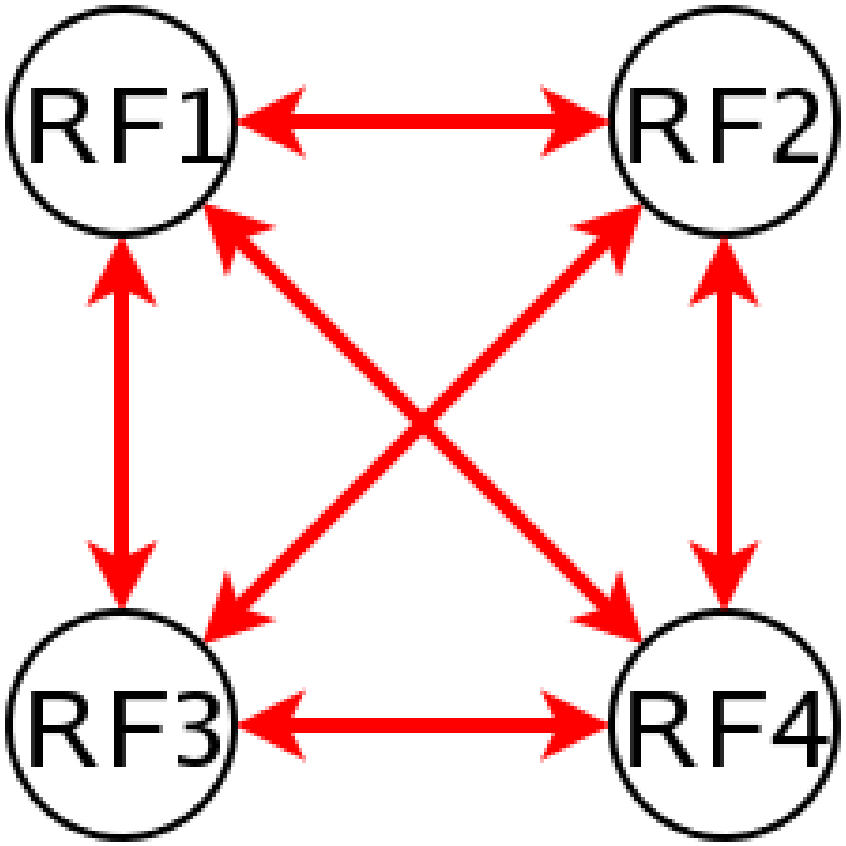


Fig. 2: Full Mesh

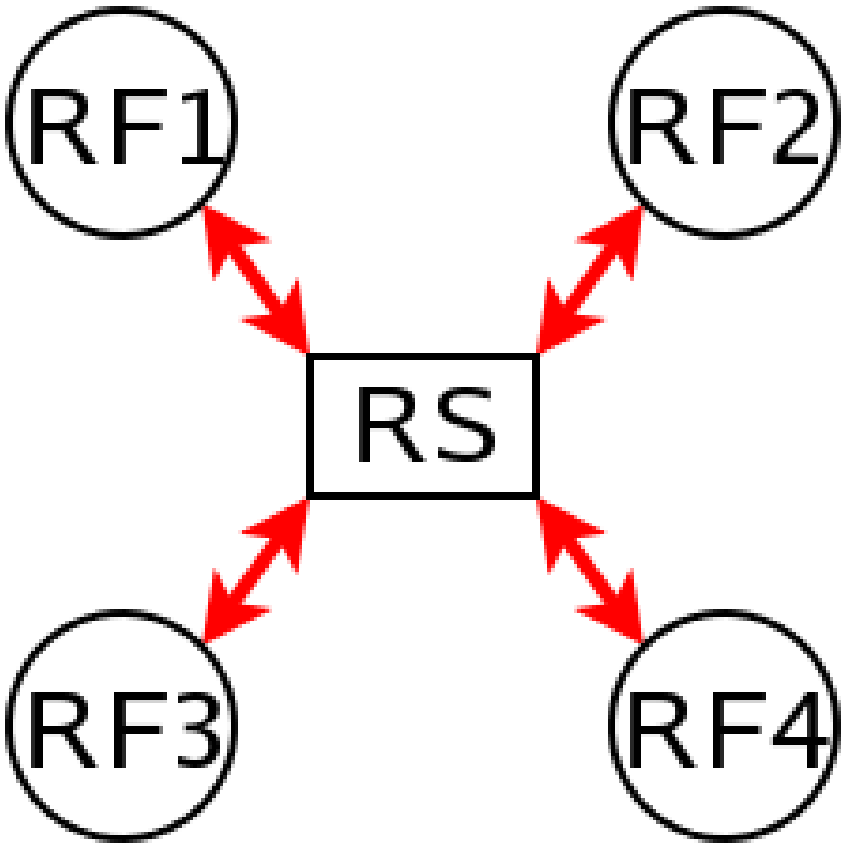


Fig. 3: Route server and clients

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.

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).

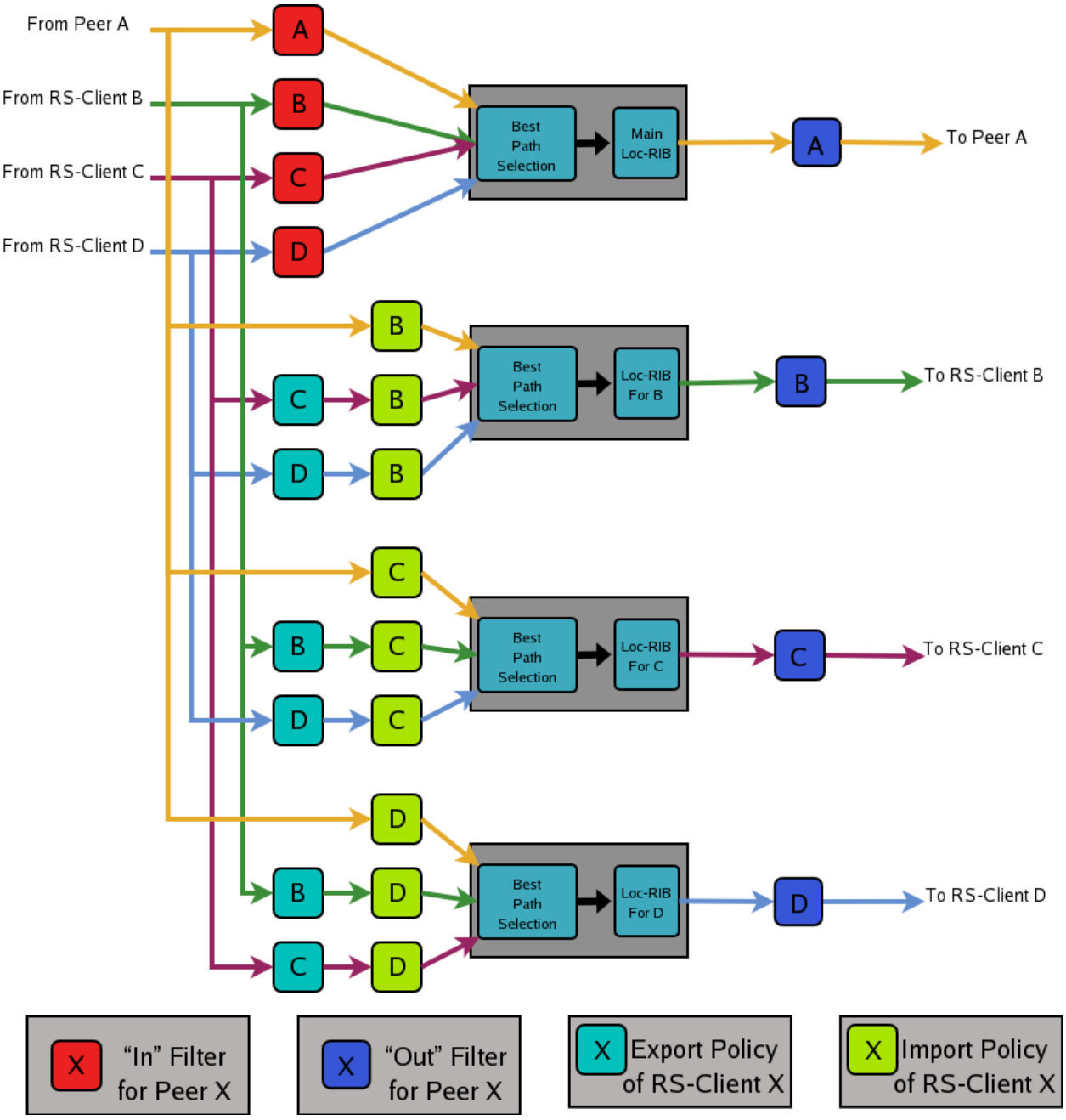


Fig. 4: Announcement processing model implemented by the Route Server

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 in|out
This set of commands can be used to specify the route-map that represents the Import or Export policy of a peer which is configured as a RS-client (with the previous command).

match peer A.B.C.D|X:X::X:X
This is a new *match* statement for use in route-maps, enabling them to describe import/export policies. As we said before, an import/export policy represents a set of input/output filters of the RS-client. This statement makes possible that a single route-map represents the full set of filters that a BGP speaker would use for its different peers in a non-RS scenario.

The *match peer* statement has different semantics whether it is used inside an import or an export route-map. In the first case the statement matches if the address of the peer who sends the announce is the same that the address specified by {A.B.C.D|X:X::X:X}. For export route-maps it matches when {A.B.C.D|X:X::X:X} is the address of the RS-Client into whose Loc-RIB the announce is going to be inserted (how the same export policy is applied before different Loc-RIBs is shown in *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
!
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
```

(continues on next page)

(continued from previous page)

```
  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
!
line vty
!
```

Which is logically much simpler than its initial configuration, as it now maintains only one BGP peering and all the filters (route-maps) have disappeared.

Configuration of the Route Server itself

As we said when we described the functions of a route server (*Description of the Route Server model*), it is in charge of all the route filtering. To achieve that, the In and Out filters from the RA, RB and RC configurations must be converted into Import and Export policies in the route server.

This is a fragment of the route server configuration (we only show the policies for client RA):

```
# Configuration for Route Server ('RS')
!
hostname RS
password ix
!
router bgp 65000 view RS
  no bgp default ipv4-unicast
  neighbor 2001:0DB8::A  remote-as 65001
  neighbor 2001:0DB8::B  remote-as 65002
  neighbor 2001:0DB8::C  remote-as 65003
!
  address-family ipv6
    neighbor 2001:0DB8::A activate
    neighbor 2001:0DB8::A route-server-client
    neighbor 2001:0DB8::A route-map RSCLIENT-A-IMPORT in
    neighbor 2001:0DB8::A route-map RSCLIENT-A-EXPORT out
    neighbor 2001:0DB8::A soft-reconfiguration inbound

    neighbor 2001:0DB8::B activate
    neighbor 2001:0DB8::B route-server-client
    neighbor 2001:0DB8::B route-map RSCLIENT-B-IMPORT in
    neighbor 2001:0DB8::B route-map RSCLIENT-B-EXPORT out
    neighbor 2001:0DB8::B soft-reconfiguration inbound

    neighbor 2001:0DB8::C activate
    neighbor 2001:0DB8::C route-server-client
    neighbor 2001:0DB8::C route-map RSCLIENT-C-IMPORT in
    neighbor 2001:0DB8::C route-map RSCLIENT-C-EXPORT out
    neighbor 2001:0DB8::C soft-reconfiguration inbound
  exit-address-family
!
ipv6 prefix-list COMMON-PREFIXES seq  5 permit 2001:0DB8:0000::/48 ge 64 le 64
ipv6 prefix-list COMMON-PREFIXES seq 10 deny any
!
ipv6 prefix-list PEER-A-PREFIXES seq  5 permit 2001:0DB8:AAAA::/48 ge 64 le 64
ipv6 prefix-list PEER-A-PREFIXES seq 10 deny any
!
ipv6 prefix-list PEER-B-PREFIXES seq  5 permit 2001:0DB8:BBBB::/48 ge 64 le 64
ipv6 prefix-list PEER-B-PREFIXES seq 10 deny any
!
ipv6 prefix-list PEER-C-PREFIXES seq  5 permit 2001:0DB8:CCCC::/48 ge 64 le 64
ipv6 prefix-list PEER-C-PREFIXES seq 10 deny any
!
route-map RSCLIENT-A-IMPORT permit 10
  match peer 2001:0DB8::B
  call A-IMPORT-FROM-B
```

(continues on next page)

(continued from previous page)

```
route-map RSCLIENT-A-IMPORT permit 20
  match peer 2001:0DB8::C
  call A-IMPORT-FROM-C
!
route-map A-IMPORT-FROM-B permit 10
  match ipv6 address prefix-list COMMON-PREFIXES
  set metric 100
route-map A-IMPORT-FROM-B permit 20
  match ipv6 address prefix-list PEER-B-PREFIXES
  set community 65001:11111
!
route-map A-IMPORT-FROM-C permit 10
  match ipv6 address prefix-list COMMON-PREFIXES
  set metric 200
route-map A-IMPORT-FROM-C permit 20
  match ipv6 address prefix-list PEER-C-PREFIXES
  set community 65001:22222
!
route-map RSCLIENT-A-EXPORT permit 10
  match peer 2001:0DB8::B
  match ipv6 address prefix-list PEER-A-PREFIXES
route-map RSCLIENT-A-EXPORT permit 20
  match peer 2001:0DB8::C
  match ipv6 address prefix-list PEER-A-PREFIXES
!
...
...
...
```

If you compare the initial configuration of RA with the route server configuration above, you can see how easy it is to generate the Import and Export policies for RA from the In and Out route-maps of RA's original configuration.

When there was no route server, RA maintained two peerings, one with RB and another with RC. Each of this peerings had an In route-map configured. To build the Import route-map for client RA in the route server, simply add route-map entries following this scheme:

```
route-map <NAME> permit 10
    match peer <Peer Address>
    call <In Route-Map for this Peer>
route-map <NAME> permit 20
    match peer <Another Peer Address>
    call <In Route-Map for this Peer>
```

This is exactly the process that has been followed to generate the route-map RSCLIENT-A-IMPORT. The route-maps that are called inside it (A-IMPORT-FROM-B and A-IMPORT-FROM-C) are exactly the same than the In route-maps from the original configuration of RA (PEER-B-IN and PEER-C-IN), only the name is different.

The same could have been done to create the Export policy for RA (route-map RSCLIENT-A-EXPORT), but in this case the original Out route-maps where so simple that we decided not to use the *call WORD* commands, and we integrated all in a single route-map (RSCLIENT-A-EXPORT).

The Import and Export policies for RB and RC are not shown, but the process would be identical.

Further considerations about Import and Export route-maps

The current version of the route server patch only allows to specify a route-map for import and export policies, while in a standard BGP speaker apart from route-maps there are other tools for performing input and output filtering (access-lists, community-lists, ...). But this does not represent any limitation, as all kinds of filters can be included in import/export route-maps. For example suppose that in the non-route-server scenario peer RA had the following filters configured for input from peer B:

```
neighbor 2001:0DB8::B prefix-list LIST-1 in
neighbor 2001:0DB8::B filter-list LIST-2 in
neighbor 2001:0DB8::B route-map PEER-B-IN in
...
...
route-map PEER-B-IN permit 10
  match ipv6 address prefix-list COMMON-PREFIXES
  set local-preference 100
route-map PEER-B-IN permit 20
  match ipv6 address prefix-list PEER-B-PREFIXES
  set community 65001:11111
```

It is possible to write a single route-map which is equivalent to the three filters (the community-list, the prefix-list and the route-map). That route-map can then be used inside the Import policy in the route server. Lets see how to do it:

```
neighbor 2001:0DB8::A route-map RSCLIENT-A-IMPORT in
...
!
...
route-map RSCLIENT-A-IMPORT permit 10
  match peer 2001:0DB8::B
  call A-IMPORT-FROM-B
...
...
!
route-map A-IMPORT-FROM-B permit 1
  match ipv6 address prefix-list LIST-1
  match as-path LIST-2
  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.

3.3.12 Prefix Origin Validation Using RPKI

Prefix Origin Validation allows BGP routers to verify if the origin AS of an IP prefix is legitimate to announce this IP prefix. The required attestation objects are stored in the Resource Public Key Infrastructure (RPKI). However, RPKI-enabled routers do not store cryptographic data itself but only validation information. The validation of the cryptographic data (so called Route Origin Authorization, or short ROA, objects) will be performed by trusted cache servers. The RPKI/RTR protocol defines a standard mechanism to maintain the exchange of the prefix/origin AS mapping between the cache server and routers. In combination with a BGP Prefix Origin Validation scheme a router is able to verify received BGP updates without suffering from cryptographic complexity.

The RPKI/RTR protocol is defined in **RFC 6810** and the validation scheme in **RFC 6811**. The current version of Prefix Origin Validation in FRR implements both RFCs.

For a more detailed but still easy-to-read background, we suggest:

- [Securing-BGP]
- [Resource-Certification]

Features of the Current Implementation

In a nutshell, the current implementation provides the following features

- The BGP router can connect to one or more RPKI cache servers to receive validated prefix to origin AS mappings. Advanced failover can be implemented by server sockets with different preference values.
- If no connection to an RPKI cache server can be established after a pre-defined timeout, the router will process routes without prefix origin validation. It still will try to establish a connection to an RPKI cache server in the background.
- By default, enabling RPKI does not change best path selection. In particular, invalid prefixes will still be considered during best path selection. However, the router can be configured to ignore all invalid prefixes.
- Route maps can be configured to match a specific RPKI validation state. This allows the creation of local policies, which handle BGP routes based on the outcome of the Prefix Origin Validation.
- Updates from the RPKI cache servers are directly applied and path selection is updated accordingly. (Soft reconfiguration **must** be enabled for this to work).

Enabling RPKI

You must install `frr-rpki-rtrlib` additional package for RPKI support, otherwise `bgpd` daemon won't startup.

`rpki`
: This command enables the RPKI configuration mode. Most commands that start with *rpki* can only be used in this mode.

  When it is used in a telnet session, leaving of this mode cause rpki to be initialized.

  Executing this command alone does not activate prefix validation. You need to configure at least one reachable cache server. See section *Configuring RPKI/RTR Cache Servers* for configuring a cache server.

Remember to add `-M rpki` to the variable `bgpd_options` in `/etc/frr/daemons` , like so:

```
bgpd_options="   -A 127.0.0.1 -M rpki"
```

instead of the default setting:

```
bgpd_options="   -A 127.0.0.1"
```

Otherwise you will encounter an error when trying to enter RPKI configuration mode due to the `rpki` module not being loaded when the BGP daemon is initialized.

Examples of the error:

```
router(config)# debug rpki
% [BGP] Unknown command: debug rpki

router(config)# rpki
% [BGP] Unknown command: rpki
```

Note that the RPKI commands will be available in vtysh when running `find rpki` regardless of whether the module is loaded.

Configuring RPKI/RTR Cache Servers

The following commands are independent of a specific cache server.

rpki polling_period (1-3600)
Set the number of seconds the router waits until the router asks the cache again for updated data.

The default value is 300 seconds.

rpki expire_interval (600-172800)
Set the number of seconds the router waits until the router expires the cache.

The default value is 7200 seconds.

rpki retry_interval (1-7200)
Set the number of seconds the router waits until retrying to connect to the cache server.

The default value is 600 seconds.

**rpki cache (A.B.C.
D|WORD) PORT [SSH_USERNAME] [SSH_PRIVKEY_PATH] [KNOWN_HOSTS_PATH] [source A.B.C.
D] preference (1-255)**
Add a cache server to the socket. By default, the connection between router and cache server is based on plain TCP. Protecting the connection between router and cache server by SSH is optional. Deleting a socket removes the associated cache server and terminates the existing connection.

A.B.C.D|WORD Address of the cache server.

PORT Port number to connect to the cache server

SSH_USERNAME SSH username to establish an SSH connection to the cache server.

SSH_PRIVKEY_PATH Local path that includes the private key file of the router.

KNOWN_HOSTS_PATH Local path that includes the known hosts file. The default value depends on the configuration of the operating system environment, usually `~/.ssh/known_hosts`.

source A.B.C.D Source address of the RPKI connection to access cache server.

Validating BGP Updates

match rpki notfound|invalid|valid
Create a clause for a route map to match prefixes with the specified RPKI state.

In the following example, the router prefers valid routes over invalid prefixes because invalid routes have a lower local preference.

```
! Allow for invalid routes in route selection process
route bgp 60001
!
! Set local preference of invalid prefixes to 10
route-map rpki permit 10
 match rpki invalid
 set local-preference 10
!
! Set local preference of valid prefixes to 500
route-map rpki permit 500
 match rpki valid
 set local-preference 500
```

match rpki-extcommunity notfound|invalid|valid
Create a clause for a route map to match prefixes with the specified RPKI state, that is derived from the Origin Validation State extended community attribute (OVS). OVS extended community is non-transitive and is exchanged only between iBGP peers.

Debugging

debug rpki
Enable or disable debugging output for RPKI.

Displaying RPKI

show rpki prefix <A.B.C.D/M|X:X::X:X/M> [(1-4294967295)] [json]
Display validated prefixes received from the cache servers filtered by the specified prefix.

show rpki as-number ASN [json]
Display validated prefixes received from the cache servers filtered by ASN.

show rpki prefix-table [json]
Display all validated prefix to origin AS mappings/records which have been received from the cache servers and stored in the router. Based on this data, the router validates BGP Updates.

show rpki cache-server [json]
Display all configured cache servers, whether active or not.

show rpki cache-connection [json]
Display all cache connections, and show which is connected or not.

show bgp [afi] [safi] <A.B.C.D|A.B.C.D/M|X:X::X:X|X:X::X:X/M> rpki <valid|invalid|notfound>
Display for the specified prefix or address the bgp paths that match the given rpki state.

show bgp [afi] [safi] rpki <valid|invalid|notfound>
Display all prefixes that match the given rpki state.

RPKI Configuration Example

```
hostname bgpd1
password zebra
! log stdout
debug bgp updates
debug bgp keepalives
debug rpki
!
rpki
 rpki polling_period 1000
 rpki timeout 10
  ! SSH Example:
  rpki cache example.com source 141.22.28.223 22 rtr-ssh ./ssh_key/id_rsa ./ssh_key/id_
↪rsa.pub preference 1
  ! TCP Example:
  rpki cache rpki-validator.realmv6.org 8282 preference 2
  exit
!
router bgp 60001
 bgp router-id 141.22.28.223
 network 192.168.0.0/16
 neighbor 123.123.123.0 remote-as 60002
 neighbor 123.123.123.0 route-map rpki in
 neighbor 123.123.123.0 update-source 141.22.28.223
!
 address-family ipv6
  neighbor 123.123.123.0 activate
  neighbor 123.123.123.0 route-map rpki in
 exit-address-family
!
route-map rpki permit 10
 match rpki invalid
 set local-preference 10
!
route-map rpki permit 20
 match rpki notfound
 set local-preference 20
!
route-map rpki permit 30
 match rpki valid
 set local-preference 30
!
route-map rpki permit 40
!
```

3.3.13 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 [Draft-IETF-idr-link-bandwidth]. 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 [Draft-IETF-mohanty-bess-ebgp-dmz] 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 [Draft-IETF-idr-link-bandwidth], the behavior reverts to standard ECMP among all the multipaths, with the link bandwidth being effectively ignored.

Note that there is no change to either the BGP best path selection algorithm or to the multipath computation algorithm; the mapping of link bandwidth to weight happens at the time of installation of the route in the RIB.

If data forwarding is implemented by means of the Linux kernel, the next hop's weight is used in the hash calculation. The kernel uses the Hash threshold algorithm and use of the next hop weight is built into it; next hops need not be expanded to achieve UCMP. UCMP for IPv4 is available in older Linux kernels too, while UCMP for IPv6 is available from the 4.16 kernel onwards.

If data forwarding is realized in hardware, common implementations expand the next hops (i.e., they are repeated) in the ECMP container in proportion to their weight. For example, if the weights associated with 3 next hops for a particular route are 50, 25 and 25 and the ECMP container has a size of 16 next hops, the first next hop will be repeated 8 times and the other 2 next hops repeated 4 times each. Other implementations are also possible.

Unequal cost multipath across a network

For the use cases listed above, it is not sufficient to support UCMP on just one router (e.g., egress router), or individually, on multiple routers; UCMP must be deployed across the entire network. This is achieved by employing the BGP link-bandwidth extended community.

At the router which originates the BGP link bandwidth, there has to be user configuration to trigger it, which is described below. Receiving routers would use the received link bandwidth from their downstream routers to determine the next hop weight as described in the earlier section. Further, if the received link bandwidth is a transitive attribute, it would be propagated to eBGP peers, with the additional change that if the next hop is set to oneself, the cumulative link bandwidth of all downstream paths is propagated to other routers. In this manner, the entire network will know how to distribute traffic to an anycast service across the network.

The BGP link-bandwidth extended community is encoded in bytes-per-second. In the use case where UCMP must be based on the number of paths, a reference bandwidth of 1 Mbps is used. So, for example, if there are 4 equal cost paths to an anycast IP, the encoded bandwidth in the extended community will be 500,000. The actual value itself doesn't matter as long as all routers originating the link-bandwidth are doing it in the same way.

Configuration Guide

The configuration for weighted ECMP using BGP link bandwidth requires one essential step - using a route-map to inject the link bandwidth extended community. An additional option is provided to control the processing of received link bandwidth.

Injecting link bandwidth into the network

At the "entry point" router that is injecting the prefix to which weighted load balancing must be performed, a route-map must be configured to attach the link bandwidth extended community.

For the use case of providing weighted load balancing for an anycast service, this configuration will typically need to be applied at the TOR or Leaf router that is connected to servers which provide the anycast service and the bandwidth would be based on the number of multipaths for the destination.

For the use case of load balancing to the exit router, the exit router should be configured with the route map specifying the a bandwidth value that corresponds to the bandwidth of the link connecting to its eBGP peer in the adjoining AS. In addition, the link bandwidth extended community must be explicitly configured to be non-transitive.

The complete syntax of the route-map set command can be found at *BGP Extended Communities in Route Map*

This route-map is supported only at two attachment points: (a) the outbound route-map attached to a peer or peer-group, per address-family (b) the EVPN advertise route-map used to inject IPv4 or IPv6 unicast routes into EVPN as type-5 routes.

Since the link bandwidth origination is done by using a route-map, it can be constrained to certain prefixes (e.g., only for anycast services) or it can be generated for all prefixes. Further, when the route-map is used in the neighbor context, the link bandwidth usage can be constrained to certain peers only.

A sample configuration is shown below and illustrates link bandwidth advertisement towards the "SPINE" peer-group for anycast IPs in the range 192.168.x.x

```
ip prefix-list anycast_ip seq 10 permit 192.168.0.0/16 le 32
route-map anycast_ip permit 10
 match ip address prefix-list anycast_ip
 set extcommunity bandwidth num-multipaths
route-map anycast_ip permit 20
```

(continues on next page)

(continued from previous page)

```
!
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 [Draft-IETF-idr-link-bandwidth].

The operator can change these behaviors with the following configuration:

bgp bestpath bandwidth <ignore | skip-missing | default-weight-for-missing>

The different options imply behavior as follows:

- ignore: Ignore link bandwidth completely for route installation (i.e., do regular ECMP, not weighted)
- skip-missing: Skip paths without link bandwidth and do UCMP among the others (if at least some paths have link-bandwidth)
- default-weight-for-missing: Assign a low default weight (value 1) to paths not having link bandwidth

This configuration is per BGP instance similar to other BGP route-selection controls; it operates on both IPv4-unicast and IPv6-unicast routes in that instance. In an EVPN network, this configuration (if required) should be implemented in the tenant VRF and is again applicable for IPv4-unicast and IPv6-unicast, including the ones sourced from EVPN type-5 routes.

A sample snippet of FRR configuration on a receiver to skip paths without link bandwidth and do weighted ECMP among the other paths (if some of them have link bandwidth) is as shown below.

```
router bgp 65021
 bgp bestpath as-path multipath-relax
 bgp bestpath bandwidth skip-missing
 neighbor LEAF peer-group
 neighbor LEAF remote-as external
 neighbor 172.16.35.2 peer-group LEAF
 neighbor 172.16.36.2 peer-group LEAF
 !
 address-family ipv4 unicast
  network 130.0.0.1/32
 exit-address-family
!
```

Stopping the propagation of the link bandwidth outside a domain

The link bandwidth extended community will get automatically propagated with the prefix to EBGP peers, if it is encoded as a transitive attribute by the originator. If this propagation has to be stopped outside of a particular domain (e.g., stopped from being propagated to routers outside of the data center core network), the mechanism available is to disable the advertisement of all BGP extended communities on the specific peering/s. In other words, the propagation cannot be blocked just for the link bandwidth extended community. The configuration to disable all extended communities can be applied to a peer or peer-group (per address-family).

Of course, the other common way to stop the propagation of the link bandwidth outside the domain is to block the prefixes themselves from being advertised and possibly, announce only an aggregate route. This would be quite common in a EVPN network.

BGP link bandwidth and UCMP monitoring & troubleshooting

Existing operational commands to display the BGP routing table for a specific prefix will show the link bandwidth extended community also, if present.

An example of an IPv4-unicast route received with the link bandwidth attribute from two peers is shown below:

```
CLI# show bgp ipv4 unicast 192.168.10.1/32
BGP routing table entry for 192.168.10.1/32
Paths: (2 available, best #2, table default)
  Advertised to non peer-group peers:
  l1(swp1) l2(swp2) l3(swp3) l4(swp4)
  65002
    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

3.3.14 Flowspec

Overview

Flowspec introduces a new NLRI (Network Layer Reachability Information) encoding format that is used to distribute traffic rule flow specifications. Basically, instead of simply relying on destination IP address for IP prefixes, the IP prefix is replaced by a n-tuple consisting of a rule. That rule can be a more or less complex combination of the following:

- Network source/destination (can be one or the other, or both).
- Layer 4 information for UDP/TCP: source port, destination port, or any port.
- Layer 4 information for ICMP type and ICMP code.
- Layer 4 information for TCP Flags.
- Layer 3 information: DSCP value, Protocol type, packet length, fragmentation.
- Misc layer 4 TCP flags.

Note that if originally Flowspec defined IPv4 rules, this is also possible to use IPv6 address-family. The same set of combinations as defined for IPv4 can be used.

A combination of the above rules is applied for traffic filtering. This is encoded as part of specific BGP extended communities and the action can range from the obvious rerouting (to nexthop or to separate VRF) to shaping, or discard.

The following IETF drafts and RFCs have been used to implement FRR Flowspec:

- **RFC 5575**
- [Draft-IETF-IDR-Flowspec-redirect-IP]
- [Draft-IETF-IDR-Flow-Spec-V6]

Design Principles

FRR implements the Flowspec client side, that is to say that BGP is able to receive Flowspec entries, but is not able to act as manager and send Flowspec entries.

Linux provides the following mechanisms to implement policy based routing:

- Filtering the traffic with `Netfilter`. `Netfilter` provides a set of tools like `ipset` and `iptables` that are powerful enough to be able to filter such Flowspec filter rule.
- using non standard routing tables via `iproute2` (via the `ip rule` command provided by `iproute2`). `iproute2` is already used by FRR's *PBR* daemon which provides basic policy based routing based on IP source and destination criterion.

Below example is an illustration of what Flowspec will inject in the underlying system:

```
# linux shell
ipset create match0x102 hash:net,net counters
ipset add match0x102 32.0.0.0/16,40.0.0.0/16
iptables -N match0x102 -t mangle
iptables -A match0x102 -t mangle -j MARK --set-mark 102
iptables -A match0x102 -t mangle -j ACCEPT
iptables -i ntfp3 -t mangle -I PREROUTING -m set --match-set match0x102
             src,dst -g match0x102
ip rule add fwmark 102 lookup 102
ip route add 40.0.0.0/16 via 44.0.0.2 table 102
```

For handling an incoming Flowspec entry, the following workflow is applied:

- Incoming Flowspec entries are handled by *bgpd*, stored in the BGP RIB.
- Flowspec entry is installed according to its complexity.

It will be installed if one of the following filtering action is seen on the BGP extended community: either redirect IP, or redirect VRF, in conjunction with rate option, for redirecting traffic. Or rate option set to 0, for discarding traffic.

According to the degree of complexity of the Flowspec entry, it will be installed in *zebra* RIB. For more information about what is supported in the FRR implementation as rule, see *Limitations / Known Issues* chapter. Flowspec entry is split in several parts before being sent to *zebra*.

- *zebra* daemon receives the policy routing configuration

Policy Based Routing entities necessary to policy route the traffic in the underlying system, are received by *zebra*. Two filtering contexts will be created or appended in `Netfilter`: `ipset` and `iptable` context. The former is used to define an IP filter based on multiple criterium. For instance, an ipset `net:net` is based on two ip addresses, while `net,port,net` is based on two ip addresses and one port (for ICMP, UDP, or TCP). The way the filtering is used (for example, is src port or dst port used?) is defined by the latter filtering context. `iptable` command will reference the `ipset` context and will tell how to filter and what to do. In our case, a marker will be set to indicate `iproute2` where to forward the traffic to. Sometimes, for dropping action, there is no need to add a marker; the `iptable` will tell to drop all packets matching the `ipset` entry.

Configuration Guide

In order to configure an IPv4 Flowspec engine, use the following configuration. As of today, it is only possible to configure Flowspec on the default VRF.

```
router bgp <AS>
  neighbor <A.B.C.D> remote-as <remoteAS>
  neighbor <A:B::C:D> remote-as <remoteAS2>
  address-family ipv4 flowspec
   neighbor <A.B.C.D> activate
  exit
  address-family ipv6 flowspec
   neighbor <A:B::C:D> activate
  exit
exit
```

You can see Flowspec entries, by using one of the following show commands:

show bgp ipv4 flowspec [detail | A.B.C.D]

show bgp ipv6 flowspec [detail | A:B::C:D]

Per-interface configuration

One nice feature to use is the ability to apply Flowspec to a specific interface, instead of applying it to the whole machine. Despite the following IETF draft [Draft-IETF-IDR-Flowspec-Interface-Set] is not implemented, it is possible to manually limit Flowspec application to some incoming interfaces. Actually, not using it can result to some unexpected behaviour like accounting twice the traffic, or slow down the traffic (filtering costs). To limit Flowspec to one specific interface, use the following command, under *flowspec address-family* node.

local-install <IFNAME | any>

By default, Flowspec is activated on all interfaces. Installing it to a named interface will result in allowing only this interface. Conversely, enabling any interface will flush all previously configured interfaces.

VRF redirection

Another nice feature to configure is the ability to redirect traffic to a separate VRF. This feature does not go against the ability to configure Flowspec only on default VRF. Actually, when you receive incoming BGP flowspec entries on that default VRF, you can redirect traffic to an other VRF.

As a reminder, BGP flowspec entries have a BGP extended community that contains a Route Target. Finding out a local VRF based on Route Target consists in the following:

- A configuration of each VRF must be done, with its Route Target set Each VRF is being configured within a BGP VRF instance with its own Route Target list. Route Target accepted format matches the following: `A.B.C.D:U16`, or `U16:U32`, `U32:U16`.
- The first VRF with the matching Route Target will be selected to route traffic to. Use the following command under ipv4 unicast address-family node

rt redirect import RTLIST...

In order to illustrate, if the Route Target configured in the Flowspec entry is `E.F.G.H:II`, then a BGP VRF instance with the same Route Target will be set set. That VRF will then be selected. The below full configuration example depicts how Route Targets are configured and how VRFs and cross VRF configuration is done. Note that the VRF

are mapped on Linux Network Namespaces. For data traffic to cross VRF boundaries, virtual ethernet interfaces are created with private IP addressing scheme.

```
router bgp <ASx>
 neighbor <A.B.C.D> remote-as <ASz>
 address-family ipv4 flowspec
  neighbor A.B.C.D activate
 exit
exit
router bgp <ASy> vrf vrf2
 address-family ipv4 unicast
  rt redirect import <E.F.G.H:II>
 exit
exit
```

Similarly, it is possible to do the same for IPv6 flowspec rules, by using an IPv6 extended community. The format is defined on **RFC 5701**, and that community contains an IPv6 address encoded in the attribute, and matches the locally configured imported route target IPv6 defined under the appropriate BGP VRF instance. Below example defines an IPv6 extended community containing *E:F::G:H* address followed by 2 bytes chosen by admin (here *JJ*).

```
router bgp <ASx>
 neighbor <A:B::C:D> remote-as <ASz>
 address-family ipv6 flowspec
  neighbor A:B::C:D activate
 exit
exit
router bgp <ASy> vrf vrf2
 address-family ipv6 unicast
  rt6 redirect import <E:F::G:H:JJ>
 exit
exit
```

Flowspec monitoring & troubleshooting

You can monitor policy-routing objects by using one of the following commands. Those command rely on the filtering contexts configured from BGP, and get the statistics information retrieved from the underlying system. In other words, those statistics are retrieved from `Netfilter`.

show pbr ipset IPSETNAME | iptable

`IPSETNAME` is the policy routing object name created by `ipset`. About rule contexts, it is possible to know which rule has been configured to policy-route some specific traffic. The `show pbr iptable` command displays for forwarded traffic, which table is used. Then it is easy to use that table identifier to dump the routing table that the forwarded traffic will match.

```
```

show ip route table TABLEID
: `TABLEID` is the table number identifier referencing the non standard routing table used in this example.

debug bgp flowspec
: You can troubleshoot Flowspec, or BGP policy based routing. For instance, if you encounter some issues when decoding a Flowspec entry, you should enable *debug bgp flowspec*.

debug bgp pbr [error]

If you fail to apply the flowspec entry into *zebra*, there should be some relationship with policy routing mechanism. Here, `debug bgp pbr error` could help.

To get information about policy routing contexts created/removed, only use `debug bgp pbr` command.

Ensuring that a Flowspec entry has been correctly installed and that incoming traffic is policy-routed correctly can be checked as demonstrated below. First of all, you must check whether the Flowspec entry has been installed or not.

```
CLI# show bgp ipv4 flowspec 5.5.5.2/32
 BGP flowspec entry: (flags 0x418)
   Destination Address 5.5.5.2/32
   IP Protocol = 17
   Destination Port >= 50 , <= 90
   FS:redirect VRF RT:255.255.255.255:255
   received for 18:41:37
   installed in PBR (match0x271ce00)
```

This means that the Flowspec entry has been installed in an `iptable` named `match0x271ce00`. Once you have confirmation it is installed, you can check whether you find the associate entry by executing following command. You can also check whether incoming traffic has been matched by looking at counter line.

```
CLI# show pbr ipset match0x271ce00
IPset match0x271ce00 type net,port
     to 5.5.5.0/24:proto 6:80-120 (8)
        pkts 1000, bytes 1000000
     to 5.5.5.2:proto 17:50-90 (5)
        pkts 1692918, bytes 157441374
```

As you can see, the entry is present. note that an `iptable` entry can be used to host several Flowspec entries. In order to know where the matching traffic is redirected to, you have to look at the policy routing rules. The policy-routing is done by forwarding traffic to a routing table number. That routing table number is reached by using a `iptable`. The relationship between the routing table number and the incoming traffic is a `MARKER` that is set by the IPtable referencing the IPSet. In Flowspec case, `iptable` referencing the `ipset` context have the same name. So it is easy to know which routing table is used by issuing following command:

```
CLI# show pbr iptable
   IPtable match0x271ce00 action redirect (5)
     pkts 1700000, bytes 158000000
     table 257, fwmark 257
...
```

As you can see, by using following Linux commands, the MARKER `0x101` is present in both `iptable` and `ip rule` contexts.

```
# iptables -t mangle --list match0x271ce00 -v
Chain match0x271ce00 (1 references)
pkts bytes target     prot opt in     out     source               destination
1700K  158M MARK       all  --  any    any     anywhere             anywhere
     MARK set 0x101
1700K  158M ACCEPT     all  --  any    any     anywhere             anywhere

# ip rule list
0:from all lookup local
0:from all fwmark 0x101 lookup 257
```

(continues on next page)

(continued from previous page)

```
32766:from all lookup main
32767:from all lookup default
```

This allows us to see where the traffic is forwarded to.

Limitations / Known Issues

As you can see, Flowspec is rich and can be very complex. As of today, not all Flowspec rules will be able to be converted into Policy Based Routing actions.

- The `Netfilter` driver is not integrated into FRR yet. Not having this piece of code prevents from injecting flowspec entries into the underlying system.
- There are some limitations around filtering contexts

  If I take example of UDP ports, or TCP ports in Flowspec, the information can be a range of ports, or a unique value. This case is handled. However, complexity can be increased, if the flow is a combination of a list of range of ports and an enumerate of unique values. Here this case is not handled. Similarly, it is not possible to create a filter for both src port and dst port. For instance, filter on src port from [1-1000] and dst port = 80. The same kind of complexity is not possible for packet length, ICMP type, ICMP code.

There are some other known issues:

- The validation procedure depicted in **RFC 5575** is not available.

  This validation procedure has not been implemented, as this feature was not used in the existing setups you shared with us.
- The filtering action shaper value, if positive, is not used to apply shaping.

  If value is positive, the traffic is redirected to the wished destination, without any other action configured by Flowspec. It is recommended to configure Quality of Service if needed, more globally on a per interface basis.
- Upon an unexpected crash or other event, *zebra* may not have time to flush PBR contexts.

  That is to say `ipset`, `iptable` and `ip rule` contexts. This is also a consequence due to the fact that ip rule / ipset / iptables are not discovered at startup (not able to read appropriate contexts coming from Flowspec).

Appendix

More information with a public presentation that explains the design of Flowspec inside FRRouting.

[Presentation]

3.3.15 BGP fast-convergence support

Whenever BGP peer address becomes unreachable we must bring down the BGP session immediately. Currently only single-hop EBGP sessions are brought down immediately.IBGP and multi-hop EBGP sessions wait for hold-timer expiry to bring down the sessions.

This new configuration option helps user to teardown BGP sessions immediately whenever peer becomes unreachable.

`bgp fast-convergence`

This configuration is available at the bgp level. When enabled, configuration is applied to all the neighbors configured in that bgp instance.

3.7.1 Starting and Stopping eigrpd

The default configuration file name of *eigrpd*'s is `eigrpd.conf`. When invocation *eigrpd* searches directory /etc/frr. If `eigrpd.conf` is not there next search current directory. If an integrated config is specified configuration is written into `frr.conf`.

The EIGRP protocol requires interface information maintained by *zebra* daemon. So running *zebra* is mandatory to run *eigrpd*. Thus minimum sequence for running EIGRP is:

```
# zebra -d
# eigrpd -d
```

Please note that *zebra* must be invoked before *eigrpd*.

To stop *eigrpd*, please use:

```
kill `cat /var/run/frr/eigrpd.pid`
```

Certain signals have special meanings to *eigrpd*.

Signal	Meaning
SIGHUP & SIGUSR1	Rotate the log file
SIGINT & SIGTERM	Sweep all installed EIGRP routes and gracefully terminate

eigrpd invocation options. Common options that can be specified (*Common Invocation Options*).

3.7.2 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.

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)
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.

3.7.3 How to Announce EIGRP route

Redistribute routes into EIGRP:

redistribute <babel|bgp|connected|isis|kernel|openfabric|ospf|rip|sharp|static|table> [metric (1-429496
The `redistribute` family of commands imports routing information from other sources into EIGRP's tables. Redistribution may be disabled with the `no` form of the commands.

Note that connected routes on interfaces EIGRP is enabled on are announced by default.

Optionally, various EIGRP metrics may be specified. These metrics will be applied to the imported routes.

3.7.4 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.

show ip eigrp [vrf NAME] neighbor
Display the list of neighbors that have been established within a particular eigrp instance.

3.7.5 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.

3.9 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.

3.9.1 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.

Like other daemons, *isisd* configuration is done in ISIS specific configuration file `isisd.conf`.

3.9.2 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.

router isis WORD [vrf NAME]
Enable or disable the ISIS process by specifying the ISIS domain with 'WORD'. *isisd* does not yet support multiple ISIS processes but you must specify the name of ISIS process. The ISIS process name 'WORD' is then used for interface (see command `ip router isis WORD`).

net XX.XXXX. ... .XXX.XX
Set/Unset network entity title (NET) provided in ISO format.

hostname dynamic
Enable support for dynamic hostname.

area-password [clear | md5] <password>

domain-password [clear | md5] <password>
Configure the authentication password for an area, respectively a domain, as clear text or md5 one.

attached-bit [receive ignore | send]
Set attached bit for inter-area traffic:

- receive If LSP received with attached bit set, create default route to neighbor
- send If L1|L2 router, set attached bit in LSP sent to L1 router

log-adjacency-changes
Log changes in adjacency state.

log-pdu-drops
Log any dropped PDUs.

metric-style [narrow | transition | wide]
Set old-style (ISO 10589) or new-style packet formats:

- narrow Use old style of TLVs with narrow metric
- transition Send and accept both styles of TLVs during transition
- wide Use new style of TLVs to carry wider metric. FRR uses this as a default value

advertise-high-metrics
Advertise high metric value on all interfaces to gracefully shift traffic off the router. Reference: RFC 3277

For narrow metrics, the high metric value is 63; for wide metrics, 16777215; for transition metrics, 62.

set-overload-bit
Set overload bit to avoid any transit traffic.

set-overload-bit on-startup (0-86400)
Set overload bit on startup for the specified duration, in seconds. Reference: RFC 3277

purge-originator
Enable or disable RFC 6232 purge originator identification.

lsp-mtu (128-4352)
Configure the maximum size of generated LSPs, in bytes.

advertise-passive-only
Advertise prefixes of passive interfaces only.

3.9.3 ISIS Timer

lsp-gen-interval [level-1 | level-2] (1-120)
Set minimum interval in seconds between regenerating same LSP, globally, for an area (level-1) or a domain (level-2).

lsp-refresh-interval [level-1 | level-2] (1-65235)
Set LSP refresh interval in seconds, globally, for an area (level-1) or a domain (level-2).

max-lsp-lifetime [level-1 | level-2] (360-65535)
Set LSP maximum LSP lifetime in seconds, globally, for an area (level-1) or a domain (level-2).

spf-interval [level-1 | level-2] (1-120)
Set minimum interval between consecutive SPF calculations in seconds.

3.9.4 ISIS Fast-Reroute

Unless stated otherwise, commands in this section apply to all LFA flavors (local LFA, Remote LFA and TI-LFA).

spf prefix-priority [critical | high | medium] WORD
Assign a priority to the prefixes that match the specified access-list.

By default loopback prefixes have medium priority and non-loopback prefixes have low priority.

fast-reroute priority-limit [critical | high | medium] [level-1 | level-2]
Limit LFA backup computation up to the specified prefix priority.

fast-reroute lfa tiebreaker [downstream | lowest-backup-metric | node-protecting] index (1-255) [level-
Configure a tie-breaker for multiple local LFA backups. Lower indexes are processed first.

fast-reroute load-sharing disable [level-1 | level-2]
Disable load sharing across multiple LFA backups.

fast-reroute remote-lfa prefix-list [WORD] [level-1 | level-2]
Configure a prefix-list to select eligible PQ nodes for remote LFA backups (valid for all protected interfaces).

3.9.5 ISIS region

is-type [level-1 | level-1-2 | level-2-only]
Define the ISIS router behavior:

- level-1 Act as a station router only
- level-1-2 Act as both a station router and an area router
- level-2-only Act as an area router only

3.9.6 ISIS interface

<ip|ipv6> router isis WORD
Activate ISIS adjacency on this interface. Note that the name of ISIS instance must be the same as the one used to configure the ISIS process (see command `router isis WORD`). To enable IPv4, issue `ip router isis WORD`; to enable IPv6, issue `ipv6 router isis WORD`.

isis circuit-type [level-1 | level-1-2 | level-2]
Configure circuit type for interface:

- level-1 Level-1 only adjacencies are formed
- level-1-2 Level-1-2 adjacencies are formed
- level-2-only Level-2 only adjacencies are formed

isis csnp-interval (1-600) [level-1 | level-2]
Set CSNP interval in seconds globally, for an area (level-1) or a domain (level-2).

isis hello padding
Add padding to IS-IS hello packets.

isis hello padding during-adjacency-formation
Add padding to IS-IS hello packets during adjacency formation only.

isis hello-interval (1-600) [level-1 | level-2]
Set Hello interval in seconds globally, for an area (level-1) or a domain (level-2).

isis hello-multiplier (2-100) [level-1 | level-2]
Set multiplier for Hello holding time globally, for an area (level-1) or a domain (level-2).

isis metric [(0-255) | (0-16777215)] [level-1 | level-2]
Set default metric value globally, for an area (level-1) or a domain (level-2). Max value depend if metric support narrow or wide value (see command *metric-style [narrow | transition | wide]*).

isis network point-to-point
Set network type to 'Point-to-Point' (broadcast by default).

isis passive
Configure the passive mode for this interface.

isis password [clear | md5] <password>
Configure the authentication password (clear or encoded text) for the interface.

isis priority (0-127) [level-1 | level-2]
Set priority for Designated Router election, globally, for the area (level-1) or the domain (level-2).

isis psnp-interval (1-120) [level-1 | level-2]
Set PSNP interval in seconds globally, for an area (level-1) or a domain (level-2).

isis three-way-handshake
Enable or disable RFC 5303 Three-Way Handshake for P2P adjacencies. Three-Way Handshake is enabled by default.

isis fast-reroute lfa [level-1 | level-2]
Enable per-prefix local LFA fast reroute link protection.

isis fast-reroute lfa [level-1 | level-2] exclude interface IFNAME
Exclude an interface from the local LFA backup nexthop computation.

isis fast-reroute remote-lfa tunnel mpls-ldp [level-1 | level-2]
Enable per-prefix Remote LFA fast reroute link protection. Note that other routers in the network need to be configured to accept LDP targeted hello messages in order for RLFA to work.

isis fast-reroute remote-lfa maximum-metric (1-16777215) [level-1 | level-2]
Limit Remote LFA PQ node selection within the specified metric.

isis fast-reroute ti-lfa [level-1|level-2] [node-protection [link-fallback]]
Enable per-prefix TI-LFA fast reroute link or node protection. When node protection is used, option link-fallback enables the computation and use of link-protecting LFAs for destinations unprotected by node protection.

3.9.7 Showing ISIS information

show isis [vrf <NAME|all>] summary [json]
Show summary information about ISIS.

show isis hostname
Show information about ISIS node.

show isis [vrf <NAME|all>] interface [detail] [IFNAME] [json]
Show state and configuration of ISIS specified interface, or all interfaces if no interface is given with or without details.

show isis [vrf <NAME|all>] neighbor [detail] [SYSTEMID] [json]
Show state and information of ISIS specified neighbor, or all neighbors if no system id is given with or without details.

show isis [vrf <NAME|all>] database [detail] [LSPID] [json]
Show the ISIS database globally, for a specific LSP id without or with details.

show isis topology [level-1|level-2] [algorithm (128-255)]
Show topology IS-IS paths to Intermediate Systems, globally, in area (level-1) or domain (level-2).

show isis route [level-1|level-2] [prefix-sid|backup] [algorithm (128-255)]
Show the ISIS routing table, as determined by the most recent SPF calculation.

show isis fast-reroute summary [level-1|level-2]
Show information about the number of prefixes having LFA protection, and network-wide LFA coverage.

3.9.8 Traffic Engineering

Note: IS-IS-TE supports RFC 5305 (base TE), RFC 6119 (IPv6) and RFC 7810 / 8570 (Extended Metric) with or without Multi-Topology. All Traffic Engineering information are stored in a database formally named TED. However, best acccuracy is provided without Multi-Topology due to inconsistency of Traffic Engineering Advertisement of 3rd party commercial routers when MT is enabled. At this time, FRR offers partial support for some of the routing protocol extensions that can be used with MPLS-TE. FRR does not currently support a complete RSVP-TE solution.

mpls-te on
Enable Traffic Engineering LSP flooding.

mpls-te router-address <A.B.C.D>
Configure stable IP address for MPLS-TE.

mpls-te router-address ipv6 <X:X::X:X>
Configure stable IPv6 address for MPLS-TE.

mpls-te export
Export Traffic Engineering DataBase to other daemons through the ZAPI Opaque Link State messages.

show isis mpls-te interface

show isis mpls-te interface INTERFACE
Show MPLS Traffic Engineering parameters for all or specified interface.

show isis mpls-te router
Show Traffic Engineering router parameters.

show isis [vrf <NAME|all>] mpls-te database [detail|json]

show isis [vrf <NAME|all>] mpls-te database vertex [WORD] [detail|json]

show isis [vrf <NAME|all>] mpls-te database edge [A.B.C.D|X:X::X:X] [detail|json]

show isis [vrf <NAME|all>] mpls-te database subnet [A.B.C.D/M|X:X::X:X/M] [detail|json]
Show Traffic Engineering Database

See also:

Traffic Engineering

3.9.9 Segment Routing

This is an EXPERIMENTAL support of Segment Routing as per RFC8667 for MPLS dataplane. It supports IPv4, IPv6 and ECMP and has been tested against Cisco & Juniper routers.

Known limitations:

- No support for level redistribution (L1 to L2 or L2 to L1)
- No support for binding SID
- No support for SRMS
- No support for SRLB
- Only one SRGB and default SPF Algorithm is supported

segment-routing on
Enable Segment Routing.

segment-routing global-block (16-1048575) (16-1048575) [local-block (16-1048575) (16-1048575)]
Set the Segment Routing Global Block i.e. the label range used by MPLS to store label in the MPLS FIB for Prefix SID. Note that the block size may not exceed 65535. Optionally sets also the Segment Routing Local Block. The negative command always unsets both.

segment-routing node-msd (1-16)
Set the Maximum Stack Depth supported by the router. The value depend of the MPLS dataplane. E.g. for Linux kernel, since version 4.13 the maximum value is 32.

segment-routing prefix <A.B.C.D/M|X:X::X:X/M> [algorithm (128-255)] <absolute (16-1048575)|index (0-65535) [no-php-flag|explicit-null] [n-flag-clea
prefix. The 'no-php-flag' means NO Penultimate Hop Popping that allows SR node to request to its neighbor to not pop the label. The 'explicit-null' flag allows SR node to request to its neighbor to send IP packet with the EXPLICIT-NULL label. The 'n-flag-clear' option can be used to explicitly clear the Node flag that is set by default for Prefix-SIDs associated to loopback addresses. This option is necessary to configure Anycast-SIDs.

show isis segment-routing node [algorithm (128-255)]
Show detailed information about all learned Segment Routing Nodes.

3.9.10 Flex-Algos (Flex-Algo)

isisd supports some features of RFC 9350 on an MPLS Segment-Routing dataplane. The compatibility has been tested against Cisco.

IS-IS uses by default the *Shortest-Path-First* algorithm that basically calculates paths based on the shortest total metric to the destinations. Flex-Algo allows new algorithms to run in parallel to compute paths in different manners, based on metrics (IGP metric or a new type of metrics such as Traffic Engineering (TE) metric and minimum delay…) and constraints. New metric types are not yet implemented but constraints are already operational. Constraints can restrict paths to links with specific affinities or avoid links with specific affinities. Combinations of these are also possible.

The administrator can configure up to 128 Flex-Algos in an IS-IS area. To do so, it defines a set of Flex-Algo Definitions (FAD) which have the following characteristics:

- a numeric identifier (ID) between 128 and 255 inclusive
- **a set of constraints (basically, include or exclude a certain given set of** links, designated by a admin-group)
- the calculation type (only the *Shortest-Path-First* is currently supported)
- the metric type (only the IGP inherited metric type is currently supported)
- some additional flags (not supported for the moment).

A subset of routers advertises the Flex-Algo Definitions (FAD) to the other routers within an area. In order to use a common set of FADs, each router runs a FAD election process for each locally configured algorithm, using the following rules:

- **If a locally configured FAD is not advertised to the area, the router does not** participate in the particular flex algorithm.
- **If a given flex algorithm is running, the participation in this particular** flex algorithm stops when its advertisements are over.
- **A router includes its own FAD in the election process if and only if it is** advertised to the other routers.
- If only one router advertises the FAD, the FAD is elected.
- **If several FADs are advertised with different priorities, the one with the** highest priority value is selected.
- **If there are multiple advertisements of the FAD with the same highest** priority, the FAD of the router with the highest IS-IS system-ID is selected.

Routers only use the specifications of the elected FAD regardless of the locally configured definitions. If a router does not support one of the FAD characteristics, it stops participating in the Flex-Algo.

For each running Flex-Algo, the Segment-Routing SIDs must be configured with values unique to the algorithm. It allows routers to identify which flex algorithm they must use for a given packet.

The following commands configure Flex-Algo at the 'router isis' configuration level. Segment-Routing prefixes must be configured for the Flex-Algo.

flexible-algorithm (128-255)
Add a Flex-Algo Definition (FAD) and enter the FAD configuration level. The algorithm ID value is in the range of 128 to 255 inclusive.

no flexible-algorithm (128-255)
Unconfigure a Flex-Algo Definition.

affinity-map NAME bit-position (0-255)
Add the specified 'affinity-map'. Affinity-map definitions are used in FADs and in interfaces admin-group definition.

Affinity-maps format in advertisement TLVs use the extended admin-group format defined in the RFC7308 section 2.2. The extended admin-group uses a 256 bits field. If an affinity-map is set, the bit at the extended admin-group 'bit-position' is set 1, else it is set to 0.

The following commands configure Flex-Algo at the 'router isis' and 'flexible-algorithm (128-255)' configuration level.

advertise-definition
Advertise the current FAD to other IS-IS routers by using specific IS-IS TLVs. By default, the definition is is not shared with other routers.

A router can advertise a FAD without participating in the Flex-Algo.

priority (0-255)
Set the specified 'priority' in the current FAD advertisements .

metric-type [igp|te|delay]
Set the 'metric-type' for the current FAD. 'igp' is the default value and refers to the classic 'Shortest-Path-First' algorithm. If the 'te' or the 'delay' metric is selected, the value is advertised but the flex algorithm is disabled locally because these types are not currently supported.

no metric-type
Reset the 'metric-type' to the default 'igp' metric.

affinity exclude-any NAME
Add the specified affinity to the list of exclude-any affinities. The Flex-Algo will compute paths that exclude the segments with any of the specified affinities.

no affinity exclude-any NAME
Remove the specified affinity to the list of exclude-any affinities.

affinity include-all NAME
Add the specified affinity to the list of include-all affinities. The Flex-Algo will compute paths that include the segments with all the specified affinities.

no affinity include-all NAME
Remove the specified affinity to the list of include-all affinities.

affinity include-any NAME
Add the specified affinity to the list of include-any affinities. The Flex-Algo will compute paths that include the segments with any of the specified affinities.

no affinity include-any NAME
Remove the specified affinity to the list of include-any affinities.

The following commands configure Flex-Algo at the 'interface' configuration level.

isis affinity flex-algo NAME
Add the specified affinity to the interface.

no isis affinity flex-algo NAME
Remove the specified affinity from the interface.

The following command show Flex-Algo information:

show isis flex-algo [(128-255)]
Show information about the elected FADs

'show isis route', 'show isis topology' and 'show isis segment-routing node' includes an 'algorithm (128-255)' optional argument. See *Showing ISIS information* and *Segment Routing*.

3.9.11 Debugging ISIS

debug isis adj-packets
IS-IS Adjacency related packets.

debug isis checksum-errors
IS-IS LSP checksum errors.

debug isis events
IS-IS Events.

debug isis local-updates
IS-IS local update packets.

debug isis packet-dump
IS-IS packet dump.

debug isis protocol-errors
IS-IS LSP protocol errors.

debug isis route-events
IS-IS Route related events.

debug isis snp-packets
IS-IS CSNP/PSNP packets.

debug isis spf-events

debug isis spf-statistics

debug isis spf-triggers
IS-IS Shortest Path First Events, Timing and Statistic Data and triggering events.

debug isis update-packets
Update related packets.

debug isis te-events
IS-IS Traffic Engineering events

debug isis sr-events
IS-IS Segment Routing events.

debug isis lfa
IS-IS LFA events.

show debugging isis
Print which ISIS debug level is activate.

3.9.12 ISIS Configuration Examples

A simple example, with MD5 authentication enabled:

```
!
interface eth0
 ip router isis FOO
 isis network point-to-point
 isis circuit-type level-2-only
!
router isis FOO
net 47.0023.0000.0000.0000.0000.0000.0000.1900.0004.00
 metric-style wide
 is-type level-2-only
```

A Traffic Engineering configuration, with Inter-ASv2 support.

First, the `zebra.conf` part:

```
hostname HOSTNAME
password PASSWORD
log file /var/log/zebra.log
!
interface eth0
 ip address 10.2.2.2/24
 link-params
  max-bw 1.25e+07
  max-rsv-bw 1.25e+06
  unrsv-bw 0 1.25e+06
  unrsv-bw 1 1.25e+06
  unrsv-bw 2 1.25e+06
  unrsv-bw 3 1.25e+06
  unrsv-bw 4 1.25e+06
  unrsv-bw 5 1.25e+06
  unrsv-bw 6 1.25e+06
  unrsv-bw 7 1.25e+06
  admin-grp 0xab
!
interface eth1
 ip address 10.1.1.1/24
 link-params
  enable
  metric 100
  max-bw 1.25e+07
  max-rsv-bw 1.25e+06
  unrsv-bw 0 1.25e+06
  unrsv-bw 1 1.25e+06
  unrsv-bw 2 1.25e+06
  unrsv-bw 3 1.25e+06
  unrsv-bw 4 1.25e+06
```

(continues on next page)

(continued from previous page)

```
  unrsv-bw 5 1.25e+06
  unrsv-bw 6 1.25e+06
  unrsv-bw 7 1.25e+06
  neighbor 10.1.1.2 as 65000
```

Then the `isisd.conf` itself:

```
hostname HOSTNAME
password PASSWORD
log file /var/log/isisd.log
!
!
interface eth0
 ip router isis FOO
!
interface eth1
 ip router isis FOO
!
!
router isis FOO
 isis net 47.0023.0000.0000.0000.0000.0000.0000.1900.0004.00
  mpls-te on
  mpls-te router-address 10.1.1.1
!
line vty
```

A Segment Routing configuration, with IPv4, IPv6, SRGB and MSD configuration.

```
hostname HOSTNAME
password PASSWORD
log file /var/log/isisd.log
!
!
interface eth0
 ip router isis SR
 isis network point-to-point
!
interface eth1
 ip router isis SR
!
!
router isis SR
 net 49.0000.0000.0000.0001.00
 is-type level-1
 topology ipv6-unicast
 lsp-gen-interval 2
 segment-routing on
 segment-routing node-msd 8
 segment-routing prefix 10.1.1.1/32 index 100 explicit-null
 segment-routing prefix 2001:db8:1000::1/128 index 101 explicit-null
!
```

3.9.13 ISIS Vrf Configuration Examples

A simple vrf example:

```
!
interface eth0 vrf RED
 ip router isis FOO vrf RED
 isis network point-to-point
 isis circuit-type level-2-only
!
router isis FOO vrf RED
 net 47.0023.0000.0000.0000.0000.0000.0000.1900.0004.00
 metric-style wide
 is-type level-2-only
```

3.10 NHRP

nhrpd is an implementation of the NHRP (Next Hop Routing Protocol). NHRP is described in **RFC 2332**.

NHRP is used to improve the efficiency of routing computer network traffic over NBMA (Non-Broadcast, Multiple Access) networks. NHRP provides an ARP-like solution that allows a system to dynamically learn the NBMA address of the other systems that are part of that network, allowing these systems to directly communicate without requiring traffic to use an intermediate hop.

NHRP is a client-server protocol. The server side is called the NHS (Next Hop Server) or the hub, while a client is referred to as the NHC (Next Hop Client) or the spoke. When a node is configured as an NHC, it registers its address with the NHS which keeps track of all registered spokes. An NHC client can then query the addresses of other clients from NHS allowing all spokes to communicate directly with each other.

Cisco Dynamic Multipoint VPN (DMVPN) is based on NHRP, and frr nhrpd implements this scenario.

3.10.1 Routing Design

nhrpd never handles routing of prefixes itself. You need to run some real routing protocol (e.g. BGP) to advertise routes over the tunnels. What nhrpd does it establishes 'shortcut routes' that optimizes the routing protocol to avoid going through extra nodes in NBMA GRE mesh.

nhrpd does route NHRP domain addresses individually using per-host prefixes. This is similar to Cisco FlexVPN; but in contrast to opennhrp which uses a generic subnet route.

To create NBMA GRE tunnel you might use the following (Linux terminal commands):

```
ip tunnel add gre1 mode gre key 42 ttl 64
ip addr add 10.255.255.2/32 dev gre1
ip link set gre1 up
```

Note that the IP-address is assigned as host prefix to gre1. nhrpd will automatically create additional host routes pointing to gre1 when a connection with these hosts is established.

The gre1 subnet prefix should be announced by routing protocol from the hub nodes (e.g. BGP 'network' announce). This allows the routing protocol to decide which is the closest hub and determine the relay hub on prefix basis when direct tunnel is not established.

nhrpd will redistribute directly connected neighbors to zebra. Within hub nodes, these routes should be internally redistributed using some routing protocol (e.g. iBGP) to allow hubs to be able to relay all traffic.

This can be achieved in hubs with the following bgp configuration (network command defines the GRE subnet):

```
router bgp 65555
 address-family ipv4 unicast
   network 172.16.0.0/16
   redistribute nhrp
 exit-address-family
```

3.10.2 Configuring NHRP

ip nhrp holdtime (1-65000)
Holdtime is the number of seconds that have to pass before stopping to advertise an NHRP NBMA address as valid. It also controls how often NHRP registration requests are sent. By default registrations are sent every one third of the holdtime.

ip nhrp map A.B.C.D|X:X::X:X A.B.C.D|local
Map an IP address of a station to the station's NBMA address.

ip nhrp network-id (1-4294967295)
Enable NHRP on this interface and set the interface's network ID. The network ID is used to allow creating multiple nhrp domains on a router when multiple interfaces are configured on the router. Interfaces configured with the same ID are part of the same logical NBMA network. The ID is a local only parameter and is not sent to other NHRP nodes and so IDs on different nodes do not need to match. When NHRP packets are received on an interface they are assigned to the local NHRP domain for that interface.

ip nhrp nhs A.B.C.D nbma A.B.C.D|FQDN
Configure the Next Hop Server address and its NBMA address.

ip nhrp nhs dynamic nbma A.B.C.D
Configure the Next Hop Server to have a dynamic address and set its NBMA address.

ip nhrp registration no-unique
Allow the client to not set the unique flag in the NHRP packets. This is useful when a station has a dynamic IP address that could change over time.

ip nhrp shortcut
Enable shortcut (spoke-to-spoke) tunnels to allow NHC to talk to each others directly after establishing a connection without going through the hub.

ip nhrp mtu
Configure NHRP advertised MTU.

3.10.3 Hub Functionality

In addition to routing nhrp redistributed host prefixes, the hub nodes are also responsible to send NHRP Traffic Indication messages that trigger creation of the shortcut tunnels.

nhrpd sends Traffic Indication messages based on network traffic captured using NFLOG. Typically you want to send Traffic Indications for network traffic that is routed from gre1 back to gre1 in rate limited manner. This can be achieved with the following iptables rule.

```
iptables -A FORWARD -i gre1 -o gre1 \\
    -m hashlimit --hashlimit-upto 4/minute --hashlimit-burst 1 \\
    --hashlimit-mode srcip,dstip --hashlimit-srcmask 24 --hashlimit-dstmask 24 \\
    --hashlimit-name loglimit-0 -j NFLOG --nflog-group 1 --nflog-range 128
```

You can fine tune the src/dstmask according to the prefix lengths you announce internal, add additional IP range matches, or rate limitation if needed. However, the above should be good in most cases.

This kernel NFLOG target's nflog-group is configured in global nhrp config with:

nhrp nflog-group (1-65535)

To start sending these traffic notices out from hubs, use the nhrp per-interface directive:

ip nhrp redirect

This enable redirect replies on the NHS similar to ICMP redirects except this is managed by the nhrp protocol. This setting allows spokes to communicate with each others directly.

3.10.4 Integration with IKE

nhrpd needs tight integration with IKE daemon for various reasons. Currently only strongSwan is supported as IKE daemon.

nhrpd connects to strongSwan using VICI protocol based on UNIX socket which can be configured using the command below (default to /var/run/charon.vici).

strongSwan currently needs few patches applied. Please check out the original patches at: https://git-old.alpinelinux.org/user/tteras/strongswan/

Actively maintained patches are also available at: https://gitlab.alpinelinux.org/alpine/aports/-/tree/master/main/strongswan

3.10.5 Multicast Functionality

nhrpd can be configured to forward multicast packets, allowing routing protocols that use multicast (such as OSPF) to be supported in the DMVPN network.

This support requires an iptables NFLOG rule to allow nhrpd to intercept multicast packets. A second iptables rule is also usually used to drop the original multicast packet.

```
iptables -A OUTPUT -d 224.0.0.0/24 -o gre1 -j NFLOG --nflog-group 2
iptables -A OUTPUT -d 224.0.0.0/24 -o gre1 -j DROP
```

nhrp multicast-nflog-group (1-65535)
: Sets the nflog group that nhrpd will listen on for multicast packets. This value must match the nflog-group value set in the iptables rule.

ip nhrp map multicast A.B.C.D|X:X::X:X A.B.C.D|dynamic
: Sends multicast packets to the specified NBMA address. If dynamic is specified then destination NBMA address (or addresses) are learnt dynamically.

3.10.6 NHRP Events

nhrp event socket SOCKET
Configure the Unix path for the event socket.

3.10.7 Show NHRP

show [ip|ipv6] nhrp cache [json]
Dump the cache entries.

show [ip|ipv6] nhrp opennhrp [json]
Dump the cache entries with opennhrp format.

show [ip|ipv6] nhrp nhs [json]
Dump the hub context.

show dmvpn [json]
Dump the security contexts.

3.10.8 Configuration Example

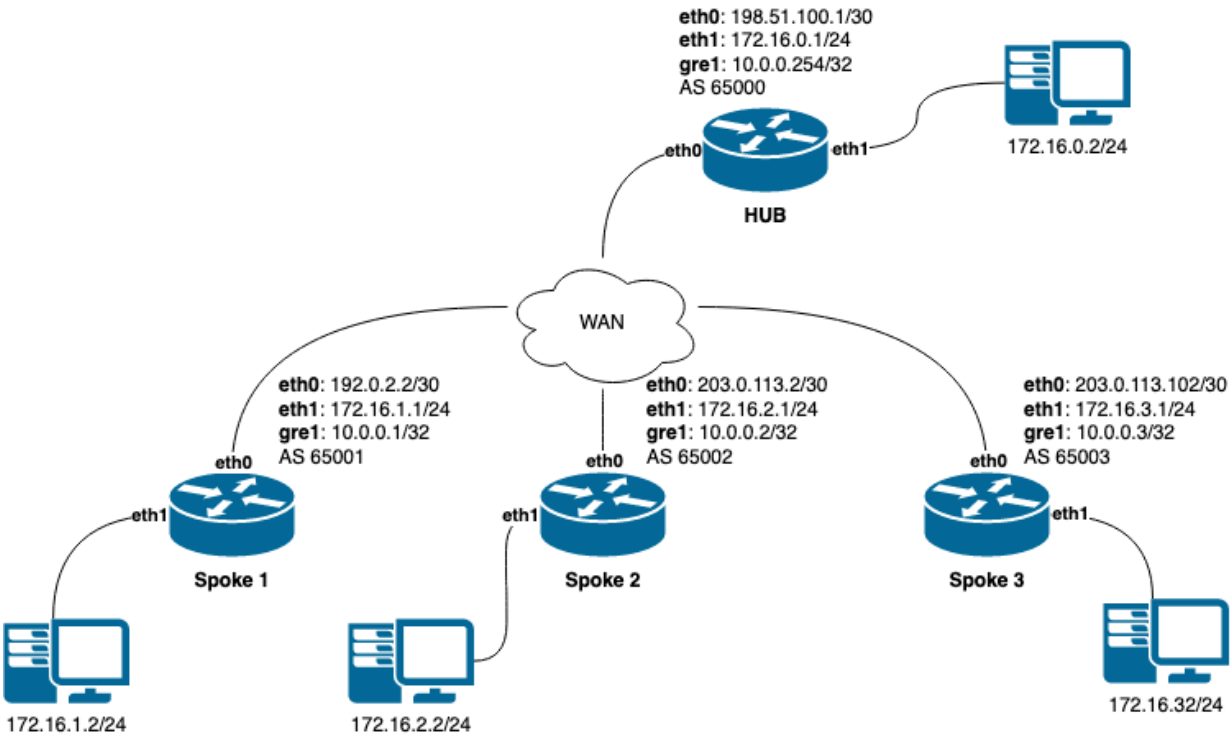


Fig. 5: image

IPSec configurration example

This changes required on all nodes as HUB and Spokes.

ipsec.conf file

```
config setup
conn dmvpn
    authby=secret
    auto=add
    keyexchange=ikev2
    ike=aes256-aes256-sha256-modp2048
    esp=aes256-aes256-sha256-modp2048
    dpdaction=clear
    dpddelay=300s
    left=%any
    leftid=%any
    right=%any
    rightid=%any
    leftprotoport=gre
    rightprotoport=gre
    type=transport
    keyingtries=%forever
```

ipsec.secrets file

```
%any : PSK "some_s3cret!"
```

HUB configuration example

Creating gre interface

```
ip tunnel add gre1 mode gre key 42 ttl 64
ip addr add 10.0.0.254/32 dev gre1
ip link set gre1 up
```

Adding iptables rules to provide possibility shortcut tunnels and connect spokes directly

```
iptables -A FORWARD -i gre1 -o gre1 \\
    -m hashlimit --hashlimit-upto 4/minute --hashlimit-burst 1 \\
    --hashlimit-mode srcip,dstip --hashlimit-srcmask 24 --hashlimit-dstmask 24 \\
    --hashlimit-name loglimit-0 -j NFLOG --nflog-group 1 --nflog-range 128
```

FRR config on HUB

```
nhrp nflog-group 1
!
interface gre1
 description DMVPN Tunnel Interface
 ip address 10.0.0.254/32
 ip nhrp network-id 1
 ip nhrp redirect
 ip nhrp registration no-unique
```

(continues on next page)

(continued from previous page)

```
 ip nhrp shortcut
 tunnel protection vici profile dmvpn
 tunnel source eth0
 !
 router bgp 65000
  bgp router-id 10.0.0.254
  no bgp ebgp-requires-policy
  neighbor SPOKES peer-group
  neighbor SPOKES disable-connected-check
  neighbor 10.0.0.1 remote-as 65001
  neighbor 10.0.0.1 peer-group SPOKES
  neighbor 10.0.0.2 remote-as 65002
  neighbor 10.0.0.2 peer-group SPOKES
  neighbor 10.0.0.3 remote-as 65003
  neighbor 10.0.0.3 peer-group SPOKES
  !
  address-family ipv4 unicast
   network 172.16.0.0/24
   redistribute nhrp
  exit-address-family
```

Spoke1 configuration

Creating gre interface

```
ip tunnel add gre1 mode gre key 42 ttl 64
ip addr add 10.0.0.1/32 dev gre1
ip link set gre1 up
```

FRR config on Spoke1

```
interface gre1
 description DMVPN Tunnel Interface
 ip address 10.0.0.1/32
 ip nhrp network-id 1
 ip nhrp nhs dynamic nbma 198.51.100.1
 ip nhrp redirect
 ip nhrp registration no-unique
 ip nhrp shortcut
 no link-detect
 tunnel protection vici profile dmvpn
 tunnel source eth0
!
router bgp 65001
 no bgp ebgp-requires-policy
 neighbor 10.0.0.254 remote-as 65000
 neighbor 10.0.0.254 disable-connected-check
 !
 address-family ipv4 unicast
  network 172.16.1.0/24
 exit-address-family
```

Spoke2 configuration

Creating gre interface

```
ip tunnel add gre1 mode gre key 42 ttl 64
ip addr add 10.0.0.1/32 dev gre1
ip link set gre1 up
```

FRR config on Spoke2

```
interface gre1
 description DMVPN Tunnel Interface
 ip address 10.0.0.2/32
 ip nhrp network-id 1
 ip nhrp nhs dynamic nbma 198.51.100.1
 ip nhrp redirect
 ip nhrp registration no-unique
 ip nhrp shortcut
 no link-detect
 tunnel protection vici profile dmvpn
 tunnel source eth0
!
router bgp 65002
 no bgp ebgp-requires-policy
 neighbor 10.0.0.254 remote-as 65000
 neighbor 10.0.0.254 disable-connected-check
 !
 address-family ipv4 unicast
  network 172.16.2.0/24
 exit-address-family
```

Spoke3 configuration

Creating gre interface

```
ip tunnel add gre1 mode gre key 42 ttl 64
ip addr add 10.0.0.3/32 dev gre1
ip link set gre1 up
```

FRR config on Spoke3

```
interface gre1
 description DMVPN Tunnel Interface
 ip address 10.0.0.3/32
 ip nhrp network-id 1
 ip nhrp nhs dynamic nbma 198.51.100.1
 ip nhrp redirect
 ip nhrp registration no-unique
 ip nhrp shortcut
 no link-detect
 tunnel protection vici profile dmvpn
 tunnel source eth0
!
```

(continues on next page)

(continued from previous page)

```
router bgp 65003
 no bgp ebgp-requires-policy
 neighbor 10.0.0.254 remote-as 65000
 neighbor 10.0.0.254 disable-connected-check
 !
 address-family ipv4 unicast
  network 172.16.3.0/24
 exit-address-family
```

3.11 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.

3.11.1 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 *ip ospf passive [A.B.C.D]*.

- Network LSA

  On multi-access links (e.g. ethernets, certain kinds of ATM and X.25 configurations), routers elect a DR. The DR is responsible for originating a Network LSA, which helps reduce the information needed to describe multi-access networks with multiple routers attached. The DR also acts as a hub for the flooding of LSA s on that link, thus reducing flooding overheads.

  The contents of the Network LSA describes the:

  – Subnet Mask

  As the LSA ID of a Network LSA must be the IP address of the DR, the Subnet Mask together with the LSA ID gives you the network address.

  – Attached Routers

  Each router fully-adjacent with the DR is listed in the LSA, by their Router-ID. This allows the corresponding Router LSA s to be easily retrieved from the LSDB.

Summary of Link State LSAs:

LSA Type	LSA ID	LSA Data Describes
Router LSA	Router ID	The OSPF enabled links of the router, within a specific link-state area.
Network LSA	The IP address of the DR for the network	The subnet mask of the network and the Router IDs of all routers on the network

With an LSDB composed of just these two types of LSA, it is possible to construct a directed graph of the connectivity between all routers and networks in a given OSPF link-state area. So, not surprisingly, when OSPF routers build updated routing tables, the first stage of SPF calculation concerns itself only with these two LSA types.

Link-State LSA Examples

The example below shows two LSA s, both originated by the same router (Router ID 192.168.0.49) and with the same LSA ID (192.168.0.49), but of different LSA types.

The first LSA being the router LSA describing 192.168.0.49's links: 2 links to multi-access networks with fully-adjacent neighbours (i.e. Transit links) and 1 being a Stub link (no adjacent neighbours).

The second LSA being a Network LSA, for which 192.168.0.49 is the DR, listing the Router IDs of 4 routers on that network which are fully adjacent with 192.168.0.49.

```
# show ip ospf database router 192.168.0.49

       OSPF Router with ID (192.168.0.53)

                Router Link States (Area 0.0.0.0)

  LS age: 38
  Options: 0x2  : *|-|-|-|-|-|E|*
  LS Flags: 0x6
  Flags: 0x2 : ASBR
  LS Type: router-LSA
  Link State ID: 192.168.0.49
  Advertising Router: 192.168.0.49
  LS Seq Number: 80000f90
  Checksum: 0x518b
  Length: 60
   Number of Links: 3

    Link connected to: a Transit Network
     (Link ID) Designated Router address: 192.168.1.3
     (Link Data) Router Interface address: 192.168.1.3
      Number of TOS metrics: 0
       TOS 0 Metric: 10

    Link connected to: a Transit Network
     (Link ID) Designated Router address: 192.168.0.49
     (Link Data) Router Interface address: 192.168.0.49
      Number of TOS metrics: 0
       TOS 0 Metric: 10
```

(continues on next page)

(continued from previous page)

```
    Link connected to: Stub Network
     (Link ID) Net: 192.168.3.190
     (Link Data) Network Mask: 255.255.255.255
      Number of TOS metrics: 0
       TOS 0 Metric: 39063
# show ip ospf database network 192.168.0.49

       OSPF Router with ID (192.168.0.53)

                Net Link States (Area 0.0.0.0)

  LS age: 285
  Options: 0x2  : *|-|-|-|-|-|E|*
  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
  |        |           |
```

(continues on next page)

(continued from previous page)

```
  |         |     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
```

(continues on next page)

(continued from previous page)

```
  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)
    (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.

3.11.2 Configuring OSPF

ospfd accepts all *Common Invocation Options*.

-n, --instance
Specify the instance number for this invocation of *ospfd*.

-a, --apiserver
Enable the OSPF API server. This is required to use `ospfclient`.

ospfd must acquire interface information from *zebra* in order to function. Therefore *zebra* must be running before invoking *ospfd*. Also, if *zebra* is restarted then *ospfd* must be too.

Like other daemons, *ospfd* configuration is done in OSPF specific configuration file `ospfd.conf` when the integrated config is not used.

Multi-instance Support

OSPF supports multiple instances. Each instance is identified by a positive nonzero integer that must be provided when adding configuration items specific to that instance. Enabling instances is done with `/etc/frr/daemons` in the following manner:

```
...
ospfd=yes
ospfd_instances=1,5,6
...
```

The `ospfd_instances` variable controls which instances are started and what their IDs are. In this example, after starting FRR you should see the following processes:

```
# ps -ef | grep "ospfd"
frr      11816     1  0 17:30 ?        00:00:00 /usr/lib/frr/ospfd --daemon -A 127.0.0.1␣
↪-n 1
frr      11822     1  0 17:30 ?        00:00:00 /usr/lib/frr/ospfd --daemon -A 127.0.0.1␣
↪-n 2
frr      11828     1  0 17:30 ?        00:00:00 /usr/lib/frr/ospfd --daemon -A 127.0.0.1␣
↪-n 3
```

The instance number should be specified in the config when addressing a particular instance:

```
router ospf 5
   ospf router-id 1.2.3.4
   area 0.0.0.0 authentication message-digest
   ...
```

Routers

To start OSPF process you have to specify the OSPF router.

router ospf [{(1-65535)|vrf NAME}]
Enable or disable the OSPF process.

Multiple instances don't support *vrf NAME*.

ospf router-id A.B.C.D
This sets the router-ID of the OSPF process. The router-ID may be an IP address of the router, but need not be - it can be any arbitrary 32bit number. However it MUST be unique within the entire OSPF domain to the OSPF speaker - bad things will happen if multiple OSPF speakers are configured with the same router-ID! If one is not specified then *ospfd* will obtain a router-ID automatically from *zebra*.

ospf abr-type TYPE
type can be cisco|ibm|shortcut|standard. The "Cisco" and "IBM" types are equivalent.

The OSPF standard for ABR behaviour does not allow an ABR to consider routes through non-backbone areas when its links to the backbone are down, even when there are other ABRs in attached non-backbone areas which still can reach the backbone - this restriction exists primarily to ensure routing-loops are avoided.

With the "Cisco" or "IBM" ABR type, the default in this release of FRR, this restriction is lifted, allowing an ABR to consider summaries learned from other ABRs through non-backbone areas, and hence route via non-backbone areas as a last resort when, and only when, backbone links are down.

Note that areas with fully-adjacent virtual-links are considered to be "transit capable" and can always be used to route backbone traffic, and hence are unaffected by this setting (*area A.B.C.D virtual-link A.B.C.D*).

More information regarding the behaviour controlled by this command can be found in RFC 3509, and *draft-ietf-ospf-shortcut-abr-02.txt*.

Quote: "Though the definition of the ABR in the OSPF specification does not require a router with multiple attached areas to have a backbone connection, it is actually necessary to provide successful routing to the inter-area and external destinations. If this requirement is not met, all traffic destined for the areas not connected to such an ABR or out of the OSPF domain, is dropped. This document describes alternative ABR behaviors implemented in Cisco and IBM routers."

ospf rfc1583compatibility
RFC 2328, the successor to RFC 1583, suggests according to section G.2 (changes) in section 16.4 a change to the path preference algorithm that prevents possible routing loops that were possible in the old version of OSPFv2. More specifically it demands that inter-area paths and intra-area backbone path are now of equal preference but still both preferred to external paths.

This command should NOT be set normally.

log-adjacency-changes [detail]
Configures ospfd to log changes in adjacency. With the optional detail argument, all changes in adjacency status are shown. Without detail, only changes to full or regressions are shown.

passive-interface default
Make all interfaces that belong to this router passive by default. For the description of passive interface look at *ip ospf passive [A.B.C.D]*. Per-interface configuration takes precedence over the default value.

timers throttle spf (0-600000) (0-600000) (0-600000)
This command sets the initial *delay*, the *initial-holdtime* and the *maximum-holdtime* between when SPF is calculated and the event which triggered the calculation. The times are specified in milliseconds and must be in the range of 0 to 600000 milliseconds.

The *delay* specifies the minimum amount of time to delay SPF calculation (hence it affects how long SPF calculation is delayed after an event which occurs outside of the holdtime of any previous SPF calculation, and also serves as a minimum holdtime).

Consecutive SPF calculations will always be separated by at least 'hold-time' milliseconds. The hold-time is adaptive and initially is set to the *initial-holdtime* configured with the above command. Events which occur within the holdtime of the previous SPF calculation will cause the holdtime to be increased by *initial-holdtime*, bounded by the *maximum-holdtime* configured with this command. If the adaptive hold-time elapses without any SPF-triggering event occurring then the current holdtime is reset to the *initial-holdtime*. The current holdtime can be viewed with `show ip ospf`, where it is expressed as a multiplier of the *initial-holdtime*.

```
router ospf
timers throttle spf 200 400 10000
```

In this example, the *delay* is set to 200ms, the initial holdtime is set to 400ms and the *maximum holdtime* to 10s. Hence there will always be at least 200ms between an event which requires SPF calculation and the actual SPF calculation. Further consecutive SPF calculations will always be separated by between 400ms to 10s, the hold-time increasing by 400ms each time an SPF-triggering event occurs within the hold-time of the previous SPF calculation.

This command supersedes the *timers spf* command in previous FRR releases.

max-metric router-lsa [on-startup (5-86400)|on-shutdown (5-100)]

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

This support may be enabled administratively (and indefinitely) or conditionally. Conditional enabling of max-metric router-lsas can be for a period of seconds after startup and/or for a period of seconds prior to shutdown.

Enabling this for a period after startup allows OSPF to converge fully first without affecting any existing routes used by other routers, while still allowing any connected stub links and/or redistributed routes to be reachable. Enabling this for a period of time in advance of shutdown allows the router to gracefully excuse itself from the OSPF domain.

Enabling this feature administratively allows for administrative intervention for whatever reason, for an indefinite period of time. Note that if the configuration is written to file, this administrative form of the stub-router command will also be written to file. If *ospfd* is restarted later, the command will then take effect until manually deconfigured.

Configured state of this feature as well as current status, such as the number of second remaining till on-startup or on-shutdown ends, can be viewed with the `show ip ospf` command.

auto-cost reference-bandwidth (1-4294967)
This sets the reference bandwidth for cost calculations, where this bandwidth is considered equivalent to an OSPF cost of 1, specified in Mbits/s. The default is 100Mbit/s (i.e. a link of bandwidth 100Mbit/s or higher will have a cost of 1. Cost of lower bandwidth links will be scaled with reference to this cost).

This configuration setting MUST be consistent across all routers within the OSPF domain.

network A.B.C.D/M area A.B.C.D

network A.B.C.D/M area (0-4294967295)
This command specifies the OSPF enabled interface(s). If the interface has an address from range 192.168.1.0/24 then the command below enables ospf on this interface so router can provide network information to the other ospf routers via this interface.

```
router ospf
network 192.168.1.0/24 area 0.0.0.0
```

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

Note that the behavior when there is a peer address defined on an interface changed after release 0.99.7. Currently, if a peer prefix has been configured, then we test whether the prefix in the network command contains the destination prefix. Otherwise, we test whether the network command prefix contains the local address prefix of the interface.

It is also possible to enable OSPF on a per interface/subnet basis using the interface command (*ip ospf area AREA [ADDR]*). However, mixing both network commands (`network`) and interface commands (`ip ospf`) on the same router is not supported.

proactive-arp
This command enables or disables sending ARP requests to update neighbor table entries. It speeds up convergence for /32 networks on a P2P connection.

This feature is enabled by default.

clear ip ospf [(1-65535)] process
This command can be used to clear the ospf process data structures. This will clear the ospf neighborship as well and it will get re-established. This will clear the LSDB too. This will be helpful when there is a change in router-id and if user wants the router-id change to take effect, user can use this cli instead of restarting the ospfd daemon.

clear ip ospf [(1-65535)] neighbor
This command can be used to clear the ospf neighbor data structures. This will clear the ospf neighborship and

it will get re-established. This command can be used when the neighbor state get stuck at some state and this can be used to recover it from that state.

maximum-paths (1-64)
Use this command to control the maximum number of equal cost paths to reach a specific destination. The upper limit may differ if you change the value of MULTIPATH_NUM during compilation. The default is MULTIPATH_NUM (64).

write-multiplier (1-100)
Use this command to tune the amount of work done in the packet read and write threads before relinquishing control. The parameter is the number of packets to process before returning. The defult value of this parameter is 20.

socket buffer <send | recv | all> (1-4000000000)
This command controls the ospf instance's socket buffer sizes. The 'no' form resets one or both values to the default.

no socket-per-interface
Ordinarily, ospfd uses a socket per interface for sending packets. This command disables those per-interface sockets, and causes ospfd to use a single socket per ospf instance for sending and receiving packets.

Areas

area A.B.C.D range A.B.C.D/M [advertise [cost (0-16777215)]]

area (0-4294967295) range A.B.C.D/M [advertise [cost (0-16777215)]]
Summarize intra area paths from specified area into one Type-3 summary-LSA announced to other areas. This command can be used only in ABR and ONLY router-LSAs (Type-1) and network-LSAs (Type-2) (i.e. LSAs with scope area) can be summarized. Type-5 AS-external-LSAs can't be summarized - their scope is AS.

```
router ospf
 network 192.168.1.0/24 area 0.0.0.0
 network 10.0.0.0/8 area 0.0.0.10
 area 0.0.0.10 range 10.0.0.0/8
```

With configuration above one Type-3 Summary-LSA with routing info 10.0.0.0/8 is announced into backbone area if area 0.0.0.10 contains at least one intra-area network (i.e. described with router or network LSA) from this range.

area A.B.C.D range A.B.C.D/M not-advertise

area (0-4294967295) range A.B.C.D/M not-advertise
Instead of summarizing intra area paths filter them - i.e. intra area paths from this range are not advertised into other areas. This command makes sense in ABR only.

area A.B.C.D range A.B.C.D/M {substitute A.B.C.D/M|cost (0-16777215)}

area (0-4294967295) range A.B.C.D/M {substitute A.B.C.D/M|cost (0-16777215)}
Substitute summarized prefix with another prefix.

```
router ospf
 network 192.168.1.0/24 area 0.0.0.0
 network 10.0.0.0/8 area 0.0.0.10
 area 0.0.0.10 range 10.0.0.0/8 substitute 11.0.0.0/8
```

One Type-3 summary-LSA with routing info 11.0.0.0/8 is announced into backbone area if area 0.0.0.10 contains at least one intra-area network (i.e. described with router-LSA or network-LSA) from range 10.0.0.0/8.

By default, the metric of the summary route is calculated as the highest metric among the summarized routes. The *cost* option, however, can be used to set an explicit metric.

This command makes sense in ABR only.

area A.B.C.D virtual-link A.B.C.D

area (0-4294967295) virtual-link A.B.C.D

area A.B.C.D shortcut

area (0-4294967295) shortcut
Configure the area as Shortcut capable. See **RFC 3509**. This requires that the 'abr-type' be set to 'shortcut'.

area A.B.C.D stub

area (0-4294967295) stub
Configure the area to be a stub area. That is, an area where no router originates routes external to OSPF and hence an area where all external routes are via the ABR(s). Hence, ABRs for such an area do not need to pass AS-External LSAs (type-5s) or ASBR-Summary LSAs (type-4) into the area. They need only pass Network-Summary (type-3) LSAs into such an area, along with a default-route summary.

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

area (0-4294967295) stub no-summary
Prevents an *ospfd* ABR from injecting inter-area summaries into the specified stub area.

area A.B.C.D nssa

area (0-4294967295) nssa
Configure the area to be a NSSA (Not-So-Stubby Area). This is an area that allows OSPF to import external routes into a stub area via a new LSA type (type 7). An NSSA autonomous system boundary router (ASBR) will generate this type of LSA. The area border router (ABR) translates the LSA type 7 into LSA type 5, which is propagated into the OSPF domain. NSSA areas are defined in RFC 3101.

area A.B.C.D nssa suppress-fa

area (0-4294967295) nssa suppress-fa
Configure the router to set the forwarding address to 0.0.0.0 in all LSA type 5 translated from LSA type 7. The router needs to be elected the translator of the area for this command to take effect. This feature causes routers that are configured not to advertise forwarding addresses into the backbone to direct forwarded traffic to the NSSA ABR translator.

area A.B.C.D nssa default-information-originate [metric-type (1-2)] [metric (0-16777214)]

area (0-4294967295) nssa default-information-originate [metric-type (1-2)] [metric (0-16777214)]
NSSA ABRs and ASBRs can be configured with the *default-information-originate* option to originate a Type-7 default route into the NSSA area. In the case of NSSA ASBRs, the origination of the default route is conditioned to the existence of a default route in the RIB that wasn't learned via the OSPF protocol.

area A.B.C.D nssa range A.B.C.D/M [<not-advertise|cost (0-16777215)>]

area (0-4294967295) nssa range A.B.C.D/M [<not-advertise|cost (0-16777215)>]
Summarize a group of external subnets into a single Type-7 LSA, which is then translated to a Type-5 LSA and avertised to the backbone. This command can only be used at the area boundary (NSSA ABR router).

By default, the metric of the summary route is calculated as the highest metric among the summarized routes. The *cost* option, however, can be used to set an explicit metric.

The *not-advertise* option, when present, prevents the summary route from being advertised, effectively filtering the summarized routes.

area A.B.C.D default-cost (0-16777215)
Set the cost of default-summary LSAs announced to stubby areas.

area A.B.C.D export-list NAME

area (0-4294967295) export-list NAME
Filter Type-3 summary-LSAs announced to other areas originated from intra- area paths from specified area.

```
router ospf
 network 192.168.1.0/24 area 0.0.0.0
 network 10.0.0.0/8 area 0.0.0.10
 area 0.0.0.10 export-list foo
!
access-list foo permit 10.10.0.0/16
access-list foo deny any
```

With example above any intra-area paths from area 0.0.0.10 and from range 10.10.0.0/16 (for example 10.10.1.0/24 and 10.10.2.128/30) are announced into other areas as Type-3 summary-LSA's, but any others (for example 10.11.0.0/16 or 10.128.30.16/30) aren't.

This command is only relevant if the router is an ABR for the specified area.

area A.B.C.D import-list NAME

area (0-4294967295) import-list NAME
Same as export-list, but it applies to paths announced into specified area as Type-3 summary-LSAs.

area A.B.C.D filter-list prefix NAME in

area A.B.C.D filter-list prefix NAME out

area (0-4294967295) filter-list prefix NAME in

area (0-4294967295) filter-list prefix NAME out
Filtering Type-3 summary-LSAs to/from area using prefix lists. This command makes sense in ABR only.

area A.B.C.D authentication

area (0-4294967295) authentication
Specify that simple password authentication should be used for the given area.

area A.B.C.D authentication message-digest

area (0-4294967295) authentication message-digest
Specify that OSPF packets must be authenticated with MD5 HMACs within the given area. Keying material must also be configured on a per-interface basis (`ip ospf message-digest-key`).

MD5 authentication may also be configured on a per-interface basis (*ip ospf authentication message-digest*). Such per-interface settings will override any per-area authentication setting.

Interfaces

ip ospf area AREA [ADDR]
Enable OSPF on the interface, optionally restricted to just the IP address given by *ADDR*, putting it in the *AREA* area. If you have a lot of interfaces, and/or a lot of subnets, then enabling OSPF via this command instead of (*network A.B.C.D/M area A.B.C.D*) may result in a slight performance improvement.

Notice that, mixing both network commands (`network`) and interface commands (`ip ospf`) on the same router is not supported. If (`ip ospf`) is present, (`network`) commands will fail.

ip ospf authentication-key AUTH_KEY
Set OSPF authentication key to a simple password. After setting *AUTH_KEY*, all OSPF packets are authenticated. *AUTH_KEY* has length up to 8 chars.

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

ip ospf authentication message-digest
Specify that MD5 HMAC authentication must be used on this interface. MD5 keying material must also be configured. Overrides any authentication enabled on a per-area basis (*area A.B.C.D authentication message-digest*)

Note that OSPF MD5 authentication requires that time never go backwards (correct time is NOT important, only that it never goes backwards), even across resets, if ospfd is to be able to promptly reestablish adjacencies with its neighbours after restarts/reboots. The host should have system time be set at boot from an external or non-volatile source (e.g. battery backed clock, NTP, etc.) or else the system clock should be periodically saved to non-volatile storage and restored at boot if MD5 authentication is to be expected to work reliably.

ip ospf message-digest-key KEYID md5 KEY
Set OSPF authentication key to a cryptographic password. The cryptographic algorithm is MD5.

KEYID identifies secret key used to create the message digest. This ID is part of the protocol and must be consistent across routers on a link.

KEY is the actual message digest key, of up to 16 chars (larger strings will be truncated), and is associated with the given KEYID.

ip ospf cost (1-65535)
Set link cost for the specified interface. The cost value is set to router-LSA's metric field and used for SPF calculation.

ip ospf dead-interval (1-65535)

ip ospf dead-interval minimal hello-multiplier (2-20)
Set number of seconds for RouterDeadInterval timer value used for Wait Timer and Inactivity Timer. This value must be the same for all routers attached to a common network. The default value is 40 seconds.

If 'minimal' is specified instead, then the dead-interval is set to 1 second and one must specify a hello-multiplier. The hello-multiplier specifies how many Hellos to send per second, from 2 (every 500ms) to 20 (every 50ms). Thus one can have 1s convergence time for OSPF. If this form is specified, then the hello-interval advertised in Hello packets is set to 0 and the hello-interval on received Hello packets is not checked, thus the hello-multiplier need NOT be the same across multiple routers on a common link.

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

This command has no effect if *ip ospf dead-interval minimal hello-multiplier (2-20)* is also specified for the interface.

ip ospf graceful-restart hello-delay (1-1800)
Set the length of time during which Grace-LSAs are sent at 1-second intervals while coming back up after an unplanned outage. During this time, no hello packets are sent.

A higher hello delay will increase the chance that all neighbors are notified about the ongoing graceful restart before receiving a hello packet (which is crucial for the graceful restart to succeed). The hello delay shouldn't be set too high, however, otherwise the adjacencies might time out. As a best practice, it's recommended to set the hello delay and hello interval with the same values. The default value is 10 seconds.

ip ospf network (broadcast|non-broadcast|point-to-multipoint [delay-reflood]|point-to-point [dmvpn])
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*. If you are doing this you *must* set the net.ipv4.conf.<interface name>.rp_filter value to 0. In order for the ospf multicast packets to be delivered by the kernel.

When used in a DMVPN network at a spoke, this OSPF will be configured in point-to-point, but the HUB will be a point-to-multipoint. To make this topology work, specify the optional 'dmvpn' parameter at the spoke.

When the network is configured as point-to-multipoint and *delay-reflood* is specified, LSAs received on the interface from neighbors on the interface will not be flooded back out on the interface immediately. Rather, they will be added to the neighbor's link state retransmission list and only sent to the neighbor if the neighbor doesn't acknowledge the LSA prior to the link state retransmission timer expiring.

Set explicitly network type for specified interface.

ip ospf priority (0-255)
Set RouterPriority integer value. The router with the highest priority will be more eligible to become Designated Router. Setting the value to 0, makes the router ineligible to become Designated Router. The default value is 1.

ip ospf retransmit-interval (1-65535)
Set number of seconds for RxmtInterval timer value. This value is used when retransmitting Database Description and Link State Request packets. The default value is 5 seconds.

ip ospf transmit-delay (1-65535) [A.B.C.D]
Set number of seconds for InfTransDelay value. LSAs' age should be incremented by this value when transmitting. The default value is 1 second.

ip ospf passive [A.B.C.D]
Do not speak OSPF on the interface, but do advertise the interface as a stub link in the router-LSA for this router. This allows one to advertise addresses on such connected interfaces without having to originate AS-External/Type-5 LSAs (which have global flooding scope) - as would occur if connected addresses were redistributed into OSPF (*Redistribution*). This is the only way to advertise non-OSPF links into stub areas.

ip ospf area (A.B.C.D|(0-4294967295))
Enable ospf on an interface and set associated area.

3.11.3 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 <babel|bgp|connected|eigrp|isis|kernel|openfabric|ospf|rip|sharp|static|table> [metric-typ
Redistribute routes of the specified protocol or kind into OSPF, with the metric type and metric set if specified, filtering the routes using the given route-map if specified. Redistributed routes may also be filtered with distribute-lists, see *ospf distribute-list configuration*.

Redistributed routes are distributed as into OSPF as Type-5 External LSAs into links to areas that accept external routes, Type-7 External LSAs for NSSA areas and are not redistributed at all into Stub areas, where external routes are not permitted.

Note that for connected routes, one may instead use the *ip ospf passive [A.B.C.D]* configuration.

default-information originate

default-information originate metric (0-16777214)

default-information originate metric (0-16777214) metric-type (1|2)

default-information originate metric (0-16777214) metric-type (1|2) route-map WORD

default-information originate always

default-information originate always metric (0-16777214)

default-information originate always metric (0-16777214) metric-type (1|2)

default-information originate always metric (0-16777214) metric-type (1|2) route-map WORD
Originate an AS-External (type-5) LSA describing a default route into all external-routing capable areas, of the specified metric and metric type. If the 'always' keyword is given then the default is always advertised, even when there is no default present in the routing table.

distribute-list NAME out <kernel|connected|static|rip|isis|bgp|eigrp|nhrp|table|vnc|babel|openfabric>
Apply the access-list filter, NAME, to redistributed routes of the given type before allowing the routes to be redistributed into OSPF (*ospf redistribution*).

default-metric (0-16777214)

distance (1-255)

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

3.11.4 Graceful Restart

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

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

When Graceful Restart is enabled and the ospfd daemon crashes or is killed abruptely (e.g. SIGKILL), it will attempt an unplanned Graceful Restart once it restarts.

graceful-restart helper enable [A.B.C.D]
Configure Graceful Restart (RFC 3623) helper support. By default, helper support is disabled for all neighbours. This config enables/disables helper support on this router for all neighbours. To enable/disable helper support for a specific neighbour, the router-id (A.B.C.D) has to be specified.

graceful-restart helper strict-lsa-checking
If 'strict-lsa-checking' is configured then the helper will abort the Graceful Restart when a LSA change occurs which affects the restarting router. By default 'strict-lsa-checking' is enabled"

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

graceful-restart helper planned-only
It helps to support as HELPER only for planned restarts. By default, it supports both planned and unplanned outages.

graceful-restart prepare ip ospf
Initiate a graceful restart for all OSPF instances configured with the "graceful-restart" command. The ospfd daemon should be restarted during the instance-specific grace period, otherwise the graceful restart will fail.

This is an EXEC-level command.

3.11.5 Showing Information

show ip ospf [vrf <NAME|all>] [json]
Show information on a variety of general OSPF and area state and configuration information.

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

show ip ospf neighbor [json]

show ip ospf [vrf <NAME|all>] neighbor INTERFACE [json]

show ip ospf neighbor detail [json]

show ip ospf [vrf <NAME|all>] neighbor A.B.C.D [detail] [json]

show ip ospf [vrf <NAME|all>] neighbor INTERFACE detail [json]
Display lsa information of LSDB. Json o/p of this command covers base route information i.e all LSAs except opaque lsa info.

show ip ospf [vrf <NAME|all>] database [self-originate] [json]
Show the OSPF database summary.

show ip ospf [vrf <NAME|all>] database max-age [json]
Show all MaxAge LSAs present in the OSPF link-state database.

show ip ospf [vrf <NAME|all>] database detail [LINK-STATE-ID] [adv-router A.B.C.D] [json]

show ip ospf [vrf <NAME|all>] database detail [LINK-STATE-ID] [self-originate] [json]

show ip ospf [vrf <NAME|all>] database (asbr-summary|external|network|router|summary|nssa-external|opaq
B.C.D] [json]

show ip ospf [vrf <NAME|all>] database (asbr-summary|external|network|router|summary|nssa-external|opaq
Show detailed information about the OSPF link-state database.

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

show ip ospf [vrf <NAME|all>] border-routers [json]
Show the list of ABR and ASBR border routers summary learnt via OSPFv2 Type-3 (Summary LSA) and Type-4 (Summary ASBR LSA). User can get that information as JSON format when `json` keyword at the end of cli is presented.

show ip ospf graceful-restart helper [detail] [json]
Displays the Grcaeful Restart Helper details including helper config changes.

3.11.6 Opaque LSA

ospf opaque-lsa

capability opaque
ospfd supports Opaque LSA (**RFC 2370**) as partial support for MPLS Traffic Engineering LSAs. The opaque-lsa capability must be enabled in the configuration. An alternate command could be "mpls-te on" (*Traffic Engineering*). Note that FRR offers only partial support for some of the routing protocol extensions that are used with MPLS-TE; it does not support a complete RSVP-TE solution.

show ip ospf [vrf <NAME|all>] database (opaque-link|opaque-area|opaque-external)

show ip ospf [vrf <NAME|all>] database (opaque-link|opaque-area|opaque-external) LINK-STATE-ID

show ip ospf [vrf <NAME|all>] database (opaque-link|opaque-area|opaque-external) LINK-STATE-ID adv-rout

show ip ospf [vrf <NAME|all>] database (opaque-link|opaque-area|opaque-external) adv-router ADV-ROUTER

show ip ospf [vrf <NAME|all>] database (opaque-link|opaque-area|opaque-external) LINK-STATE-ID self-orig

show ip ospf [vrf <NAME|all>] database (opaque-link|opaque-area|opaque-external) self-originate
Show Opaque LSA from the database.

show ip ospf (1-65535) reachable-routers

show ip ospf [vrf <NAME|all>] reachable-routers
Show routing table of reachable routers.

3.11.7 Traffic Engineering

Note: At this time, FRR offers partial support for some of the routing protocol extensions that can be used with MPLS-TE. FRR does not support a complete RSVP-TE solution currently.

mpls-te on
Enable Traffic Engineering LSA flooding.

mpls-te router-address <A.B.C.D>
Configure stable IP address for MPLS-TE. This IP address is then advertise in Opaque LSA Type-10 TLV=1 (TE) option 1 (Router-Address).

mpls-te inter-as area <area-id>|as
Enable **RFC 5392** support - Inter-AS TE v2 - to flood Traffic Engineering parameters of Inter-AS link. 2 modes are supported: AREA and AS; LSA are flood in AREA <area-id> with Opaque Type-10, respectively in AS with Opaque Type-11. In all case, Opaque-LSA TLV=6.

mpls-te export
Export Traffic Engineering Data Base to other daemons through the ZAPI Opaque Link State messages.

show ip ospf mpls-te interface

show ip ospf mpls-te interface INTERFACE
Show MPLS Traffic Engineering parameters for all or specified interface.

show ip ospf mpls-te router
Show Traffic Engineering router parameters.

show ip ospf mpls-te database [verbose|json]

show ip ospf mpls-te database vertex [self-originate|adv-router ADV-ROUTER] [verbose|json]

show ip ospf mpls-te database edge [A.B.C.D] [verbose|json]

show ip ospf mpls-te database subnet [A.B.C.D/M] [verbose|json]
Show Traffic Engineering Database

3.11.8 Router Information

router-info [as | area]
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>

pce domain as (0-65535)

pce neighbor as (0-65535)

pce flag BITPATTERN

pce scope BITPATTERN
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.

3.11.9 Segment Routing

This is an EXPERIMENTAL support of Segment Routing as per *RFC 8665* for MPLS dataplane.

segment-routing on
Enable Segment Routing. Even if this also activate routing information support, it is preferable to also activate routing information, and set accordingly the Area or AS flooding.

segment-routing global-block (16-1048575) (16-1048575) [local-block (16-1048575) (16-1048575)]
Set the Segment Routing Global Block i.e. the label range used by MPLS to store label in the MPLS FIB for Prefix SID. Optionally also set the Local Block, i.e. the label range used for Adjacency SID. The negative version of the command always unsets both ranges.

segment-routing node-msd (1-16)
Fix the Maximum Stack Depth supported by the router. The value depend of the MPLS dataplane. E.g. for Linux kernel, since version 4.13 it is 32.

segment-routing prefix A.B.C.D/M [index (0-65535)|no-php-flag|explicit-null]
prefix with /32 corresponding to a loopback interface are currently supported. The 'no-php-flag' means NO Penultimate Hop Popping that allows SR node to request to its neighbor to not pop the label. The 'explicit-null' means that neighbor nodes must swap the incoming label by the MPLS Explicit Null label before delivering the packet.

show ip ospf database segment-routing <adv-router ADVROUTER|self-originate> [json]
Show Segment Routing Data Base, all SR nodes, specific advertised router or self router. Optional JSON output can be obtained by appending 'json' to the end of the command.

3.11.10 External Route Summarisation

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

summary-address A.B.C.D/M [tag (1-4294967295)]
This command enable/disables summarisation for the configured address range. Tag is the optional parameter. If tag configured Summary route will be originated with the configured tag.

summary-address A.B.C.D/M no-advertise
This command to ensure not advertise the summary lsa for the matched external LSAs.

aggregation timer (5-1800)
Configure aggregation delay timer interval. Summarisation starts only after this delay timer expiry. By default, delay interval is 5 seconds.

The no form of the command resets the aggregation delay interval to default value.

show ip ospf [vrf <NAME|all>] summary-address [detail] [json]
Show configuration for display all configured summary routes with matching external LSA information.

3.11.11 TI-LFA

Experimental support for Topology Independent LFA (Loop-Free Alternate), see for example 'draft-bashandy-rtgwg-segment-routing-ti-lfa-05'. Note that TI-LFA requires a proper Segment Routing configuration.

fast-reroute ti-lfa [node-protection]
Configured on the router level. Activates TI-LFA for all interfaces.

Note that so far only P2P interfaces are supported.

3.11.12 Debugging OSPF

debug ospf [(1-65535)] bfd
Enable or disable debugging for BFD events. This will show BFD integration library messages and OSPF BFD integration messages that are mostly state transitions and validation problems.

debug ospf [(1-65535)] client-api
Show debug information for the OSPF opaque data client API.

debug ospf [(1-65535)] default-information
Show debug information of default information

debug ospf [(1-65535)] packet (hello|dd|ls-request|ls-update|ls-ack|all) (send|recv) [detail]
Dump Packet for debugging

debug ospf [(1-65535)] ism [status|events|timers]
Show debug information of Interface State Machine

debug ospf [(1-65535)] nsm [status|events|timers]
Show debug information of Network State Machine

debug ospf [(1-65535)] event
Show debug information of OSPF event

debug ospf [(1-65535)] nssa
Show debug information about Not So Stub Area

debug ospf [(1-65535)] ldp-sync
Show debug information about LDP-Sync

debug ospf [(1-65535)] lsa [aggregate|flooding|generate|install|refresh]
Show debug detail of Link State messages

debug ospf [(1-65535)] sr
Show debug information about Segment Routing

debug ospf [(1-65535)] te
Show debug information about Traffic Engineering LSA

debug ospf [(1-65535)] ti-lfa
Show debug information about SR TI-LFA

debug ospf [(1-65535)] zebra [interface|redistribute]
Show debug information of ZEBRA API

debug ospf [(1-65535)] graceful-restart
Enable/disable debug information for OSPF Graceful Restart Helper

show debugging ospf

3.11.13 Sample Configuration

A simple example, with MD5 authentication enabled:

```
!
interface bge0
 ip ospf authentication message-digest
 ip ospf message-digest-key 1 md5 ABCDEFGHIJK
!
router ospf
 network 192.168.0.0/16 area 0.0.0.1
 area 0.0.0.1 authentication message-digest
```

An ABR router, with MD5 authentication and performing summarisation of networks between the areas:

```
!
password ABCDEF
log file /var/log/frr/ospfd.log
service advanced-vty
!
interface eth0
 ip ospf authentication message-digest
 ip ospf message-digest-key 1 md5 ABCDEFGHIJK
!
interface ppp0
 ip ospf passive
!
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
 network 192.168.0.0/24 area 0.0.0.0
 network 10.0.0.0/16 area 0.0.0.0
```

(continues on next page)

(continued from previous page)

```
 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
!
```

A Traffic Engineering configuration, with Inter-ASv2 support.

First, the `zebra.conf` part:

```
interface eth0
 ip address 198.168.1.1/24
 link-params
  enable
  admin-grp 0xa1
  metric 100
  max-bw 1.25e+07
  max-rsv-bw 1.25e+06
  unrsv-bw 0 1.25e+06
  unrsv-bw 1 1.25e+06
  unrsv-bw 2 1.25e+06
  unrsv-bw 3 1.25e+06
  unrsv-bw 4 1.25e+06
  unrsv-bw 5 1.25e+06
  unrsv-bw 6 1.25e+06
  unrsv-bw 7 1.25e+06
!
interface eth1
 ip address 192.168.2.1/24
 link-params
  enable
  metric 10
  max-bw 1.25e+07
  max-rsv-bw 1.25e+06
  unrsv-bw 0 1.25e+06
  unrsv-bw 1 1.25e+06
  unrsv-bw 2 1.25e+06
  unrsv-bw 3 1.25e+06
  unrsv-bw 4 1.25e+06
  unrsv-bw 5 1.25e+06
  unrsv-bw 6 1.25e+06
  unrsv-bw 7 1.25e+06
  neighbor 192.168.2.2 as 65000
   hostname HOSTNAME
   password PASSWORD
   log file /var/log/zebra.log
   !
   interface eth0
    ip address 198.168.1.1/24
    link-params
     enable
```

(continues on next page)

(continued from previous page)

```
    admin-grp 0xa1
    metric 100
    max-bw 1.25e+07
    max-rsv-bw 1.25e+06
    unrsv-bw 0 1.25e+06
    unrsv-bw 1 1.25e+06
    unrsv-bw 2 1.25e+06
    unrsv-bw 3 1.25e+06
    unrsv-bw 4 1.25e+06
    unrsv-bw 5 1.25e+06
    unrsv-bw 6 1.25e+06
    unrsv-bw 7 1.25e+06
  !
  interface eth1
   ip address 192.168.2.1/24
   link-params
    enable
    metric 10
    max-bw 1.25e+07
    max-rsv-bw 1.25e+06
    unrsv-bw 0 1.25e+06
    unrsv-bw 1 1.25e+06
    unrsv-bw 2 1.25e+06
    unrsv-bw 3 1.25e+06
    unrsv-bw 4 1.25e+06
    unrsv-bw 5 1.25e+06
    unrsv-bw 6 1.25e+06
    unrsv-bw 7 1.25e+06
    neighbor 192.168.2.2 as 65000
```

Then the `ospfd.conf` itself:

```
hostname HOSTNAME
password PASSWORD
log file /var/log/ospfd.log
!
!
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
 ospf opaque-lsa
 mpls-te
 mpls-te router-address 192.168.1.1
```

(continues on next page)

(continued from previous page)

```
 mpls-te inter-as 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
 capability opaque
 mpls-te
 mpls-te router-address 192.168.1.1
 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
!
```

3.12 OSPFv3

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

3.12.1 OSPF6 router

router ospf6 [vrf NAME]

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

timers throttle spf (0-600000) (0-600000) (0-600000)
This command sets the initial *delay*, the *initial-holdtime* and the *maximum-holdtime* between when SPF is calculated and the event which triggered the calculation. The times are specified in milliseconds and must be in the range of 0 to 600000 milliseconds.

The *delay* specifies the minimum amount of time to delay SPF calculation (hence it affects how long SPF calculation is delayed after an event which occurs outside of the holdtime of any previous SPF calculation, and also serves as a minimum holdtime).

Consecutive SPF calculations will always be separated by at least 'hold-time' milliseconds. The hold-time is adaptive and initially is set to the *initial-holdtime* configured with the above command. Events which occur within the holdtime of the previous SPF calculation will cause the holdtime to be increased by *initial-holdtime*, bounded by the *maximum-holdtime* configured with this command. If the adaptive hold-time elapses without any SPF-triggering event occurring then the current holdtime is reset to the *initial-holdtime*.

```
router ospf6
 timers throttle spf 200 400 10000
```

In this example, the *delay* is set to 200ms, the initial holdtime is set to 400ms and the *maximum holdtime* to 10s. Hence there will always be at least 200ms between an event which requires SPF calculation and the actual SPF calculation. Further consecutive SPF calculations will always be separated by between 400ms to 10s, the hold-time increasing by 400ms each time an SPF-triggering event occurs within the hold-time of the previous SPF calculation.

auto-cost reference-bandwidth COST
This sets the reference bandwidth for cost calculations, where this bandwidth is considered equivalent to an OSPF cost of 1, specified in Mbits/s. The default is 100Mbit/s (i.e. a link of bandwidth 100Mbit/s or higher will have a cost of 1. Cost of lower bandwidth links will be scaled with reference to this cost).

This configuration setting MUST be consistent across all routers within the OSPF domain.

maximum-paths (1-64)
Use this command to control the maximum number of parallel routes that OSPFv3 can support. The default is 64.

write-multiplier (1-100)
Use this command to tune the amount of work done in the packet read and write threads before relinquishing control. The parameter is the number of packets to process before returning. The default value of this parameter is 20.

clear ipv6 ospf6 process [vrf NAME]
This command clears up the database and routing tables and resets the neighborship by restarting the interface state machine. This will be helpful when there is a change in router-id and if user wants the router-id change to take effect, user can use this cli instead of restarting the ospf6d daemon.

clear ipv6 ospf6 [vrf NAME] interface [IFNAME]
This command restarts the interface state machine for all interfaces in the VRF or only for the specific interface if `IFNAME` is specified.

3.12.2 ASBR Summarisation Support in OSPFv3

External routes in OSPFv3 are carried by type 5/7 LSA (external LSAs). External LSAs are generated by ASBR (Autonomous System Boundary Router). Large topology database requires a large amount of router memory, which slows down all processes, including SPF calculations. It is necessary to reduce the size of the OSPFv3 topology database, especially in a large network. Summarising routes keeps the routing tables smaller and easier to troubleshoot.

External route summarization must be configured on ASBR. Stub area do not allow ASBR because they don't allow type 5 LSAs.

An ASBR will inject a summary route into the OSPFv3 domain.

Summary route will only be advertised if you have at least one subnet that falls within the summary range.

Users will be allowed an option in the CLI to not advertise range of ipv6 prefixes as well.

The configuration of ASBR Summarisation is supported using the CLI command

summary-address X:X::X:X/
M [tag (1-4294967295)] [{metric (0-16777215) | metric-type (1-2)}]
This command will advertise a single External LSA on behalf of all the prefixes falling under this range configured by the CLI. The user is allowed to configure tag, metric and metric-type as well. By default, tag is not configured, default metric as 20 and metric-type as type-2 gets advertised. A summary route is created when one or more specific routes are learned and removed when no more specific route exist. The summary route is also installed in the local system with Null0 as next-hop to avoid leaking traffic.

no summary-address X:X::X:X/
M [tag (1-4294967295)] [{metric (0-16777215) | metric-type (1-2)}]
This command can be used to remove the summarisation configuration. This will flush the single External LSA if it was originated and advertise the External LSAs for all the existing individual prefixes.

summary-address X:X::X:X/M no-advertise
This command can be used when user do not want to advertise a certain range of prefixes using the no-advertise option. This command when configured will flush all the existing external LSAs falling under this range.

no summary-address X:X::X:X/M no-advertise
This command can be used to remove the previous configuration. When configured, tt will resume originating external LSAs for all the prefixes falling under the configured range.

aggregation timer (5-1800)
The summarisation command takes effect after the aggregation timer expires. By default the value of this timer is 5 seconds. User can modify the time after which the external LSAs should get originated using this command.

no aggregation timer (5-1800)
This command removes the timer configuration. It reverts back to default 5 second timer.

show ipv6 ospf6 summary-address [detail] [json]
This command can be used to see all the summary-address related information. When detail option is used, it shows all the prefixes falling under each summary-configuration apart from other information.

3.12.3 OSPF6 area

area A.B.C.D range X:X::X:X/M [<advertise|not-advertise|cost (0-16777215)>]

area (0-4294967295) range X:X::X:X/M [<advertise|not-advertise|cost (0-16777215)>]
Summarize a group of internal subnets into a single Inter-Area-Prefix LSA. This command can only be used at the area boundary (ABR router).

By default, the metric of the summary route is calculated as the highest metric among the summarized routes. The *cost* option, however, can be used to set an explicit metric.

The *not-advertise* option, when present, prevents the summary route from being advertised, effectively filtering the summarized routes.

area A.B.C.
D nssa [no-summary] [default-information-originate [metric-type (1-2)] [metric (0-16777214)]]

area (0-4294967295) nssa [no-summary] [default-information-originate [metric-type (1-2)] [metric (0-167
Configure the area to be a NSSA (Not-So-Stubby Area).

The following functionalities are implemented as per RFC 3101:

1. Advertising Type-7 LSA into NSSA area when external route is redistributed into OSPFv3.
2. Processing Type-7 LSA received from neighbor and installing route in the route table.
3. Support for NSSA ABR functionality which is generating Type-5 LSA when backbone area is configured. Currently translation of Type-7 LSA to Type-5 LSA is enabled by default.
4. Support for NSSA Translator functionality when there are multiple NSSA ABR in an area.

An NSSA ABR can be configured with the *no-summary* option to prevent the advertisement of summaries into the area. In that case, a single Type-3 LSA containing a default route is originated into the NSSA.

NSSA ABRs and ASBRs can be configured with *default-information-originate* option to originate a Type-7 default route into the NSSA area. In the case of NSSA ASBRs, the origination of the default route is conditioned to the existence of a default route in the RIB that wasn't learned via the OSPF protocol.

area A.B.C.D nssa range X:X::X:X/M [<not-advertise|cost (0-16777215)>]

area (0-4294967295) nssa range X:X::X:X/M [<not-advertise|cost (0-16777215)>]
Summarize a group of external subnets into a single Type-7 LSA, which is then translated to a Type-5 LSA and avertised to the backbone. This command can only be used at the area boundary (NSSA ABR router).

By default, the metric of the summary route is calculated as the highest metric among the summarized routes. The *cost* option, however, can be used to set an explicit metric.

The *not-advertise* option, when present, prevents the summary route from being advertised, effectively filtering the summarized routes.

area A.B.C.D export-list NAME

area (0-4294967295) export-list NAME
Filter Type-3 summary-LSAs announced to other areas originated from intra- area paths from specified area.

```
router ospf6
 area 0.0.0.10 export-list foo
!
ipv6 access-list foo permit 2001:db8:1000::/64
ipv6 access-list foo deny any
```

With example above any intra-area paths from area 0.0.0.10 and from range 2001:db8::/32 (for example 2001:db8:1::/64 and 2001:db8:2::/64) are announced into other areas as Type-3 summary-LSA's, but any others (for example 2001:200::/48) aren't.

This command is only relevant if the router is an ABR for the specified area.

area A.B.C.D import-list NAME

area (0-4294967295) import-list NAME
Same as export-list, but it applies to paths announced into specified area as Type-3 summary-LSAs.

area A.B.C.D filter-list prefix NAME in

area A.B.C.D filter-list prefix NAME out

area (0-4294967295) filter-list prefix NAME in

area (0-4294967295) filter-list prefix NAME out
Filtering Type-3 summary-LSAs to/from area using prefix lists. This command makes sense in ABR only.

3.12.4 OSPF6 interface

ipv6 ospf6 area <A.B.C.D|(0-4294967295)>
Enable OSPFv3 on the interface and add it to the specified area.

ipv6 ospf6 cost COST
Sets interface's output cost. Default value depends on the interface bandwidth and on the auto-cost reference bandwidth.

ipv6 ospf6 hello-interval HELLOINTERVAL
Sets interface's Hello Interval. Default 10

ipv6 ospf6 dead-interval DEADINTERVAL
Sets interface's Router Dead Interval. Default value is 40.

ipv6 ospf6 graceful-restart hello-delay HELLODELAYINTERVAL
Set the length of time during which Grace-LSAs are sent at 1-second intervals while coming back up after an unplanned outage. During this time, no hello packets are sent.

A higher hello delay will increase the chance that all neighbors are notified about the ongoing graceful restart before receiving a hello packet (which is crucial for the graceful restart to succeed). The hello delay shouldn't be set too high, however, otherwise the adjacencies might time out. As a best practice, it's recommended to set the hello delay and hello interval with the same values. The default value is 10 seconds.

ipv6 ospf6 retransmit-interval RETRANSMITINTERVAL
Sets interface's Rxmt Interval. Default value is 5.

ipv6 ospf6 priority PRIORITY
Sets interface's Router Priority. Default value is 1.

ipv6 ospf6 transmit-delay TRANSMITDELAY
Sets interface's Inf-Trans-Delay. Default value is 1.

ipv6 ospf6 network (broadcast|point-to-point)
Set explicitly network type for specified interface.

3.12.5 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.

3.12.6 Redistribute routes to OSPF6

redistribute <babel|bgp|connected|isis|kernel|openfabric|ripng|sharp|static|table> [metric-type (1-2)]
Redistribute routes of the specified protocol or kind into OSPFv3, with the metric type and metric set if specified, filtering the routes using the given route-map if specified.

default-information originate [{always|metric (0-16777214)|metric-type (1-2)|route-map WORD}]
The command injects default route in the connected areas. The always argument injects the default route regardless of it being present in the router. Metric values and route-map can also be specified optionally.

3.12.7 Graceful Restart

graceful-restart [grace-period (1-1800)]
Configure Graceful Restart (RFC 5187) restarting support. When enabled, the default grace period is 120 seconds.

To perform a graceful shutdown, the "graceful-restart prepare ipv6 ospf" EXEC-level command needs to be issued before restarting the ospf6d daemon.

When Graceful Restart is enabled and the ospf6d daemon crashes or is killed abruptely (e.g. SIGKILL), it will attempt an unplanned Graceful Restart once it restarts.

graceful-restart helper enable [A.B.C.D]
Configure Graceful Restart (RFC 5187) helper support. By default, helper support is disabled for all neighbours. This config enables/disables helper support on this router for all neighbours. To enable/disable helper support for a specific neighbour, the router-id (A.B.C.D) has to be specified.

graceful-restart helper strict-lsa-checking
If 'strict-lsa-checking' is configured then the helper will abort the Graceful Restart when a LSA change occurs which affects the restarting router. By default 'strict-lsa-checking' is enabled"

graceful-restart helper supported-grace-time (10-1800)
Supports as HELPER for configured grace period.

graceful-restart helper planned-only
It helps to support as HELPER only for planned restarts. By default, it supports both planned and unplanned outages.

graceful-restart prepare ipv6 ospf
Initiate a graceful restart for all OSPFv3 instances configured with the "graceful-restart" command. The ospf6d daemon should be restarted during the instance-specific grace period, otherwise the graceful restart will fail.

This is an EXEC-level command.

3.12.8 Authentication trailer support:

IPv4 version of OSPF supports authentication as part of the base RFC. When IPv6 version of OSPF was developed there was IPSec support for IPv6, Hence OSPFv3(IPv6 version of OSPF) suggest to use IPSec as authentication and encryption mechanism. IPSec supports authentication using AH header and Encryption using ESP.

There are few disadvantages of using IPSec with OSPFv3.

1. If encryption is enabled for OSPFv3 packets, then its not possible to give priority to control packets.
2. IPSec has platform dependency and may not be supported in all platforms.
3. It is performance intensive.
4. Its difficult to configure.

Some advantages of OSPFv3 authentication trailer feature.

1. It provides replay protection via sequence number.
2. It provides IPv6 source address protection.
3. No platform dependency.
4. Easy to implement and maintain.

This feature is support for RFC7166.

FRR supports MD5 and SHA256 internally and relays on openssl for other hash algorithms. If user wants to use only MD5 and SHA256, no special action is required. If user wants complete support of authentication trailer with all hash algorithms follow below steps.

Installing Dependencies:

```
sudo apt update
sudo apt-get install openssl
```

Compile:

Follow normal compilation as mentioned in the build page. If you want to use all the hash algorithms then follow the steps mentioned in note before compiling.

Note: If your platform supports `openssl`, please make sure to add `--with-crypto=openssl` to your configure options. Default value is `--with-crypto=internal`

CLI Configuration:

There are two ways in which authentication trailer can be configured for OSPFv3. These commands are mutually exclusive, only one can be configured at any time.

1. Using manual key configuration.
2. Using keychain.

List of hash algorithms supported:

Without openssl:

MD5 HMAC-SHA-256

With openssl:

MD5 HMAC-SHA-1 HMAC-SHA-256 HMAC-SHA-384 HMAC-SHA-512

Example configuration of manual key:

Without openssl:

ipv6 ospf6 authentication key-id (1-65535) hash-algo <md5|hmac-sha-256> key WORD

With openssl:

ipv6 ospf6 authentication key-id (1-65535) hash-algo <md5|hmac-sha-256|hmac-sha-1|hmac-sha-384|hmac-sha

Example configuration of keychain:

ipv6 ospf6 authentication keychain KEYCHAIN_NAME

Running configuration:

Manual key:

```
frr# show running-config
Building configuration...

Current configuration:
!
interface ens192
 ipv6 address 2001:DB8::2/64
 ipv6 ospf6 authentication key-id 10 hash-algo hmac-sha-256 key abhinay
```

Keychain:

```
frr# show running-config
Building configuration...

Current configuration:
!
interface ens192
 ipv6 address 2001:DB8::2/64
 ipv6 ospf6 authentication keychain abhinay
```

Example keychain config:

```
frr#show running-config
Building configuration...

Current configuration:
!
 key chain abcd
  key 100
   key-string password
   cryptographic-algorithm sha1
  exit
  key 200
   key-string password
   cryptographic-algorithm sha256
  exit
 !
 key chain pqr
  key 300
   key-string password
   cryptographic-algorithm sha384
  exit
  key 400
   key-string password
   cryptographic-algorithm sha384
```

(continues on next page)

(continued from previous page)

```
 exit
!
```

Show commands:

There is an interface show command that displays if authentication trailer is enabled or not. json output is also supported.

There is support for drop counters, which will help in debugging the feature.

```
frr# show ipv6 ospf6 interface ens192
ens192 is up, type BROADCAST
  Interface ID: 5
  Number of I/F scoped LSAs is 2
    0 Pending LSAs for LSUpdate in Time 00:00:00 [thread off]
    0 Pending LSAs for LSAck in Time 00:00:00 [thread off]
  Authentication trailer is enabled with manual key         ==> new info added
    Packet drop Tx 0, Packet drop Rx 0
```

OSPFv3 supports options in hello and database description packets hence the presence of authentication trailer needs to be stored in OSPFv3 neighbor info. Since RFC specifies that we need to handled sequence number for every ospf6 packet type, sequence number recvd in authentication header from the neighbor is stored in neighbor to validate the packet. json output is also supported.

```
frr# show ipv6 ospf6 neighbor 2.2.2.2 detail
 Neighbor 2.2.2.2%ens192
    Area 1 via interface ens192 (ifindex 3)
    0 Pending LSAs for LSUpdate in Time 00:00:00 [thread off]
    0 Pending LSAs for LSAck in Time 00:00:00 [thread off]
    Authentication header present                           ==> new info added
                         hello        DBDesc       LSReq        LSUpd        LSAck
      Higher sequence no 0x0          0x0          0x0          0x0          0x0
      Lower sequence no  0x242E       0x1DC4       0x1DC3       0x23CC       0x1DDA
```

Sent packet sequence number is maintained per ospf6 router for every packet that is sent out of router, so sequence number is maintained per ospf6 process.

```
frr# show ipv6 ospf6
 OSPFv3 Routing Process (0) with Router-ID 2.2.2.2
 Number of areas in this router is 1
 Authentication Sequence number info
  Higher sequence no 3, Lower sequence no 1656
```

Debug command:

Below command can be used to enable ospfv3 authentication trailer specific logs if you have to debug the feature.

debug ospf6 authentication [<tx|rx>]

Feature supports authentication trailer tx/rx drop counters for debugging, which can be used to see if packets are getting dropped due to error in processing authentication trailer information in OSPFv3 packet. json output is also supported.

```
frr# show ipv6 ospf6 interface ens192
ens192 is up, type BROADCAST
  Interface ID: 5
  Number of I/F scoped LSAs is 2
    0 Pending LSAs for LSUpdate in Time 00:00:00 [thread off]
    0 Pending LSAs for LSAck in Time 00:00:00 [thread off]
  Authentication trailer is enabled with manual key
    Packet drop Tx 0, Packet drop Rx 0                    ==> new counters
```

Clear command:

Below command can be used to clear the tx/rx drop counters in interface. Below command can be used to clear all ospfv3 interface or specific interface by specifying the interface name.

clear ipv6 ospf6 auth-counters interface [IFNAME]

3.12.9 Showing OSPF6 information

show ipv6 ospf6 [vrf <NAME|all>] [json]
Show information on a variety of general OSPFv3 and area state and configuration information. JSON output can be obtained by appending 'json' to the end of command.

show ipv6 ospf6 [vrf <NAME|all>] database [<detail|dump|internal>] [json]
This command shows LSAs present in the LSDB. There are three view options. These options helps in viewing all the parameters of the LSAs. JSON output can be obtained by appending 'json' to the end of command. JSON option is not applicable with 'dump' option.

show ipv6 ospf6 [vrf <NAME|all>] database <router|network|inter-prefix|inter-router|as-external|group-m
These options filters out the LSA based on its type. The three views options works here as well. JSON output can be obtained by appending 'json' to the end of command.

show ipv6 ospf6 [vrf <NAME|all>] database adv-router A.B.C.D linkstate-id A.B.C.D [json]
The LSAs additinally can also be filtered with the linkstate-id and advertising-router fields. We can use the LSA type filter and views with this command as well and visa-versa. JSON output can be obtained by appending 'json' to the end of command.

show ipv6 ospf6 [vrf <NAME|all>] database self-originated [json]
This command is used to filter the LSAs which are originated by the present router. All the other filters are applicable here as well.

show ipv6 ospf6 [vrf <NAME|all>] interface [json]
To see OSPF interface configuration like costs. JSON output can be obtained by appending "json" in the end.

show ipv6 ospf6 [vrf <NAME|all>] neighbor [json]
Shows state and chosen (Backup) DR of neighbor. JSON output can be obtained by appending 'json' at the end.

show ipv6 ospf6 [vrf <NAME|all>] interface traffic [json]
Shows counts of different packets that have been received and transmitted by the interfaces. JSON output can be obtained by appending "json" at the end.

show ipv6 route ospf6
This command shows internal routing table.

show ipv6 ospf6 zebra [json]
Shows state about what is being redistributed between zebra and OSPF6. JSON output can be obtained by appending "json" at the end.

show ipv6 ospf6 [vrf <NAME|all>] redistribute [json]
Shows the routes which are redistributed by the router. JSON output can be obtained by appending 'json' at the end.

show ipv6 ospf6 [vrf <NAME|all>] route [<intra-area|inter-area|external-1|external-2|X:X::X:X|X:X::X:X/M|detail|summary>] [json]
This command displays the ospfv3 routing table as determined by the most recent SPF calculations. Options are provided to view the different types of routes. Other than the standard view there are two other options, detail and summary. JSON output can be obtained by appending 'json' to the end of command.

show ipv6 ospf6 [vrf <NAME|all>] route X:X::X:X/M match [detail] [json]
The additional match option will match the given address to the destination of the routes, and return the result accordingly.

show ipv6 ospf6 [vrf <NAME|all>] interface [IFNAME] prefix [detail|<X:X::X:X|X:X::X:X/M> [<match|detail>]] [json]
This command shows the prefixes present in the interface routing table. Interface name can also be given. JSON output can be obtained by appending 'json' to the end of command.

show ipv6 ospf6 [vrf <NAME|all>] spf tree [json]
This commands shows the spf tree from the recent spf calculation with the calling router as the root. If json is appended in the end, we can get the tree in JSON format. Each area that the router belongs to has it's own JSON object, with each router having "cost", "isLeafNode" and "children" as arguments.

show ipv6 ospf6 graceful-restart helper [detail] [json]
This command shows the graceful-restart helper details including helper configuration parameters.

3.12.10 OSPFv3 Debugging

The following debug commands are supported:

debug ospf6 abr
Toggle OSPFv3 ABR debugging messages.

debug ospf6 asbr
Toggle OSPFv3 ASBR debugging messages.

debug ospf6 border-routers {router-id [A.B.C.D] | area-id [A.B.C.D]}
Toggle OSPFv3 border router debugging messages. This can be specified for a router with specific Router-ID/Area-ID.

debug ospf6 flooding
Toggle OSPFv3 flooding debugging messages.

debug ospf6 interface
Toggle OSPFv3 interface related debugging messages.

debug ospf6 lsa
Toggle OSPFv3 Link State Advertisements debugging messages.

debug ospf6 lsa aggregation
Toggle OSPFv3 Link State Advertisements summarization debugging messages.

debug ospf6 message
Toggle OSPFv3 message exchange debugging messages.

debug ospf6 neighbor
Toggle OSPFv3 neighbor interaction debugging messages.

debug ospf6 nssa
Toggle OSPFv3 Not So Stubby Area (NSSA) debugging messages.

debug ospf6 route
Toggle OSPFv3 routes debugging messages.

debug ospf6 spf
Toggle OSPFv3 Shortest Path calculation debugging messages.

debug ospf6 zebra
Toggle OSPFv3 zebra interaction debugging messages.

debug ospf6 graceful-restart
Toggle OSPFv3 graceful-restart helper debugging messages.

3.12.11 Sample configuration

Example of ospf6d configured on one interface and area:

```
interface eth0
 ipv6 ospf6 area 0.0.0.0
 ipv6 ospf6 instance-id 0
!
router ospf6
 ospf6 router-id 212.17.55.53
 area 0.0.0.0 range 2001:770:105:2::/64
!
```

Larger example with policy and various options set:

```
debug ospf6 neighbor state
!
interface fxp0
 ipv6 ospf6 area 0.0.0.0
 ipv6 ospf6 cost 1
 ipv6 ospf6 hello-interval 10
 ipv6 ospf6 dead-interval 40
 ipv6 ospf6 retransmit-interval 5
 ipv6 ospf6 priority 0
 ipv6 ospf6 transmit-delay 1
 ipv6 ospf6 instance-id 0
!
interface lo0
 ipv6 ospf6 cost 1
 ipv6 ospf6 hello-interval 10
 ipv6 ospf6 dead-interval 40
 ipv6 ospf6 retransmit-interval 5
```

(continues on next page)

See also:

PBR Details

3.16.5 PBR Debugs

debug pbr events|map|nht|zebra
Debug pbr in pbrd daemon. You specify what types of debugs to turn on.

3.16.6 PBR Details

Under the covers a PBR map is translated into two separate constructs in the Linux kernel.

The PBR map specified creates a *ip rule ...* that is inserted into the Linux kernel that points to a table to use for forwarding once the rule matches.

The creation of a nexthop or nexthop-group is translated to a default route in a table with the nexthops specified as the nexthops for the default route.

3.16.7 Sample configuration

```
nexthop-group TEST
  nexthop 4.5.6.7
  nexthop 5.6.7.8
!
pbr-map BLUE seq 100
  match dst-ip 9.9.9.0/24
  match src-ip 10.10.10.0/24
  set nexthop-group TEST
!
int swp1
  pbr-policy BLUE
```

3.17 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.

3.17.1 Starting and Stopping ripd

The default configuration file name of *ripd*'s is `ripd.conf`. When invocation *ripd* searches directory /etc/frr. If `ripd.conf` is not there next search current directory.

RIP uses UDP port 520 to send and receive RIP packets. So the user must have the capability to bind the port, generally this means that the user must have superuser privileges. RIP protocol requires interface information maintained by *zebra* daemon. So running *zebra* is mandatory to run *ripd*. Thus minimum sequence for running RIP is like below:

```
# zebra -d
# ripd -d
```

Please note that *zebra* must be invoked before *ripd*.

To stop *ripd*. Please use:

```
kill `cat /var/run/frr/ripd.pid`
```

Certain signals have special meanings to *ripd*.

Signal	Action
SIGHUP	Reload configuration file `ripd.conf`. All configurations are reset. All routes learned so far are cleared and removed from routing table.
SIGUSR1	Rotate the *ripd* logfile.
SIGINT SIGTERM	Sweep all installed routes and gracefully terminate.

ripd invocation options. Common options that can be specified (*Common Invocation Options*).

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.

3.17.2 RIP Configuration

router rip [vrf NAME]
: The *router rip* command is necessary to enable RIP. To disable RIP, use the *no router rip* command. RIP must be enabled before carrying out any of the RIP commands.

network NETWORK
: Set the RIP enable interface by NETWORK. The interfaces which have addresses matching with NETWORK are enabled.

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

network IFNAME
Set a RIP enabled interface by IFNAME. Both the sending and receiving of RIP packets will be enabled on the port specified in the *network ifname* command. The *no network ifname* command will disable RIP on the specified interface.

neighbor A.B.C.D
Specify a RIP neighbor to send updates to. This is required when a neighbor is connected via a network that does not support multicast, or when it is desired to statically define a neighbor. RIP updates will be sent via unicast to each neighbour. Neighbour updates are in addition to any multicast updates sent when an interface is not in passive mode (see the *passive-interface* command). RIP will continue to process updates received from both the neighbor and any received via multicast. The *no neighbor a.b.c.d* command will disable the RIP neighbor.

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

```
!
router rip
 network 10.0.0.0/8
 network eth0
!
```

passive-interface (IFNAME|default)
This command sets the specified interface to passive mode. On passive mode interface, all receiving packets are processed as normal and ripd does not send either multicast or unicast RIP packets except to RIP neighbors specified with *neighbor* command. The interface may be specified as *default* to make ripd default to passive on all interfaces.

The default is to be passive on all interfaces.

ip split-horizon [poisoned-reverse]
Control split-horizon on the interface. Default is *ip split-horizon*. If you don't perform split-horizon on the interface, please specify *no ip split-horizon*.

If *poisoned-reverse* is also set, the router sends the poisoned routes with highest metric back to the sending router.

allow-ecmp [1-MULTIPATH_NUM]
Control how many ECMP paths RIP can inject for the same prefix. If specified without a number, a maximum is taken (compiled with `--enable-multipath`).

3.17.3 RIP Version Control

RIP can be configured to send either Version 1 or Version 2 packets. The default is to send RIPv2 while accepting both RIPv1 and RIPv2 (and replying with packets of the appropriate version for REQUESTS / triggered updates). The version to receive and send can be specified globally, and further overridden on a per-interface basis if needs be for send and receive separately (see below).

It is important to note that RIPv1 cannot be authenticated. Further, if RIPv1 is enabled then RIP will reply to REQUEST packets, sending the state of its RIP routing table to any remote routers that ask on demand. For a more detailed discussion on the security implications of RIPv1 see *RIP Authentication*.

version VERSION
Set RIP version to accept for reads and send. VERSION can be either `1` or `2`.

Disabling RIPv1 by specifying version 2 is STRONGLY encouraged, *RIP Authentication*. This may become the default in a future release.

Default: Send Version 2, and accept either version.

ip rip send version VERSION
VERSION can be 1, 2, or 1 2.

This interface command overrides the global rip version setting, and selects which version of RIP to send packets with, for this interface specifically. Choice of RIP Version 1, RIP Version 2, or both versions. In the latter case, where 1 2 is specified, packets will be both broadcast and multicast.

Default: Send packets according to the global version (version 2)

ip rip receive version VERSION
VERSION can be 1, 2, or 1 2.

This interface command overrides the global rip version setting, and selects which versions of RIP packets will be accepted on this interface. Choice of RIP Version 1, RIP Version 2, or both.

Default: Accept packets according to the global setting (both 1 and 2).

3.17.4 How to Announce RIP route

redistribute <babel|bgp|connected|eigrp|isis|kernel|openfabric|ospf|sharp|static|table> [metric (0-16)]
Redistribute routes from other sources into RIP.

If you want to specify RIP only static routes:

default-information originate

route A.B.C.D/M
This command is specific to FRR. The *route* command makes a static route only inside RIP. This command should be used only by advanced users who are particularly knowledgeable about the RIP protocol. In most cases, we recommend creating a static route in FRR and redistributing it in RIP using *redistribute static*.

3.17.5 Filtering RIP Routes

RIP routes can be filtered by a distribute-list.

distribute-list [prefix] LIST <in|out> IFNAME
You can apply access lists to the interface with a *distribute-list* command. If prefix is specified LIST is a prefix-list. If prefix is not specified then LIST is the access list name. *in* specifies packets being received, and *out* specifies outgoing packets. Finally if an interface is specified it will be applied against a specific interface.

The *distribute-list* command can be used to filter the RIP path. *distribute-list* can apply access-lists to a chosen interface. First, one should specify the access-list. Next, the name of the access-list is used in the distribute-list command. For example, in the following configuration `eth0` will permit only the paths that match the route 10.0.0.0/8

```
!
router rip
 distribute-list private in eth0
!
access-list private permit 10 10.0.0.0/8
access-list private deny any
!
```

distribute-list can be applied to both incoming and outgoing data.

3.17.6 RIP Metric Manipulation

RIP metric is a value for distance for the network. Usually *ripd* increment the metric when the network information is received. Redistributed routes' metric is set to 1.

default-metric (1-16)
This command modifies the default metric value for redistributed routes. The default value is 1. This command does not affect connected route even if it is redistributed by *redistribute connected*. To modify connected route's metric value, please use `redistribute connected metric` or *route-map*. *offset-list* also affects connected routes.

offset-list ACCESS-LIST (in|out)

offset-list ACCESS-LIST (in|out) IFNAME

3.17.7 RIP distance

Distance value is used in zebra daemon. Default RIP distance is 120.

distance (1-255)
Set default RIP distance to specified value.

distance (1-255) A.B.C.D/M
Set default RIP distance to specified value when the route's source IP address matches the specified prefix.

distance (1-255) A.B.C.D/M ACCESS-LIST
Set default RIP distance to specified value when the route's source IP address matches the specified prefix and the specified access-list.

3.17.8 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.

3.17.9 RIP Authentication

RIPv2 allows packets to be authenticated via either an insecure plain text password, included with the packet, or via a more secure MD5 based HMAC (keyed-Hashing for Message AuthentiCation), RIPv1 can not be authenticated at all, thus when authentication is configured *ripd* will discard routing updates received via RIPv1 packets.

However, unless RIPv1 reception is disabled entirely, *RIP Version Control*, RIPv1 REQUEST packets which are received, which query the router for routing information, will still be honoured by *ripd*, and *ripd* WILL reply to such packets. This allows *ripd* to honour such REQUESTs (which sometimes is used by old equipment and very simple devices to bootstrap their default route), while still providing security for route updates which are received.

In short: Enabling authentication prevents routes being updated by unauthenticated remote routers, but still can allow routes (I.e. the entire RIP routing table) to be queried remotely, potentially by anyone on the internet, via RIPv1.

To prevent such unauthenticated querying of routes disable RIPv1, *RIP Version Control*.

ip rip authentication mode md5
Set the interface with RIPv2 MD5 authentication.

ip rip authentication mode text
Set the interface with RIPv2 simple password authentication.

ip rip authentication string STRING
RIP version 2 has simple text authentication. This command sets authentication string. The string must be shorter than 16 characters.

ip rip authentication key-chain KEY-CHAIN
Specify Keyed MD5 chain.

```
!
key chain test
 key 1
  key-string test
!
interface eth1
 ip rip authentication mode md5
 ip rip authentication key-chain test
!
```

3.17.10 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.

3.17.11 Show RIP Information

To display RIP routes.

show ip rip [vrf NAME]
Show RIP routes.

The command displays all RIP routes. For routes that are received through RIP, this command will display the time the packet was sent and the tag information. This command will also display this information for routes redistributed into RIP.

show ip rip [vrf NAME] status
The command displays current RIP status. It includes RIP timer, filtering, version, RIP enabled interface and RIP peer information.

```
ripd> **show ip rip status**
Routing Protocol is "rip"
  Sending updates every 30 seconds with +/-50%, next due in 35 seconds
  Timeout after 180 seconds, garbage collect after 120 seconds
  Outgoing update filter list for all interface is not set
  Incoming update filter list for all interface is not set
  Default redistribution metric is 1
  Redistributing: kernel connected
  Default version control: send version 2, receive version 2
    Interface  Send  Recv
  Routing for Networks:
    eth0
    eth1
    1.1.1.1
    203.181.89.241
  Routing Information Sources:
    Gateway    BadPackets BadRoutes  Distance Last Update
```

3.17.12 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.

3.17.13 Sample configuration

```
debug rip events
debug rip packet

router rip
 network 11.0.0.0/8
 network eth0
 route 10.0.0.0/8
 distribute-list private-only in eth0

access-list private-only permit 10.0.0.0/8
access-list private-only deny any
```

3.18 RIPng

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

3.18.1 Invoking ripngd

There are no *ripngd* specific invocation options. Common options can be specified (*Common Invocation Options*).

3.18.2 ripngd Configuration

Currently ripngd supports the following commands:

router ripng [vrf NAME]
Enable RIPng.

network NETWORK
Set RIPng enabled interface by NETWORK.

network IFNAME
Set RIPng enabled interface by IFNAME.

route NETWORK
Set RIPng static routing announcement of NETWORK.

allow-ecmp [1-MULTIPATH_NUM]
Control how many ECMP paths RIPng can inject for the same prefix. If specified without a number, a maximum is taken (compiled with `--enable-multipath`).

3.18.3 ripngd Terminal Mode Commands

show ipv6 ripng [vrf NAME] status

show debugging ripng

debug ripng events

debug ripng packet

debug ripng zebra

3.18.4 ripngd Filtering Commands

RIPng routes can be filtered by a distribute-list.

distribute-list [prefix] LIST <in|out> IFNAME
You can apply access lists to the interface with a *distribute-list* command. If prefix is specified LIST is a prefix-list. If prefix is not specified then LIST is the access list name. *in* specifies packets being received, and *out* specifies outgoing packets. Finally if an interface is specified it will be applied against a specific interface.

The `distribute-list` command can be used to filter the RIPNG path. `distribute-list` can apply access-lists to a chosen interface. First, one should specify the access-list. Next, the name of the access-list is used in the distribute-list command. For example, in the following configuration `eth0` will permit only the paths that match the route 10.0.0.0/8

```
!
router ripng
 distribute-list private in 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.

3.18.5 Sample configuration

```
debug ripng events
debug ripng packet

router ripng
 network sit1
 route 3ffe:506::0/32
 distribute-list local-only out sit1
```

(continues on next page)

(continued from previous page)

```
ipv6 access-list local-only permit 3ffe:506::0/32
ipv6 access-list local-only deny any
```

3.19 SHARP

SHARP (Super Happy Advanced Routing Process) is a daemon that provides miscellaneous functionality used for testing FRR and creating proof-of-concept labs.

3.19.1 Starting SHARP

Default configuration file for *sharpd* is `sharpd.conf`. The typical location of `sharpd.conf` is /etc/frr/sharpd.conf.

If the user is using integrated config, then `sharpd.conf` need not be present and the `frr.conf` is read instead.

SHARP supports all the common FRR daemon start options which are documented elsewhere.

3.19.2 Using SHARP

All sharp commands are under the enable node and preceded by the `sharp` keyword. At present, no sharp commands will be preserved in the config.

sharp install routes A.B.C.D <nexthop <E.F.G.H|X:X::X:X>|nexthop-group NAME> (1-1000000) [instance (0-255)] [repeat (2-1000)] [opaque WORD]

Install up to 1,000,000 (one million) /32 routes starting at `A.B.C.D` with specified nexthop `E.F.G.H` or `X:X::X:X`. The nexthop is a `NEXTHOP_TYPE_IPV4` or `NEXTHOP_TYPE_IPV6` and must be reachable to be installed into the kernel. Alternatively a nexthop-group NAME can be specified and used as the nexthops. The routes are installed into zebra as `ZEBRA_ROUTE_SHARP` and can be used as part of a normal route redistribution. Route installation time is noted in the debug log. When zebra successfully installs a route into the kernel and SHARP receives success notifications for all routes this is logged as well. Instance (0-255) if specified causes the routes to be installed in a different instance. If repeat is used then we will install/uninstall the routes the number of times specified. If the keyword opaque is specified then the next word is sent down to zebra as part of the route installation.

sharp remove routes A.B.C.D (1-1000000)

Remove up to 1,000,000 (one million) /32 routes starting at `A.B.C.D`. The routes are removed from zebra. Route deletion start is noted in the debug log and when all routes have been successfully deleted the debug log will be updated with this information as well.

sharp data route

Allow end user doing route install and deletion to get timing information from the vty or vtysh instead of having to read the log file. This command is informational only and you should look at sharp_vty.c for explanation of the output as that it may change.

sharp label <ipv4|ipv6> vrf NAME label (0-1000000)

Install a label into the kernel that causes the specified vrf NAME table to be used for pop and forward operations when the specified label is seen.

sharp watch <nexthop <A.B.C.D|X:X::X:X>|import <A.B.C.D/M:X:X::X:X/M> [connected]

Instruct zebra to monitor and notify sharp when the specified nexthop is changed. The notification from zebra is written into the debug log. The nexthop or import choice chooses the type of nexthop we are asking zebra to watch for us. This choice affects zebra's decision on what matches. Connected tells zebra whether or not that we

3.24 WATCHFRR

WATCHFRR is a daemon that handles failed daemon processes and intelligently restarts them as needed.

3.24.1 Starting WATCHFRR

WATCHFRR is started as per normal systemd startup and typically does not require end users management.

3.24.2 WATCHFRR commands

show watchfrr
: Give status information about the state of the different daemons being watched by WATCHFRR

watchfrr ignore DAEMON
: Tell WATCHFRR to ignore a particular DAEMON if it goes unresponsive. This is particularly useful when you are a developer and need to debug a working system, without watchfrr pulling the rug out from under you.

3.25 MGMTd (Management Daemon)

The FRR Management Daemon (from now on referred to as MGMTd) is a new centralized entity representing the FRR Management Plane which can take management requests from any kind of UI/Frontend entity (e.g. CLI, Netconf, Restconf, Grpc etc.) over a new unified and common Frontend interface and can help maintain configurational data or retrieve operational data from any number of FRR managed entities/components that have been integrated with the new FRR Centralised Management Framework.

For organizing the management data to be owned by the FRR Management plane, management data is stored in YANG in compliance with a pre-defined set of YANG based schema. Data shall also be stored/retrieved in YANG format only.

The MGMTd also acts as a separate computational entity for offloading much of the management related computational overload involved in maintaining of management data and processing of management requests, from individual component daemons (which can otherwise be a signficant burden on the individual components, affecting performance of its other functionalities).

Lastly, the MGMTd works in-tandem with one (or more) MGMT Frontend Clients and a bunch of MGMT Backend Clients to realize the entirety of the FRR Management plane. Some of the advanatages of this new framework are:

1. Consolidation and management of all Management data by a single entity.
2. Better control over configuration validation, commit and rollback.
3. Faster collection of configuration data (without needing to involve individual component daemons).
4. Offload computational burden of YANG data parsing and validations of new configuration data being provisoned away from individual component daemons
5. Improve performance of individual component daemons while loading huge configuration or retrieving huge operational dataset.

The new FRR Management Daemon consists of the following sub-components:

- MGMT Frontend Interface
- MGMT Backend Interface
- MGMT Transaction Engine

3.25.1 MGMT Frontend Interface

The MGMT Frontend Interface is a bunch of message-based APIs that lets any UI/Frontend client to interact with the MGMT daemon to requests a set of management operations on a specific datastore/database. Following is a list of databases/datastores supported by the MGMT Frontend Interface and MGMTd:

- Candidate Database:
- Consists of configuration data items only.
- Data can be edited anytime using SET_CONFIG API.
- Data can be retrieved anytime using GET_CONFIG/GET_DATA API.
- Running Database:
- Consists of configuration data items only.
- Data cannot be edited using SET_CONFIG API.
- Data can only be modified using COMMIT_CONFIG API after which un-committed data from Candidate database will be first validated and applied to individualBackend component(s). Only on successful validation and apply on all individual components will the new data be copied over to the Running database.
- Data can be retrieved anytime using GET_CONFIG/GET_DATA API.
- Operational Database:
- Consists of non-configurational data items.
- Data is not stored on MGMT daemon. Rather it will be need to be fetched in real-time from the corresponding Backend component (if present).
- Data can be retrieved anytime using GET_DATA API.

Frontend Clients connected to MGMTd via Frontend Interface can themselves have multiple connections from one (or more) of its own remote clients. The MGMT Frontend Interface supports reresenting each of the remote clients for a given Frontend client(e.g. Netconf clients on a single Netconf server) as individual Frontend Client Sessions. So a single connection from a single Frontend Client can create more than one Frontend Client sessions.

Following are some of the management operations supported:

- INIT_SESSION/CLOSE_SESSION: Create/Destroy a session. Rest of all the operations are supported only in the context of a specific session.
- LOCK_DB/UNLOCK_DB: Lock/Unlock Management datastores/databases.
- GET_CONFIG/GET_DATA: Retrieve configurational/operational data from a specific datastore/database.
- SET_CONFIG/DELETE_CONFIG: Add/Modify/Delete specific data in a specific datastore/database.
- COMMIT_CONFIG: Validate and/or apply the uncommited set of configurations from one configuration database to another.
- Currently committing configurations from Candidate to Running database is only allowed, and not vice versa.

The exact set of message-based APIs are represented as Google Protobuf messages and can be found in the following file distributed with FRR codebase.

```
lib/mgmt.proto
```

The MGMT daemon implements a MGMT Frontend Server that opens a UNIX socket-based IPC channel on the following path to listen for incoming connections from all possible Frontend clients:

```
/var/run/frr/mgmtd_fe.sock
```

Each connection received from a Frontend client is managed and tracked as a MGMT Frontend adapter by the MGMT Frontend Adapter sub-component implemented by MGMTd.

To facilitate faster development/integration of Frontend clients with MGMT Frontend Interface, a C-based library has been developed. The API specification of this library can be found at:

```
lib/mgmt_fe_client.h
```

Following is a list of message types supported on the MGMT Frontend Interface:

- SESSION_REQ<Client-Connection-Id, Destroy>
- SESSION_REPLY<Client-Connection-Id, Destroy, Session-Id>
- LOCK_DB_REQ <Session-Id, Database-Id>
- LOCK_DB_REPLY <Session-Id, Database-Id>
- UNLOCK_DB_REQ <Session-Id, Database-Id>
- UNLOCK_DB_REPLY <Session-Id, Database-Id>
- GET_CONFIG_REQ <Session-Id, Database-Id, Base-Yang-Xpath>
- GET_CONFIG_REPLY <Session-Id, Database-Id, Base-Yang-Xpath, Yang-Data-Set>
- SET_CONFIG_REQ <Session-Id, Database-Id, Base-Yang-Xpath, Delete, . . . >
- SET_CONFIG_REPLY <Session-Id, Database-id, Base-Yang-Xpath, . . . , Status>
- COMMIT_CONFIG_REQ <Session-Id, Source-Db-Id, Dest-Db-Id>
- COMMIT_CONFIG_REPLY <Session-Id, Source-Db-id, Dest-Db-Id, Status>
- GET_DATA_REQ <Session-Id, Database-Id, Base-Yang-Xpath>
- GET_DATA_REPLY <Session-Id, Database-id, Base-Yang-Xpath, Yang-Data-Set>
- REGISTER_NOTIFY_REQ <Session-Id, Database-Id, Base-Yang-Xpath>
- DATA_NOTIFY_REQ <Database-Id, Base-Yang-Xpath, Yang-Data-Set>

Please refer to the MGMT Frontend Client Developers Reference and Guide (coming soon) for more details.

3.25.2 MGMTD Backend Interface

The MGMT Backend Interface is a bunch of message-based APIs that can be used by individual component daemons like BGPd, Staticd, Zebra to connect with MGMTd and utilize the new FRR Management Framework to let any Frontend clients to retrieve any operational data or manipulate any configuration data owned by the individual daemon component.

Like the MGMT Frontend Interface, the MGMT Backend Interface is is also comprised of the following:

- MGMT Backend Server (running on MGMT daemon)
- MGMT Backend Adapter (running on MGMT daemon)
- MGMT Backend client (running on Backend component daemons)

The MGMT Backend Client and MGMT Backend Adapter sub-component communicates using a specific set of message-based APIs.

The exact set of message-based APIs are represented as Google Protobuf messages and can be found in the following file distributed with FRR codebase.

```
lib/mgmt.proto
```

The MGMT daemon implements a MGMT Backend Server that opens a UNIX socket-based IPC channel on the following path to listen for incoming connections from all possible Backend clients:

```
/var/run/frr/mgmtd_be.sock
```

Each connection received from a Backend client is managed and tracked as a MGMT Backend adapter by the MGMT Backend Adapter sub-component implemented by MGMTd.

To facilitate faster development/integration of Backend clients with MGMTd, a C-based library has been developed. The API specification of this library can be found at:

```
lib/mgmt_be_client.h
```

Following is a list of message types supported on the MGMT Backend Interface:

- SUBSCRIBE_REQ <Req-Id, Base-Yang-Xpath, Filter-Type>
- SUBSCRIBE_REPLY <Req-Id, Status>
- TXN_REQ <Txn-Id, Create>
- TXN_REPLY <Txn-Id, Status>
- CREATE_CFGDATA_REQ <Txn-Id, Req-Id, Batch-Id, ConfigDataContents>
- CREATE_CFGDATA_ERROR <Txn-Id, Req-Id, Batch-Id, Status>
- VALIDATE_CFGDATA_REQ <Txn-Id, Batch-Id>
- VALIDATE_CFGDATA_REPLY <Txn-Id, Batch-Id, Status, ErrorInfo>
- APPLY_CFGDATA_REQ <Txn-Id, Batch-Id>
- APPLY_CFGDATA_REPLY <Txn-Id, Batch-Id, Status, ErrorInfo>
- GET_OPERDATA_REQ <Txn-Id, Base-Yang-Xpath, Filter-Type>
- GET_OPERDATA_REPLY <Txn-Id, OperDataContents>

Please refer to the MGMT Backend Client Developers Reference and Guide (coming soon) for more details.

3.25.3 MGMTD Transaction Engine

The MGMT Transaction sub-component is the main brain of the MGMT daemon that takes management requests from one (or more) Frontend Client translates them into transactions and drives them to completion in co-oridination with one (or more) Backend client daemons involved in the request.

A transaction can be seen as a set of management procedures executed over the Backend Interface with one (or more) individual Backend component daemons, as a result of some management request initiated from a specific Frontend client session. These group of operations on the Backend Interface with one (or more) individual components involved should be executed without taking any further management requests from other Frontend client sessions. To maintain this kind of atomic behavior a lock needs to be acquired (sometimes implicitly if not explicitly) by the corresponding Frontend client session, on the various datastores/databases involved in the management request being executed. The same datastores/databases need to be unlocked when all the procedures have been executed and the transaction is being closed.

Following are some of the transaction types supported by MGMT:

- Configuration Transactions
- Used to execute management operations like SET_CONFIG and COMMIT_CONFIG that involve writing/over-writing the contents of Candidate and Running databases.
- One (and only) can be created and be in-progress at any given time.
- Once initiated by a specific Frontend Client session and is still in-progress, all subsequent SET_CONFIG and COMMIT_CONFIG operations from other Frontend Client sessions will be rejected and responded with failure.
- Requires acquiring write-lock on Candidate (and later Running) databases.
- Show Transactions
- Used to execute management operations like GET_CONFIG and GET_DATA that involve only reading the contents of Candidate and Running databases (and sometimes real-time retrieval of operational data from individual component daemons).
- Multiple instance of this transaction type can be created and be in-progress at any given time.
- However, when a configuration transaction is currently in-progress show transaction can be initiated by any Frontend Client session.
- Requires acquiring read-lock on Candidate and/or Running databases.
- NOTE: Currently GET_DATA on Operational database is NOT supported. To be added in a future time soon.

3.25.4 MGMTD Configuration Rollback and Commit History

The MGMT daemon maintains upto 10 last configuration commit buffers and can rollback the contents of the Running Database to any of the commit-ids maintained in the commit buffers.

Once the number of commit buffers exceeds 10, the oldest commit buffer is deleted to make space for the latest commit. Also on rollback to a specific commit-id, buffer of all the later commits are deleted from commit record.

Configuration rollback is only allowed via VTYSH shell as of today and is not possible through the MGMT Frontend interface.

3.25.5 MGMT Configuration commands

mgmt set-config XPATH VALUE
: This command uses a SET_CONFIG request over the MGMT Frontend Interface for the specified xpath with specific value. This command is used for testing purpose only. But can be used to set configuration data from CLI using SET_CONFIG operations.

mgmt delete-config XPATH
: This command uses a SET_CONFIG request (with delete option) over the MGMT Frontend Interface o delete the YANG data node at the given xpath unless it is a key-leaf node(in which case it is not deleted).

mgmt load-config FILE <merge|replace>
: This command loads configuration in JSON format from the filepath specified, and merges or replaces the Candidate DB as per the option specified.

mgmt save-config <candidate|running> FILE
: This command dumps the DB specified in the db-name into the file in JSON format. This command in not supported for the Operational DB.

mgmt commit abort
This command will abort any configuration present on the Candidate but not been applied to the Running DB.

mgmt commit apply
This command commits any uncommited changes in the Candidate DB to the Running DB.

mgmt commit check
This command validates the configuration but does not apply them to the Running DB.

mgmt rollback commit-id WORD
This command rolls back the Running Database contents to the state corresponding to the commit-id specified.

mgmt rollback last WORD
This command rolls back the last specified number of recent commits.

3.25.6 MGMT Show commands

show mgmt backend-adapter all
This command shows the backend adapter information and the clients/daemons connected to the adapters.

show mgmt backend-yang-xpath-registry
This command shows which Backend adapters are registered for which YANG data subtree(s).

show mgmt frontend-adapter all [detail]
This command shows the frontend adapter information and the clients connected to the adapters.

show mgmt transaction all
Shows the list of transaction and bunch of information about the transaction.

show mgmt get-config [candidate|running] XPATH
This command uses the GET_CONFIG operation over the MGMT Frontend interface and returns the xpaths and values of the nodes of the subtree pointed by the <xpath>.

show mgmt get-data [candidate|operation|running] XPATH
This command uses the GET_DATA operation over the MGMT Frontend interface and returns the xpaths and values of the nodes of the subtree pointed by the <xpath>. Currenlty supported values for 'candidate' and 'running' only ('operational' shall be supported in future soon).

show mgmt database-contents [candidate|operation|running] [xpath WORD] [file WORD] json|xml
This command dumps the subtree pointed by the xpath in JSON or XML format. If filepath is not present then the tree will be printed on the shell.

show mgmt commit-history
This command dumps details of upto last 10 commits handled by MGMTd.

3.25.7 MGMT Daemon debug commands

The following debug commands enable debugging within the management daemon:

[no] debug mgmt backend
Enable[/Disable] debugging messages related to backend operations within the management daemon.

[no] debug mgmt datastore
Enable[/Disable] debugging messages related to YANG datastore operations within the management daemon.

[no] debug mgmt frontend
Enable[/Disable] debugging messages related to frontend operations within the management daemon.

[no] debug mgmt transaction
Enable[/Disable] debugging messages related to transactions within the management daemon.

3.25.8 MGMT Client debug commands

The following debug commands enable debugging within the management front and backend clients:

[no] debug mgmt client backend
Enable[/Disable] debugging messages related to backend operations inside the backend mgmtd clients.

[no] debug mgmt client frontend
Enable[/Disable] debugging messages related to frontend operations inside the frontend mgmtd clients.

RIFT

Routing in Fat Tree (RIFT) is a routing protocol with no operational expenses, designed for packet routing in CLOS-based and Fat Tree network topologies. This protocol blends link-state and distance-vector methods, offering multiple advantages for IP fabrics, including simplified management and enhanced network resilience.

riftd is based on draft-ietf-rift-rift-19.

Starting and Stopping riftd

The default configuration file name of *riftd*'s is `riftd.conf`. When invocation *riftd* searches directory /etc/frr. If `riftd.conf` is not there next search current directory.

RIFT uses UDP ports 914 and 915 to send and receive LIE/TIE packets. So the user must have the capability to bind the port, generally this means that the user must have superuser privileges. RIFT requires interface information maintained by *zebra* daemon. So running *zebra* is mandatory to run *riftd*. Thus minimum sequence for running RIFT is like below:

```
# zebra -d
# riftd -d
```

Please note that *zebra* must be invoked before *riftd*.

To stop *riftd*. Please use: kill `cat /var/run/frr/riftd.pid`

RIFT Configuration

```
router rift
```

The `router rift` command is necessary to enable RIFT. To disable RIFT, use the `no router rift` command. RIFT must be enabled before carrying out any of the RIFT commands.

```
system-id (1-4294967295)
```

Set RIFT's system ID. System ID is optional. If not specified, RIFT daemon will generate a unique one.

```
no system-id (1-4294967295)
```

Unset RIFT's system ID.

```
level <1-20|leaf|ew|tof>
```

Clos and Fat Tree networks are topologically partially ordered graphs and `level` denotes the set of nodes at the same height in such a network. `leaf` means that the node is at the last level of the network. `ew` describes an East-West link. `tof` is a node at the top of the network (maximum level number). If not specified, the level is auto-negotiated by RIFT with southbound and northbound neighbours.

```
no level <1-20|leaf|ew|tof>
```

Unset the node level value.

```
lie-address A.B.C.D
```

This command overrides the default IPv4 multicast address used to send LIE packets.

```
no lie-address A.B.C.D
```

This command removes the IPv4 multicast address configured to send LIE packets (fallbacks to default one).

```
lie-address X:X::X:X
```

This command overrides the default IPv6 multicast address used to send LIE packets.

```
no lie-address X:X::X:X
```

This command removes the IPv6 multicast address configured to send LIE packets (fallbacks to default one).

```
interface IFNAME [active-key (0-255)]
```

This command enables RIFT on the specified interface. Optionally, an active key can be passed to validate incoming packets. By default, it accepts all the packets without checking any key.

By default, RIFT is not enabled on all interfaces, you have to specify the `interface` command in order to enable it.

```
no interface IFNAME
```

This command disables RIFT on the specified interface.

Below is very simple RIFT configuration for a leaf node that uses LIE multicast address `224.0.0.150` and enables RIFT on interface `eth1`.

```
router rift
  system-id 1234
  level leaf
  lie address 224.0.0.150
  interface eth1
```

How to Announce RIFT route

```
redistribute <babel|bgp|connected|eigrp|isis|kernel|openfabric|ospf|sharp|
              static|table> [metric (0-16)] [route-map WORD]
```

Redistribute routes from other sources into RIFT.

```
no redistribute <babel|bgp|connected|eigrp|isis|kernel|openfabric|ospf|sharp|
              static|table> [metric (0-16)] [route-map WORD]
```

Remove the routes redistribution into RIFT from the specified source.

If you want to specify RIFT static prefixes:

```
prefix <A.B.C.D/M|X:X::X:X/M>
```

Specify either a IPv4 or IPv6 prefix.

```
no prefix <A.B.C.D/M|X:X::X:X/M>
```

Unset a previously specified IPv4 or IPv6 prefix.

Below is very simple RIFT configuration for a leaf node that enables RIFT on interface `eth3` and announces a IPv6 prefix `ecaf::/64`.

```
router rift
  system-id 58964
  level leaf
```

```
  interface eth3
  prefix ecaf::/64
```

RIFT route-map

Usage of *riftd*'s `route-map` support.

Optional argument `route-map MAP_NAME` can be added to each redistribute statement. You have to specify the `direction` (either `northbound` or `southbound`).

```
redistribute static [route-map MAP_NAME direction <northbound|southbound>]
redistribute connected [route-map MAP_NAME direction <northbound|southbound>]
.....
```

Route-map statement (Route Maps) is needed to use `route-map` functionality.

Show RIFT Information

To display RIFT routes.

```
show rift
```

Show RIFT routes.

```
show rift tie-db [direction <northbound|southbound>]
```

The command display the content of the Topology Information Element Database (TIE-DB). If the `direction` is provided, only shows the TIE-DB in that direction.

```
show rift disaggregation
```

The command displays all information related to automatic disaggregation (positive and negative).

Sample configuration

```
ip prefix-list DEF_ONLY_4 permit 0.0.0.0/0
ip prefix-list DEF_ONLY_4 deny any

route-map FILTER_DEFAULT_V4 deny 10
    match ip address DEF_ONLY_4

router rift
  system-id 1337
  level tof
  interface eth0
  interface eth1
  interface eth2
  redistribute connected
  redistribute static route-map FILTER_DEFAULT_V4 direction northbound
  prefix 2001::/64
  prefix 200.0.0.0/24
```