GNU Zebra

A routing software package for TCP/IP networks
Zebra version 0.93b
September 2002

Kunihiro Ishiguro

1 Overview

Zebra is a routing software package that provides TCP/IP based routing services with routing protocols support such as RIPv1, RIPv2, RIPng, OSPFv2, OSPFv3, BGP-4, and BGP-4+ (see Section 1.4 [Supported RFC], page 3). Zebra also supports special BGP Route Reflector and Route Server behavior. In addition to traditional IPv4 routing protocols, Zebra also supports IPv6 routing protocols. With SNMP daemon which supports SMUX protocol, Zebra provides routing protocol MIBs (see Chapter 15 [SNMP Support], page 75).

Zebra uses an advanced software architecture to provide you with a high quality, multi server routing engine. Zebra has an interactive user interface for each routing protocol and supports common client commands. Due to this design, you can add new protocol daemons to Zebra easily. You can use Zebra library as your program's client user interface.

Zebra is an official GNU software and distributed under the GNU General Public License.

1.1 About Zebra

Today, TCP/IP networks are covering all of the world. The Internet has been deployed in many countries, companies, and to the home. When you connect to the Internet your packet will pass many routers which have TCP/IP routing functionality.

A system with Zebra installed acts as a dedicated router. With Zebra, your machine exchanges routing information with other routers using routing protocols. Zebra uses this information to update the kernel routing table so that the right data goes to the right place. You can dynamically change the configuration and you may view routing table information from the Zebra terminal interface.

Adding to routing protocol support, Zebra can setup interface's flags, interface's address, static routes and so on. If you have a small network, or a stub network, or xDSL connection, configuring the Zebra routing software is very easy. The only thing you have to do is to set up the interfaces and put a few commands about static routes and/or default routes. If the network is rather large, or if the network structure changes frequently, you will want to take advantage of Zebra's dynamic routing protocol support for protocols such as RIP, OSPF or BGP. Zebra is with you.

Traditionally, UNIX based router configuration is done by `ifconfig` and `route` commands. Status of routing table is displayed by `netstat` utility. Almost of these commands work only if the user has root privileges. Zebra has a different system administration method. There are two user modes in Zebra. One is normal mode, the other is enable mode. Normal mode user can only view system status, enable mode user can change system configuration. This UNIX account independent feature will be great help to the router administrator.

Currently, Zebra supports common unicast routing protocols. Multicast routing protocols such as BGMP, PIM-SM, PIM-DM will be supported in Zebra 2.0. MPLS support is going on. In the future, TCP/IP filtering control, QoS control, diffserv configuration will be added to Zebra. Zebra project's final goal is making a productive, quality free TCP/IP routing software.

1.2 System Architecture

Traditional routing software is made as a one process program which provides all of the routing protocol functionalities. Zebra takes a different approach. It is made from a collection of several daemons that work together to build the routing table. There may be several protocol-specific routing daemons and zebra the kernel routing manager.

The `ripd` daemon handles the RIP protocol, while `ospfd` is a daemon which supports OSPF version 2. `bgpd` supports the BGP-4 protocol. For changing the kernel routing table and for redistribution of routes between different routing protocols, there is a kernel routing table manager `zebra` daemon. It is easy to add a new routing protocol daemons to the entire routing system without affecting any other software. You need to run only the protocol daemon associated with routing protocols in use. Thus, user may run a specific daemon and send routing reports to a central routing console.

There is no need for these daemons to be running on the same machine. You can even run several same protocol daemons on the same machine. This architecture creates new possibilities for the routing system.

```
+----+  +----+  +-----+  +-----+
|bgpd|  |ripd|  |ospfd|  |zebra|
+----+  +----+  +-----+  +-----+
                            |
+---------------------------|--+
|                           v  |
|  UNIX Kernel  routing table  |
|                              |
+------------------------------+

    Zebra System Architecture
```

Multi-process architecture brings extensibility, modularity and maintainability. At the same time it also brings many configuration files and terminal interfaces. Each daemon has it's own configuration file and terminal interface. When you configure a static route, it must be done in `zebra` configuration file. When you configure BGP network it must be done in `bgpd` configuration file. This can be a very annoying thing. To resolve the problem, Zebra provides integrated user interface shell called `vtysh`. `vtysh` connects to each daemon with UNIX domain socket and then works as a proxy for user input.

Zebra was planned to use multi-threaded mechanism when it runs with a kernel that supports multi-threads. But at the moment, the thread library which comes with GNU/Linux or FreeBSD has some problems with running reliable services such as routing software, so we don't use threads at all. Instead we use the `select(2)` system call for multiplexing the events.

When `zebra` runs under a GNU Hurd kernel it will act as a kernel routing table itself. Under GNU Hurd, all TCP/IP services are provided by user processes called `pfinet`. Zebra will provide all the routing selection mechanisms for the process. This feature will be implemented when GNU Hurd becomes stable.

1.3 Supported Platforms

Currently Zebra supports GNU/Linux, BSD and Solaris. Below is a list of OS versions on which Zebra runs. Porting Zebra to other platforms is not so too difficult. Platform dependent codes exist only in `zebra` daemon. Protocol daemons are platform independent. Please let us know when you find out Zebra runs on a platform which is not listed below.

- GNU/Linux 2.0.37
- GNU/Linux 2.2.x
- GNU/Linux 2.3.x
- FreeBSD 2.2.8
- FreeBSD 3.x
- FreeBSD 4.x
- NetBSD 1.4
- OpenBSD 2.5
- Solaris 2.6
- Solaris 7

Some IPv6 stacks are in development. Zebra supports following IPv6 stacks. For BSD, we recommend KAME IPv6 stack. Solaris IPv6 stack is not yet supported.

- Linux IPv6 stack for GNU/Linux 2.2.x and higher.
- KAME IPv6 stack for BSD.
- INRIA IPv6 stack for BSD.

1.4 Supported RFC

Below is the list of currently supported RFC's.

RFC1058	*Routing Information Protocol. C.L. Hedrick. Jun-01-1988.*
RF2082	*RIP-2 MD5 Authentication. F. Baker, R. Atkinson. January 1997.*
RFC2453	*RIP Version 2. G. Malkin. November 1998.*
RFC2080	*RIPng for IPv6. G. Malkin, R. Minnear. January 1997.*
RFC2328	*OSPF Version 2. J. Moy. April 1998.*
RFC2740	*OSPF for IPv6. R. Coltun, D. Ferguson, J. Moy. December 1999.*
RFC1771	*A Border Gateway Protocol 4 (BGP-4). Y. Rekhter & T. Li. March 1995.*
RFC1965	*Autonomous System Confederations for BGP. P. Traina. June 1996.*
RFC1997	*BGP Communities Attribute. R. Chandra, P. Traina & T. Li. August 1996.*
RFC2545	*Use of BGP-4 Multiprotocol Extensions for IPv6 Inter-Domain Routing. P. Marques, F. Dupont. March 1999.*

RFC2796	*BGP Route Reflection An alternative to full mesh IBGP. T. Bates & R. Chandrasekeran. June 1996.*
RFC2858	*Multiprotocol Extensions for BGP-4. T. Bates, Y. Rekhter, R. Chandra, D. Katz. June 2000.*
RFC2842	*Capabilities Advertisement with BGP-4. R. Chandra, J. Scudder. May 2000.*

When SNMP support is enabled, below RFC is also supported.

RFC1227	*SNMP MUX protocol and MIB. M.T. Rose. May-01-1991.*
RFC1657	*Definitions of Managed Objects for the Fourth Version of the Border Gateway Protocol (BGP-4) using SMIv2. S. Willis, J. Burruss, J. Chu, Editor. July 1994.*
RFC1724	*RIP Version 2 MIB Extension. G. Malkin & F. Baker. November 1994.*
RFC1850	*OSPF Version 2 Management Information Base. F. Baker, R. Coltun. November 1995.*

1.5 How to get Zebra

Zebra is still beta software and there is no officially released version. So currently Zebra is distributed from Zebra beta ftp site located at:

`ftp://ftp.zebra.org/pub/zebra`

Once Zebra is released you can get it from GNU FTP site and its mirror sites. We are planning Zebra-1.0 as the first released version.

Zebra's official web page is located at:

`http://www.gnu.org/software/zebra/zebra.html`.

There is a Zebra beta tester web page at:

`http://www.zebra.org/`.

You can get the latest beta software information from this page.

1.6 Mailing List

There is a mailing list for discussions about Zebra. If you have any comments or suggestions to Zebra, please send mail to `zebra@zebra.org`. New snapshot announcements, improvement notes, and patches are sent to the list.

To subscribe to the Zebra mailing list, please send a mail to `majordomo@zebra.org` with a message body that includes only:

subscribe zebra

To unsubscribe from the list, please send a mail to `majordomo@zebra.org` with a message body that includes only:

unsubscribe zebra

1.7 Bug Reports

If you think you have found a bug, please send a bug report to bug-zebra@gnu.org. When you send a bug report, please be careful about the points below.

- Please note what kind of OS you are using. If you use the IPv6 stack please note that as well.
- Please show us the results of `netstat -rn` and `ifconfig -a`. Information from zebra's VTY command `show ip route` will also be helpful.
- Please send your configuration file with the report. If you specify arguments to the configure script please note that too.

Bug reports are very important for us to improve the quality of Zebra. Zebra is still in the development stage, but please don't hesitate to send a bug report to bug-zebra@gnu.org.

2 Installation

There are three steps for installing the software: configuration, compilation, and installation.

The easiest way to get Zebra running is to issue the following commands:

```
% configure
% make
% make install
```

2.1 Configure the Software

Zebra has an excellent configure script which automatically detects most host configurations. There are several additional configure options you can use to turn off IPv6 support, to disable the compilation of specific daemons, and to enable SNMP support.

'`--enable-guile`'
: Turn on compilation of the zebra-guile interpreter. You will need the guile library to make this. zebra-guile implementation is not yet finished. So this option is only useful for zebra-guile developers.

'`--disable-ipv6`'
: Turn off IPv6 related features and daemons. Zebra configure script automatically detects IPv6 stack. But sometimes you might want to disable IPv6 support of Zebra.

'`--disable-zebra`'
: Do not build zebra daemon.

'`--disable-ripd`'
: Do not build ripd.

'`--disable-ripngd`'
: Do not build ripngd.

'`--disable-ospfd`'
: Do not build ospfd.

'`--disable-ospf6d`'
: Do not build ospf6d.

'`--disable-bgpd`'
: Do not build bgpd.

'`--disable-bgp-announce`'
: Make `bgpd` which does not make bgp announcements at all. This feature is good for using `bgpd` as a BGP announcement listener.

'`--enable-netlink`'
: Force to enable GNU/Linux netlink interface. Zebra configure script detects netlink interface by checking a header file. When the header file does not match to the current running kernel, configure script will not turn on netlink support.

'`--enable-snmp`'
: Enable SNMP support. By default, SNMP support is disabled.

You may specify any combination of the above options to the configure script. By default, the executables are placed in '`/usr/local/sbin`' and the configuration files in '`/usr/local/etc`'. The '`/usr/local/`' installation prefix and other directories may be changed using the following options to the configuration script.

'`--prefix=`*prefix*'
: Install architecture-independent files in *prefix* [/usr/local].

'`--sysconfdir=`*dir*'
: Read-only sample configuration file in *dir* [*prefix*/etc].

```
% ./configure --disable-ipv6
```

This command will configure zebra and the routing daemons.

There are several options available only to GNU/Linux systems:[1].

2.2 Build the Software

After configuring the software, you will need to compile it for your system. Simply issue the command `make` in the root of the source directory and the software will be compiled. If you have *any* problems at this stage, be certain to send a bug report See Section 1.7 [Bug Reports], page 5.

```
% ./configure
.
.
.
./configure output
.
.
.
% make
```

[1] GNU/Linux has very flexible kernel configuration features. If you use GNU/Linux, make sure that the current kernel configuration is what you want. Zebra will run with any kernel configuration but some recommendations do exist.

CONFIG_NETLINK
: Kernel/User netlink socket. This is a brand new feature which enables an advanced interface between the Linux kernel and Zebra (see Chapter 14 [Kernel Interface], page 73).

CONFIG_RTNETLINK
: Routing messages. This makes it possible to receive netlink routing messages. If you specify this option, `zebra` can detect routing information updates directly from the kernel (see Chapter 14 [Kernel Interface], page 73).

CONFIG_IP_MULTICAST
: IP: multicasting. This option should be specified when you use `ripd` or `ospfd` because these protocols use multicast.

IPv6 support has been added in GNU/Linux kernel version 2.2. If you try to use the Zebra IPv6 feature on a GNU/Linux kernel, please make sure the following libraries have been installed. Please note that these libraries will not be needed when you uses GNU C library 2.1 or upper.

`net-tools`
: The `net-tools` package provides an IPv6 enabled interface and routing utility. It contains `ifconfig`, `route`, `netstat`, and other tools. `net-tools` may be found at `http://www.tazenda.demon.co.uk/phil/net-tools/`.

`inet6-apps`
: The `inet6-apps` package includes basic IPv6 related libraries such as `inet_ntop` and `inet_pton`. Some basic IPv6 programs such as `ping`, `ftp`, and `inetd` are also included. The `inet-apps` can be found at `ftp://ftp.inner.net/pub/ipv6/`.

2.3 Install the Software

Installing the software to your system consists of copying the compiled programs and supporting files to a standard location. After the installation process has completed, these files have been copied from your work directory to '`/usr/local/bin`', and '`/usr/local/etc`'.

To install the Zebra suite, issue the following command at your shell prompt: `make install`.

```
%
% make install
%
```

Zebra daemons have their own terminal interface or VTY. After installation, you have to setup each beast's port number to connect to them. Please add the following entries to '`/etc/services`'.

```
zebrasrv      2600/tcp   # zebra service
zebra         2601/tcp   # zebra vty
ripd          2602/tcp   # RIPd vty
ripngd        2603/tcp   # RIPngd vty
ospfd         2604/tcp   # OSPFd vty
bgpd          2605/tcp   # BGPd vty
ospf6d        2606/tcp   # OSPF6d vty
```

If you use a FreeBSD newer than 2.2.8, the above entries are already added to '`/etc/services`' so there is no need to add it. If you specify a port number when starting the daemon, these entries may not be needed.

You may need to make changes to the config files in '`/usr/local/etc/*.conf`'. See Section 3.1 [Config Commands], page 11.

3 Basic commands

There are five routing daemons in use, and there is one manager daemon. These daemons may be located on separate machines from the manager daemon. Each of these daemons will listen on a particular port for incoming VTY connections. The routing daemons are:

- `ripd`, `ripngd`, `ospfd`, `ospf6d`, `bgpd`
- `zebra`

The following sections discuss commands common to all the routing daemons.

3.1 Config Commands

In a config file, you can write the debugging options, a vty's password, routing daemon configurations, a log file name, and so forth. This information forms the initial command set for a routing beast as it is starting.

Config files are generally found in:

‘`/usr/local/etc/*.conf`’

Each of the daemons has its own config file. For example, zebra's default config file name is:

‘`/usr/local/etc/zebra.conf`’

The daemon name plus ‘`.conf`’ is the default config file name. You can specify a config file using the *`-f`* or *`--config-file`* options when starting the daemon.

3.1.1 Basic Config Commands

hostname *hostname* Command

Set hostname of the router.

password *password* Command

Set password for vty interface. If there is no password, a vty won't accept connections.

enable password *password* Command

Set enable password.

log stdout Command
no log stdout Command

Set logging output to stdout.

log file *filename* Command

If you want to log into a file please specify `filename` as follows.

```
log file /usr/local/etc/bgpd.log
```

log syslog Command
no log syslog Command

Set logging output to syslog.

write terminal Command
: Displays the current configuration to the vty interface.

write file Command
: Write current configuration to configuration file.

configure terminal Command
: Change to configuration mode. This command is the first step to configuration.

terminal length <*0-512*> Command
: Set terminal display length to <*0-512*>. If length is 0, no display control is performed.

who Command

list Command
: List commands.

service password-encryption Command
: Encrypt password.

service advanced-vty Command
: Enable advanced mode VTY.

service terminal-length <*0-512*> Command
: Set system wide line configuration. This configuration command applies to all VTY interfaces.

show version Command
: Show the current version of the Zebra and its build host information.

line vty Command
: Enter vty configuration mode.

banner motd default Command
: Set default motd string.

no banner motd Command
: No motd banner string will be printed.

exec-timeout *minute* Line Command

exec-timeout *minute second* Line Command
: Set VTY connection timeout value. When only one argument is specified it is used for timeout value in minutes. Optional second argument is used for timeout value in seconds. Default timeout value is 10 minutes. When timeout value is zero, it means no timeout.

no exec-timeout Line Command
: Do not perform timeout at all. This command is as same as `exec-timeout 0 0`.

access-class *access-list* Line Command
: Restrict vty connections with an access list.

3.1.2 Sample Config File

Below is a sample configuration file for the zebra daemon.

```
!
! Zebra configuration file
!
hostname Router
password zebra
enable password zebra
!
log stdout
!
!
```

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

```
password zebra!password
```

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

3.2 Common Invocation Options

These options apply to all Zebra daemons.

'`-d`'
'`--daemon`'
: Runs in daemon mode.

'`-f` *file*'
'`--config_file=`*file*'
: Set configuration file name.

'`-h`'
'`--help`' Display this help and exit.

'`-i` *file*'
'`--pid_file=`*file*'
: Upon startup the process identifier of the daemon is written to a file, typically in '`/var/run`'. This file can be used by the init system to implement commands such as `.../init.d/zebra status`, `.../init.d/zebra restart` or `.../init.d/zebra stop`.

: The file name is an run-time option rather than a configure-time option so that multiple routing daemons can be run simultaneously. This is useful when using Zebra to implement a routing looking glass. One machine can be used to collect differing routing views from differing points in the network.

'`-P` *port*'
'`--vty_port=`*port*'
: Set the VTY port number.

'-v'
'--version'
Print program version.

3.3 Virtual Terminal Interfaces

VTY – Virtual Terminal [aka TeletYpe] Interface is a command line interface (CLI) for user interaction with the routing daemon.

3.3.1 VTY Overview

VTY stands for Virtual TeletYpe interface. It means you can connect to the daemon via the telnet protocol.

To enable a VTY interface, you have to setup a VTY password. If there is no VTY password, one cannot connect to the VTY interface at all.

```
% telnet localhost 2601
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.

Hello, this is zebra (version 0.93b)
Copyright 1997-2000 Kunihiro Ishiguro


User Access Verification

Password: XXXXX
Router> ?
  enable           Turn on privileged commands
  exit             Exit current mode and down to previous mode
  help             Description of the interactive help system
  list             Print command list
  show             Show running system information
  who              Display who is on a vty
Router> enable
Password: XXXXX
Router# configure terminal
Router(config)# interface eth0
Router(config-if)# ip address 10.0.0.1/8
Router(config-if)# ^Z
Router#
```

'?' is very useful for looking up commands.

3.3.2 VTY Modes

There are three basic VTY modes:

There are commands that may be restricted to specific VTY modes.

3.3.2.1 VTY View Mode

This mode is for read-only access to the CLI. One may exit the mode by leaving the system, or by entering `enable` mode.

3.3.2.2 VTY Enable Mode

This mode is for read-write access to the CLI. One may exit the mode by leaving the system, or by escaping to view mode.

3.3.2.3 VTY Other Modes

This page is for describing other modes.

3.3.3 VTY CLI Commands

Commands that you may use at the command-line are described in the following three subsubsections.

3.3.3.1 CLI Movement Commands

These commands are used for moving the CLI cursor. The ⟨C⟩ character means press the Control Key.

C-f ⟨RIGHT⟩	Move forward one character.
C-b ⟨LEFT⟩	Move backward one character.
M-f	Move forward one word.
M-b	Move backward one word.
C-a	Move to the beginning of the line.
C-e	Move to the end of the line.

3.3.3.2 CLI Editing Commands

These commands are used for editing text on a line. The ⟨C⟩ character means press the Control Key.

C-h ⟨DEL⟩	Delete the character before point.
C-d	Delete the character after point.
M-d	Forward kill word.
C-w	Backward kill word.
C-k	Kill to the end of the line.
C-u	Kill line from the beginning, erasing input.
C-t	Transpose character.

3.3.3.3 CLI Advanced Commands

There are several additional CLI commands for command line completions, insta-help, and VTY session management.

C-c Interrupt current input and moves to the next line.

C-z End current configuration session and move to top node.

C-n
⟨DOWN⟩ Move down to next line in the history buffer.

C-p
⟨UP⟩ Move up to previous line in the history buffer.

TAB Use command line completion by typing ⟨TAB⟩.

You can use command line help by typing `help` at the beginning of the line. Typing `?` at any point in the line will show possible completions.

4 Zebra

`zebra` is an IP routing manager. It provides kernel routing table updates, interface lookups, and redistribution of routes between different routing protocols.

4.1 Invoking zebra

Besides the common invocation options (see Section 3.2 [Common Invocation Options], page 13), the `zebra` specific invocation options are listed below.

'`-b`'
'`--batch`' Runs in batch mode. `zebra` parses configuration file and terminates immediately.

'`-k`'
'`--keep_kernel`'
When zebra starts up, don't delete old self inserted routes.

'`-l`'
'`--log_mode`'
Set verbose logging on.

'`-r`'
'`--retain`'
When program terminates, retain routes added by zebra.

4.2 Interface Commands

interface *ifname* Command
shutdown Interface Command
no shutdown Interface Command
Up or down the current interface.

ip address *address* Interface Command
Set ip address for the interface.

description *description* **...** Interface Command
Set description for the interface.

multicast Interface Command
no multicast Interface Command
Enable or disables multicast flag for the interface.

bandwidth <1-10000000> Interface Command
no bandwidth <1-10000000> Interface Command
Set bandwidth value to the interface. This is for calculating OSPF cost. This command does not affect the actual device configuration.

4.3 Static Route Commands

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

ip route *network gateway* Command

network is destination prefix with format of A.B.C.D/M. *gateway* is gateway for the prefix. When *gateway* is A.B.C.D format. It is taken as a IPv4 address gateway. Otherwise it is treated as an interface name.

```
ip route 10.0.0.0/8 10.0.0.2
ip route 10.0.0.0/8 ppp0
```

First example defines 10.0.0.0/8 static route with gateway 10.0.0.2. Second one defines the same prefix but with gateway to interface ppp0.

ip route *network netmask gateway* Command

This is alternate version of above command. When *network* is A.B.C.D format, user must define *netmask* value with A.B.C.D format. *gateway* is same option as above command

```
ip route 10.0.0.0 255.255.255.0 10.0.0.2
ip route 10.0.0.0 255.255.255.0 ppp0
```

This is a same setting using this statement.

ip route *network gateway distance* Command

Multiple nexthop static route

```
ip route 10.0.0.1/32 10.0.0.2
ip route 10.0.0.1/32 10.0.0.3
ip route 10.0.0.1/32 eth0
```

If there is no route to 10.0.0.2 and 10.0.0.3, and interface eth0 is reachable, then the last route is installed into the kernel.

```
zebra> show ip route
S>  10.0.0.1/32 [1/0] via 10.0.0.2 inactive
                      via 10.0.0.3 inactive
  *                   is directly connected, eth0
```

Floating static route

ipv6 route *network gateway* Command
ipv6 route *network gateway distance* Command
table *tableno* Command

Select the primary kernel routing table to be used. This only works for kernels supporting multiple routing tables (like GNU/Linux 2.2.x and later). After setting *tableno* with this command, static routes defined after this are added to the specified table.

4.4 zebra Terminal Mode Commands

show ip route Command

Display current routes which zebra holds in its database.

```
Router# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       B - BGP * - FIB route.

K* 0.0.0.0/0              203.181.89.241
S  0.0.0.0/0              203.181.89.1
C* 127.0.0.0/8            lo
C* 203.181.89.240/28      eth0
```

show ipv6 route Command

show interface Command

show ipforward Command

Display whether the host's IP forwarding function is enabled or not. Almost any UNIX kernel can be configured with IP forwarding disabled. If so, the box can't work as a router.

show ipv6forward Command

Display whether the host's IP v6 forwarding is enabled or not.

5 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 broadcasted to its neighboring router.

`ripd` supports RIP version 2 as described in RFC2453 and RIP version 1 as described in RFC1058.

5.1 Starting and Stopping ripd

The default configuration file name of `ripd`'s is '`ripd.conf`'. When invocation `ripd` searches directory /usr/local/etc. If '`ripd.conf`' is not there next search current directory.

RIP uses UDP port 521 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/ripd.pid‘`. Certain signals have special meaningss to `ripd`.

'`SIGHUP`' Reload configuration file '`ripd.conf`'. All configurations are reseted. All routes learned so far are cleared and removed from routing table.

'`SIGUSR1`' Rotate `ripd` logfile.

'`SIGINT`'
'`SIGTERM`' `ripd` sweeps all installed RIP routes then terminates properly.

`ripd` invocation options. Common options that can be specified (see Section 3.2 [Common Invocation Options], page 13).

'`-r`'
'`--retain`'
When the program terminates, retain routes added by `ripd`.

5.1.1 RIP netmask

The netmask features of `ripd` support both version 1 and version 2 of RIP. Version 1 of RIP originally contained no netmask information. In RIP version 1, network classes were originally used to determine the size of the netmask. Class A networks use 8 bits of mask, Class B networks use 16 bits of masks, while Class C networks use 24 bits of mask. Today, the most widely used method of a network mask is assigned to the packet on the basis of the interface that received the packet. Version 2 of RIP supports a variable length subnet mask (VLSM). By extending the subnet mask, the mask can be divided and reused. Each subnet can be used for different purposes such as large to middle size LANs and WAN links. Zebra `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.

5.2 RIP Configuration

router rip Command

The `router rip` command is necessary to enable RIP. To disable RIP, use the `no router rip` command. RIP must be enabled before carrying out any of the RIP commands.

no rouer rip Command

Disable RIP.

RIP can be configured to process either Version 1 or Version 2 packets, the default mode is Version 2. If no version is specified, then the RIP daemon will default to Version 2. If RIP is set to Version 1, the setting "Version 1" will be displayed, but the setting "Version 2" will not be displayed whether or not Version 2 is set explicitly as the version of RIP being used.

network *network* RIP Command
no network *network* RIP Command

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* RIP Command
no network *ifname* RIP Command

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* RIP Command
no neighbor *a.b.c.d* RIP Command

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

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

```
!
router rip
 network 10.0.0.0/8
 network eth0
!
```

Passive interface

passive-interface *IFNAME* RIP command
no passive-interface *IFNAME* RIP command

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.

RIP version handling

version *version* RIP Command

Set RIP process's version. *version* can be "1" or "2".

ip rip send version *version* Interface command

version can be '1', '2', '1 2'. This configuration command overrides the router's rip version setting. The command will enable the selected interface to send packets with RIP Version 1, RIP Version 2, or both. In the case of '1 2', packets will be both broadcast and multicast.

ip rip receive version *version* Interface command

Version setting for incoming RIP packets. This command will enable the selected interface to receive packets in RIP Version 1, RIP Version 2, or both.

RIP split-horizon

ip split-horizon Interface command
no ip split-horizon Interface command

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`.

5.3 How to Announce RIP route

redistribute kernel RIP command
redistribute kernel metric <0-16> RIP command
redistribute kernel route-map *route-map* RIP command
no redistribute kernel RIP command

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

redistribute static RIP command
redistribute static metric <0-16> RIP command
redistribute static route-map *route-map* RIP command
no redistribute static RIP command

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

redistribute connected RIP command
redistribute connected metric <0-16> RIP command
redistribute connected route-map *route-map* RIP command
no redistribute connected RIP command

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

redistribute ospf RIP command
redistribute ospf metric <0-16> RIP command
redistribute ospf route-map *route-map* RIP command
no redistribute ospf RIP command

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

redistribute bgp RIP command
redistribute bgp metric <0-16> RIP command
redistribute bgp route-map *route-map* RIP command
no redistribute bgp RIP command

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

If you want to specify RIP only static routes:

default-information originate RIP command
route *a.b.c.d/m* RIP command
no route *a.b.c.d/m* RIP command

This command is specific to Zebra. 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 Zebra and redistributing it in RIP using `redistribute static`.

5.4 Filtering RIP Routes

RIP routes can be filtered by a distribute-list.

distribute-list *access_list direct ifname* Command

You can apply access lists to the interface with a `distribute-list` command. *access_list* is the access list name. *direct* is '`in`' or '`out`'. If *direct* is '`in`' the access list is applied to input packets.

The `distribute-list` command can be used to filter the RIP path. `distribute-list` can apply access-lists to a chosen interface. First, one should specify the access-list. Next, the name of the access-list is used in the distribute-list command. For example, in the following configuration '`eth0`' will permit only the paths that match the route 10.0.0.0/8

```
!
router rip
 distribute-list private in eth0
!
access-list private permit 10 10.0.0.0/8
access-list private deny any
!
```

`distribute-list` can be applied to both incoming and outgoing data.

distribute-list prefix *prefix_list* **(in|out)** *ifname* Command

You can apply prefix lists to the interface with a `distribute-list` command. *prefix_list* is the prefix list name. Next is the direction of '`in`' or '`out`'. If *direct* is '`in`' the access list is applied to input packets.

5.5 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> RIP command
no default-metric <1-16> RIP command

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)** RIP command
offset-list *access-list* **(in|out)** *ifname* RIP command

5.6 RIP distance

Distance value is used in zebra daemon. Default RIP distance is 120.

distance <1-255> RIP command
no distance <1-255> RIP command

Set default RIP distance to specified value.

distance <1-255> *A.B.C.D/M* RIP command
no distance <1-255> *A.B.C.D/M* RIP command

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* RIP command
no distance <1-255> *A.B.C.D/M access-list* RIP command

Set default RIP distance to specified value when the route's source IP address matches the specified prefix and the specified access-list.

5.7 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 Zebra'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 (see Chapter 12 [Route Map], page 69) is needed to use route-map functionality.

match interface *word* Route Map

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* Route Map
match ip address prefix-list *word* Route Map

Match if route destination is permitted by access-list.

match ip next-hop A.B.C.D Route Map

Cisco uses here <access-list>, `ripd` IPv4 address. Match if route has this next-hop (meaning next-hop listed in the rip route table - "show ip rip")

match metric <0-4294967295> Route Map

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 Route Map

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> Route Map

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.

5.8 RIP Authentication

ip rip authentication mode md5 Interface command
no ip rip authentication mode md5 Interface command

Set the interface with RIPv2 MD5 authentication.

ip rip authentication mode text Interface command
no ip rip authentication mode text Interface command

Set the interface with RIPv2 simple password authentication.

ip rip authentication string *string* Interface command
no ip rip authentication string *string* Interface command

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* Interface command
no ip rip authentication key-chain *key-chain* Interface command

Specifiy Keyed MD5 chain.

```
!
key chain test
 key 1
  key-string test
!
interface eth1
 ip rip authentication mode md5
 ip rip authentication key-chain test
!
```

5.9 RIP Timers

timers basic *update timeout garbage* RIP command

RIP protocol has several timers. User can configure those timers' values by `timers basic` command.

The default settings for the timers are as follows:

- The update timer is 30 seconds. Every update timer seconds, the RIP process is awakened to send an unsolicited Response message containing the complete routing table to all neighboring RIP routers.
- The timeout timer is 180 seconds. Upon expiration of the timeout, the route is no longer valid; however, it is retained in the routing table for a short time so that neighbors can be notified that the route has been dropped.

- The garbage collect timer is 120 seconds. Upon expiration of the garbage-collection timer, the route is finally removed from the routing table.

The `timers basic` command allows the the default values of the timers listed above to be changed.

no timers basic RIP command

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

5.10 Show RIP Information

To display RIP routes.

show ip rip Command

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 protocols Command

The command displays current RIP status. It includes RIP timer, filtering, version, RIP enabled interface and RIP peer inforation.

```
ripd> show ip protocols
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
```

5.11 RIP Debug Commands

Debug for RIP protocol.

debug rip events Command

Debug rip events.

`debug rip` will show RIP events. Sending and receiving packets, timers, and changes in interfaces are events shown with `ripd`.

debug rip packet Command

Debug rip packet.

`debug rip packet` will 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 Command

Debug rip between zebra communication.

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 Command

Display `ripd`'s debugging option.

`show debugging rip` will show all information currently set for ripd debug.

6 RIPng

`ripngd` supports the RIPng protocol as described in RFC2080. It's an IPv6 reincarnation of the RIP protocol.

6.1 Invoking ripngd

There are no `ripngd` specific invocation options. Common options can be specified (see Section 3.2 [Common Invocation Options], page 13).

6.2 ripngd Configuration

Currently ripngd supports the following commands:

router ripng Command

Enable RIPng.

flush_timer *time* RIPng Command

Set flush timer.

network *network* RIPng Command

Set RIPng enabled interface by *network*

network *ifname* RIPng Command

Set RIPng enabled interface by *ifname*

route *network* RIPng Command

Set RIPng static routing announcement of *network*.

router zebra Command

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

6.3 ripngd Terminal Mode Commands

show ip ripng Command
show debugging ripng Command
debug ripng events Command
debug ripng packet Command
debug ripng zebra Command

6.4 ripngd Filtering Commands

distribute-list *access_list* **(in|out)** *ifname* Command

You can apply an access-list to the interface using the `distribute-list` command. *access_list* is an access-list name. *direct* is '`in`' or '`out`'. If *direct* is '`in`', the access-list is applied only to incoming packets.

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

7 OSPFv2

OSPF version 2 is a routing protocol which described in RFC2328 - *OSPF Version 2*. OSPF is IGP (Interior Gateway Protocols). Compared with RIP, OSPF can provide scalable network support and faster convergence time. OSPF is widely used in large networks such as ISP backbone and enterprise networks.

7.1 Configuring ospfd

There is no `ospfd` specific options. Common options can be specified (see Section 3.2 [Common Invocation Options], page 13) to `ospfd`. `ospfd` needs interface information from `zebra`. So please make it sure `zebra` is running before invoking `ospfd`.

Like other daemons, `ospfd` configuration is done in OSPF specific configuration file '`ospfd.conf`'.

7.2 OSPF router

To start OSPF process you have to specify the OSPF router. As of this writing, `ospfd` does not support multiple OSPF processes.

router ospf Command
no router ospf Command

Enable or disable the OSPF process. `ospfd` does not yet support multiple OSPF processes. So you can not specify an OSPF process number.

ospf router-id *a.b.c.d* OSPF Command
no ospf router-id OSPF Command
ospf abr-type *type* OSPF Command
no ospf abr-type *type* OSPF Command

type can be cisco|ibm|shortcut|standard

ospf rfc1583compatibility OSPF Command
no ospf rfc1583compatibility OSPF Command
passive interface *interface* OSPF Command
no passive interface *interface* OSPF Command
timers spf <0-4294967295> <0-4294967295> OSPF Command
no timers spf OSPF Command
refresh group-limit <0-10000> OSPF Command
refresh per-slice <0-10000> OSPF Command
refresh age-diff <0-10000> OSPF Command
auto-cost refrence-bandwidth <1-4294967> OSPF Command
no auto-cost refrence-bandwidth OSPF Command
network *a.b.c.d/m* **area** *a.b.c.d* OSPF Command
network *a.b.c.d/m* **area** *<0-4294967295>* OSPF Command
no network *a.b.c.d/m* **area** *a.b.c.d* OSPF Command
no network *a.b.c.d/m* **area** *<0-4294967295>* OSPF Command

This command specifies the OSPF enabled interface. If the interface has an address of 10.0.0.1/8 then the command below provides network information to the ospf routers

```
router ospf
 network 10.0.0.0/8 area 0
```

the network command's mask length should be the same as the interface address's mask.

7.3 OSPF area

area *a.b.c.d* **range** *a.b.c.d/m*	OSPF Command
area <0-4294967295> range *a.b.c.d/m*	OSPF Command
no area *a.b.c.d* **range** *a.b.c.d/m*	OSPF Command
no area <0-4294967295> range *a.b.c.d/m*	OSPF Command
area *a.b.c.d* **range IPV4_PREFIX suppress**	OSPF Command
no area *a.b.c.d* **range IPV4_PREFIX suppress**	OSPF Command
area *a.b.c.d* **range IPV4_PREFIX substitute IPV4_PREFIX**	OSPF Command
no area *a.b.c.d* **range IPV4_PREFIX substitute IPV4_PREFIX**	OSPF Command
area *a.b.c.d* **virtual-link** *a.b.c.d*	OSPF Command
area <0-4294967295> virtual-link *a.b.c.d*	OSPF Command
no area *a.b.c.d* **virtual-link** *a.b.c.d*	OSPF Command
no area <0-4294967295> virtual-link *a.b.c.d*	OSPF Command
area *a.b.c.d* **shortcut**	OSPF Command
area <0-4294967295> shortcut	OSPF Command
no area *a.b.c.d* **shortcut**	OSPF Command
no area <0-4294967295> shortcut	OSPF Command
area *a.b.c.d* **stub**	OSPF Command
area <0-4294967295> stub	OSPF Command
no area *a.b.c.d* **stub**	OSPF Command
no area <0-4294967295> stub	OSPF Command
area *a.b.c.d* **stub no-summary**	OSPF Command
area <0-4294967295> stub no-summary	OSPF Command
no area *a.b.c.d* **stub no-summary**	OSPF Command
no area <0-4294967295> stub no-summary	OSPF Command
area *a.b.c.d* **default-cost <0-16777215>**	OSPF Command
no area *a.b.c.d* **default-cost <0-16777215>**	OSPF Command
area *a.b.c.d* **export-list NAME**	OSPF Command
area <0-4294967295> export-list NAME	OSPF Command
no area *a.b.c.d* **export-list NAME**	OSPF Command
no area <0-4294967295> export-list NAME	OSPF Command
area *a.b.c.d* **import-list NAME**	OSPF Command
area <0-4294967295> import-list NAME	OSPF Command
no area *a.b.c.d* **import-list NAME**	OSPF Command
no area <0-4294967295> import-list NAME	OSPF Command
area *a.b.c.d* **authentication**	OSPF Command
area <0-4294967295> authentication	OSPF Command
no area *a.b.c.d* **authentication**	OSPF Command
no area <0-4294967295> authentication	OSPF Command
area *a.b.c.d* **authentication message-digest**	OSPF Command
area <0-4294967295> authentication message-digest	OSPF Command

7.4 OSPF interface

ip ospf authentication-key AUTH_KEY Interface Command
no ip ospf authentication-key Interface Command

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.

ip ospf message-digest-key KEYID md5 KEY Interface Command
no ip ospf message-digest-key Interface Command

Set OSPF authentication key to a cryptographic password. The cryptographic algorithm is MD5. KEYID identifies secret key used to create the message digest. KEY is the actual message digest key up to 16 chars.

ip ospf cost <1-65535> Interface Command
no ip ospf cost Interface Command

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> Interface Command
no ip ospf dead-interval Interface Command

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.

ip ospf hello-interval <1-65535> Interface Command
no ip ospf hello-interval Interface Command

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.

ip ospf network (broadcast|non-broadcast|point-to-multipoint|point-to-point) Interface Command
no ip ospf network Interface Command

Set explicitly network type for specifed interface.

ip ospf priority <0-255> Interface Command
no ip ospf priority Interface Command

Set RouterPriority integer value. Setting higher value, router will be more eligible to become Designated Router. Setting the value to 0, router is no longer eligible to Designated Router. The default value is 1.

ip ospf retransmit-interval <1-65535> Interface Command
no ip ospf retransmit interval Interface Command

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 Interface Command
no ip ospf transmit-delay Interface Command

Set number of seconds for InfTransDelay value. LSAs' age should be incremented by this value when transmitting. The default value is 1 seconds.

7.5 Redistribute routes to OSPF

redistribute (kernel|connected|static|rip|bgp) — OSPF Command

redistribute (kernel|connected|static|rip|bgp) *route-map* — OSPF Command

redistribute (kernel|connected|static|rip|bgp) metric-type (1|2) — OSPF Command

redistribute (kernel|connected|static|rip|bgp) metric-type (1|2) route-map *word* — OSPF Command

redistribute (kernel|connected|static|rip|bgp) metric <0-16777214> — OSPF Command

redistribute (kernel|connected|static|rip|bgp) metric <0-16777214> route-map *word* — OSPF Command

redistribute (kernel|connected|static|rip|bgp) metric-type (1|2) metric <0-16777214> — OSPF Command

redistribute (kernel|connected|static|rip|bgp) metric-type (1|2) metric <0-16777214> route-map *word* — OSPF Command

no redistribute (kernel|connected|static|rip|bgp) — OSPF Command

default-information originate — OSPF Command

default-information originate metric <0-16777214> — OSPF Command

default-information originate metric <0-16777214> metric-type (1|2) — OSPF Command

default-information originate metric <0-16777214> metric-type (1|2) route-map *word* — OSPF Command

default-information originate always — OSPF Command

default-information originate always metric <0-16777214> — OSPF Command

default-information originate always metric <0-16777214> metric-type (1|2) — OSPF Command

default-information originate always metric <0-16777214> metric-type (1|2) route-map *word* — OSPF Command

no default-information originate — OSPF Command

distribute-list NAME out (kernel|connected|static|rip|ospf — OSPF Command

no distribute-list NAME out (kernel|connected|static|rip|ospf — OSPF Command

default-metric <0-16777214> — OSPF Command

no default-metric — OSPF Command

distance <1-255> — OSPF Command

no distance <1-255> — OSPF Command

distance ospf (intra-area|inter-area|external) <1-255> — OSPF Command

no distance ospf — OSPF Command

router zebra — Command

no router zebra — Command

7.6 Showing OSPF information

show ip ospf Command

show ip ospf interface [INTERFACE] Command

show ip ospf neighbor Command

show ip ospf neighbor INTERFACE Command

show ip ospf neighbor detail Command

show ip ospf neighbor INTERFACE detail Command

show ip ospf database Command

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

show ip ospf database (asbr-summary|external|network|router|summary) *link-state-id* Command

show ip ospf database (asbr-summary|external|network|router|summary) *link-state-id* **adv-router** *adv-router* Command

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

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

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

show ip ospf database max-age Command

show ip ospf database self-originate Command

show ip ospf refresher Command

show ip ospf route Command

7.7 Debugging OSPF

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

no debug ospf packet (hello|dd|ls-request|ls-update|ls-ack|all) (send|recv) [detail] Command

debug ospf ism Command

debug ospf ism (status|events|timers) Command

no debug ospf ism Command

no debug ospf ism (status|events|timers) Command

debug ospf nsm Command

debug ospf nsm (status|events|timers) Command

no debug ospf nsm Command

no debug ospf nsm (status|events|timers) Command

debug ospf lsa Command

debug ospf lsa (generate|flooding|refresh) Command

no debug ospf lsa Command

no debug ospf lsa (generate|flooding|refresh) Command

debug ospf zebra Command

debug ospf zebra (interface|redistribute) Command

no debug ospf zebra Command

no debug ospf zebra (interface|redistribute) Command

show debugging ospf Command

8 OSPFv3

`ospf6d` is a daemon support OSPF version 3 for IPv6 network. OSPF for IPv6 is described in RFC2740.

8.1 OSPF6 router

router ospf6 — Command

router-id *a.b.c.d* — OSPF6 Command

Set router's Router-ID.

interface *ifname* **area** *area* — OSPF6 Command

Bind interface to specified area, and start sending OSPF packets. *area* can be specified as 0.

8.2 OSPF6 area

Area support for OSPFv3 is not yet implemented.

8.3 OSPF6 interface

ipv6 ospf6 cost COST — Interface Command

Sets interface's output cost. Default value is 1.

ipv6 ospf6 hello-interval HELLOINTERVAL — Interface Command

Sets interface's Hello Interval. Default 40

ipv6 ospf6 dead-interval DEADINTERVAL — Interface Command

Sets interface's Router Dead Interval. Default value is 40.

ipv6 ospf6 retransmit-interval RETRANSMITINTERVAL — Interface Command

Sets interface's Rxmt Interval. Default value is 5.

ipv6 ospf6 priority PRIORITY — Interface Command

Sets interface's Router Priority. Default value is 1.

ipv6 ospf6 transmit-delay TRANSMITDELAY — Interface Command

Sets interface's Inf-Trans-Delay. Default value is 1.

8.4 Redistribute routes to OSPF6

redistribute static — OSPF6 Command

redistribute connected — OSPF6 Command

redistribute ripng — OSPF6 Command

8.5 Showing OSPF6 information

show ipv6 ospf6 [INSTANCE_ID] Command

INSTANCE_ID is an optional OSPF instance ID. To see router ID and OSPF instance ID, simply type "show ipv6 ospf6 <cr>".

show ipv6 ospf6 database Command

This command shows LSA database summary. You can specify the type of LSA.

show ipv6 ospf6 interface Command

To see OSPF interface configuration like costs.

show ipv6 ospf6 neighbor Command

Shows state and chosen (Backup) DR of neighbor.

show ipv6 ospf6 request-list A.B.C.D Command

Shows requestlist of neighbor.

show ipv6 route ospf6 Command

This command shows internal routing table.

9 BGP

BGP stands for a Border Gateway Protocol. The lastest BGP version is 4. It is referred as BGP-4. BGP-4 is one of the Exterior Gateway Protocols and de-fact standard of Inter Domain routing protocol. BGP-4 is described in `RFC1771` - *A Border Gateway Protocol 4 (BGP-4)*.

Many extentions are added to `RFC1771`. `RFC2858` - *Multiprotocol Extensions for BGP-4* provide multiprotocol support to BGP-4.

9.1 Starting BGP

Default configuration file of `bgpd` is '`bgpd.conf`'. `bgpd` searches the current directory first then /usr/local/etc/bgpd.conf. All of bgpd's command must be configured in '`bgpd.conf`'.

`bgpd` specific invocation options are described below. Common options may also be specified (see Section 3.2 [Common Invocation Options], page 13).

'`-p` *PORT*'
'`--bgp_port=`*PORT*'
: Set the bgp protocol's port number.

'`-r`'
'`--retain`'
: When program terminates, retain BGP routes added by zebra.

9.2 BGP router

First of all you must configure BGP router with `router bgp` command. To configure BGP router, you need AS number. AS number is an identification of autonomous system. BGP protocol uses the AS number for detecting whether the BGP connection is internal one or external one.

router bgp *asn* — Command

Enable a BGP protocol process with the specified *asn*. After this statement you can input any `BGP Commands`. You can not create different BGP process under different *asn* without specifying `multiple-instance` (see Section 9.13.1 [Multiple instance], page 59).

no router bgp *asn* — Command

Destroy a BGP protocol process with the specified *asn*.

bgp router-id *A.B.C.D* — BGP

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.

9.2.1 BGP distance

distance bgp <1-255> <1-255> <1-255> BGP

This command change distance value of BGP. Each argument is distance value for external routes, internal routes and local routes.

distance <1-255> *A.B.C.D/M* BGP
distance <1-255> *A.B.C.D/M word* BGP

This command set distance value to

9.2.2 BGP decision process

1. Weight check
2. Local preference check.
3. Local route check.
4. AS path length check.
5. Origin check.
6. MED check.

9.3 BGP network

9.3.1 BGP route

network *A.B.C.D/M* BGP

This command adds the announcement network.

```
router bgp 1
 network 10.0.0.0/8
```

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; `bgp` doesn't care about IGP routes when announcing its routes.

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

9.3.2 Route Aggregation

aggregate-address *A.B.C.D/M* BGP

This command specifies an aggregate address.

aggregate-address *A.B.C.D/M* **as-set** BGP

This command specifies an aggregate address. Resulting routes inlucde AS set.

aggregate-address *A.B.C.D/M* **summary-only** BGP

This command specifies an aggregate address. Aggreated routes will not be announce.

no aggregate-address *A.B.C.D/M* BGP

9.3.3 Redistribute to BGP

redistribute kernel BGP

Redistribute kernel route to BGP process.

redistribute static BGP

Redistribute static route to BGP process.

redistribute connected BGP

Redistribute connected route to BGP process.

redistribute rip BGP

Redistribute RIP route to BGP process.

redistribute ospf BGP

Redistribute OSPF route to BGP process.

9.4 BGP Peer

9.4.1 Defining Peer

neighbor *peer* **remote-as** *asn* BGP

Creates a new neighbor whose remote-as is *asn*. *peer* can be an IPv4 address or an IPv6 address.

```
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
```

9.4.2 BGP Peer commands

In a `router bgp` clause there are neighbor specific configurations required.

neighbor *peer* **shutdown** BGP
no neighbor *peer* **shutdown** BGP

Shutdown the peer. We can delete the neighbor's configuration by `no neighbor` *peer* `remote-as` *as-number* 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.

neighbor *peer* **ebgp-multihop** BGP
no neighbor *peer* **ebgp-multihop** BGP
neighbor *peer* **description ...** BGP
no neighbor *peer* **description ...** BGP

Set description of the peer.

neighbor *peer* **version** *version* BGP
: Set up the neighbor's BGP version. *version* can be *4*, *4+* or *4-*. BGP version *4* is the default value used for BGP peering. BGP version *4+* means that the neighbor supports Multiprotocol Extensions for BGP-4. BGP version *4-* is similar but the neighbor speaks the old Internet-Draft revision 00's Multiprotocol Extensions for BGP-4. Some routing software is still using this version.

neighbor *peer* **interface** *ifname* BGP
no neighbor *peer* **interface** *ifname* BGP
: 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.

neighbor *peer* **next-hop-self** BGP
no neighbor *peer* **next-hop-self** BGP
: This command specifies an announced route's nexthop as being equivalent to the address of the bgp router.

neighbor *peer* **update-source** BGP
no neighbor *peer* **update-source** BGP
neighbor *peer* **default-originate** BGP
no neighbor *peer* **default-originate** BGP
: `bgpd`'s default is to not announce the default route (0.0.0.0/0) even it is in routing table. When you want to announce default routes to the peer, use this command.

neighbor *peer* **port** *port* BGP
neighbor *peer* **port** *port* BGP
neighbor *peer* **send-community** BGP
neighbor *peer* **send-community** BGP
neighbor *peer* **weight** *weight* BGP
no neighbor *peer* **weight** *weight* BGP
: This command specifies a default *weight* value for the neighbor's routes.

neighbor *peer* **maximum-prefix** *number* BGP
no neighbor *peer* **maximum-prefix** *number* BGP

9.4.3 Peer filtering

neighbor *peer* **distribute-list** *name* **[in|out]** BGP
: This command specifies a distribute-list for the peer. *direct* is '`in`' or '`out`'.

neighbor *peer* **prefix-list** *name* **[in|out]** BGP command
neighbor *peer* **filter-list** *name* **[in|out]** BGP command
neighbor *peer* **route-map** *name* **[in|out]** BGP
: Apply a route-map on the neighbor. *direct* must be `in` or `out`.

9.5 BGP Peer Group

neighbor *word* **peer-group** BGP

This command defines a new peer group.

neighbor *peer* **peer-group** *word* BGP

This command bind specific peer to peer group *word*.

9.6 BGP Address Family

9.7 Autonomous System

AS (Autonomous System) is one of the essential element of BGP. BGP is a distance vector routing protocol. AS framework provides distance vector metric and loop detection to BGP. `RFC1930` - *Guidelines for creation, selection, and registration of an Autonomous System (AS)* describes how to use AS.

AS number is tow octet digita value. So the value range is from 1 to 65535. AS numbers 64512 through 65535 are defined as private AS numbers. Private AS numbers must not to be advertised in the global Internet.

9.7.1 AS Path Regular Expression

AS path regular expression can be used for displaying BGP routes and AS path access list. AS path regular expression is based on `POSIX 1003.2` regular expressions. Following description is just a subset of `POSIX` regular expression. User can use full `POSIX` regular expression. Adding to that special character '_' is added for AS path regular expression.

`.`	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.
`_`	Character `_` has special meanings in AS path regular expression. 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. `show ip bgp regexp _7675_` matches to all of BGP routes which as AS number include *7675*.

9.7.2 Display BGP Routes by AS Path

To show BGP routes which has specific AS path information `show ip bgp` command can be used.

show ip bgp regexp *line* Command

This commands display BGP routes that matches AS path regular expression *line*.

9.7.3 AS Path Access List

AS path access list is user defined AS path.

ip as-path access-list *word* **{permit|deny}** *line* Command

This command defines a new AS path access list.

no ip as-path access-list *word* Command
no ip as-path access-list *word* **{permit|deny}** *line* Command

9.7.4 Using AS Path in Route Map

match as-path *word* Route Map
set as-path prepend *as-path* Route Map

9.7.5 Private AS Numbers

9.8 BGP Communities Attribute

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 `RFC1997` - *BGP Communities Attribute* and `RFC1998` - *An Application of the BGP Community Attribute in Multi-home Routing*. It is an optional transitive attribute, therefore local policy can travel through different autonomous system.

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

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

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

`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.

When BGP communities attribute is received, duplicated communities value in the communities attribute is ignored and each communities values are sorted in numerical order.

9.8.1 BGP Community Lists

BGP community list is a user defined BGP communites attribute list. BGP community list can be used for matching or manipulating BGP communities attribute in updates.

There are two types of community list. One is standard community list and another is expanded community list. Standard community list defines communities attribute. Expanded community list defines communities attribute string with regular expression. Standard community list is compiled into binary format when user define it. Standard community list will be directly compared to BGP communities attribute in BGP updates. Therefore the comparison is faster than expanded community list.

ip community-list standard *name* **{permit|deny}** *community* Command

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.

ip community-list expanded *name* **{permit|deny}** *line* Command

This command defines a new expanded community list. *line* is a string expression of communities attribute. *line* can include regular expression to match communities attribute in BGP updates.

no ip community-list *name* Command
no ip community-list standard *name* Command
no ip community-list expanded *name* Command

These commands delete community lists specified by *name*. All of community lists shares a single name space. So community lists can be removed simpley specifying community lists name.

show ip community-list Command
show ip community-list *name* Command

This command display current community list information. When *name* is specified the specified community list's information is shown.

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

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

9.8.2 Numbered BGP Community Lists

When number is used for BGP community list name, the number has special meanings. Community list number in the range from 1 and 99 is standard community list. Community list number in the range from 100 to 199 is expanded community list. These community lists are called as numbered community lists. On the other hand normal community lists is called as named community lists.

ip community-list <1-99> {permit|deny} *community* Command

This command defines a new community list. <1-99> is standard community list number. Community list name within this range defines standard community list. When *community* is empty it matches to any routes.

ip community-list <100-199> {permit|deny} *community* — Command

This command defines a new community list. <100-199> is expanded community list number. Community list name within this range defines expanded community list.

ip community-list *name* **{permit|deny}** *community* — Command

When community list type is not specifed, 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 compability. Use of this feature is not recommended.

9.8.3 BGP Community in Route Map

In Route Map (see Chapter 12 [Route Map], page 69), we can match or set BGP communities attribute. Using this feature network operator can implement their network policy based on BGP communities attribute.

Following commands can be used in Route Map.

match community *word* — Route Map
match community *word* **exact-match** — Route Map

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 spcified, match happen only when BGP updates have completely same communities value specified in the community list.

set community none — Route Map
set community *community* — Route Map
set community *community* **additive** — Route Map

This command manipulate communities value in BGP updates. When `none` is specified as communities value, it removes entire communities attribute from BGP updates. When *community* is not `none`, specified communities value is set to BGP updates. If BGP updates already has BGP communities value, the existing BGP communities value is replaced with specified *community* value. When `additive` keyword is specified, *community* is appended to the existing communities value.

set comm-list *word* **delete** — Route Map

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.

9.8.4 Display BGP Routes by Community

To show BGP routes which has specific BGP communities attribute, `show ip bgp` command can be used. The *community* value and community list can be used for `show ip bgp` command.

show ip bgp community Command
show ip bgp community *community* Command
show ip bgp community *community* **exact-match** Command

`show ip bgp community` displays BGP routes which has communities attribute. When *community* is specified, BGP routes that matches *community* value is displayed. For this command, `internet` keyword can't be used for *community* value. When `exact-match` is specified, it display only routes that have an exact match.

show ip bgp community-list *word* Command
show ip bgp community-list *word* **exact-match** Command

This commands display BGP routes that matches community list *word*. When `exact-match` is specified, display only routes that have an exact match.

9.8.5 Using BGP Communities Attribute

Following configuration is the most typical usage of BGP communities attribute. AS 7675 provides upstream Internet connection to AS 100. When following configuration exists in AS 7675, AS 100 networks operator 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
 neighbor 192.168.0.1 route-map RMAP in
!
ip community-list 70 permit 7675:70
ip community-list 70 deny
ip community-list 80 permit 7675:80
ip community-list 80 deny
ip community-list 90 permit 7675:90
ip community-list 90 deny
!
route-map RMAP permit 10
 match community 70
 set local-preference 70
!
route-map RMAP permit 20
 match community 80
 set local-preference 80
!
route-map RMAP permit 30
 match community 90
 set local-preference 90
```

Following configuration announce 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, announced route's local preference will be set to value 80.

```
router bgp 100
 network 10.0.0.0/8
 neighbor 192.168.0.2 remote-as 7675
 neighbor 192.168.0.2 route-map RMAP out
```

```
!
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
```

Following configuration is an example of BGP route filtering using communities attribute. This configuration only permit BGP routes which has BGP communities value 0:80 or 0:90. Network operator can put special internal communities value at BGP border router, then limit the BGP routes announcement into the internal network.

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

Following exmaple filter BGP routes which has communities value 1:1. When there is no match community-list returns deny. To avoid filtering all of routes, we need to define permit any at last.

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

Communities value keyword `internet` has special meanings in standard community lists. In below example `internet` act as match any. It matches all of BGP routes even if the route does not have communities attribute at all. So community list `INTERNET` is same as above example's `FILTER`.

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

Following configuration is an example of communities value deletion. With this configuration communities value 100:1 and 100:2 is 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
 neighbor 192.168.0.1 route-map RMAP in
!
ip community-list standard DEL permit 100:1 100:2
!
route-map RMAP permit 10
 set comm-list DEL delete
```

9.9 BGP 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. `10.0.0.1:100` represents

9.9.1 BGP Extended Community Lists

Expanded Community Lists is a user defined BGP Expanded Community Lists.

ip extcommunity-list standard *name* **{permit|deny}** *extcommunity* — Command

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.

ip extcommunity-list expanded *name* **{permit|deny}** *line* — Command

This command defines a new expanded extcommunity-list. *line* is a string expression of extended communities attribute. *line* can include regular expression to match extended communities attribute in BGP updates.

no ip extcommunity-list *name* — Command
no ip extcommunity-list standard *name* — Command
no ip extcommunity-list expanded *name* — Command

These commands delete extended community lists specified by *name*. All of extended community lists shares a single name space. So extended community lists can be removed simpley specifying the name.

show ip extcommunity-list — Command
show ip extcommunity-list *name* — Command

This command display current extcommunity-list information. When *name* is specified the community list's information is shown.

```
# show ip extcommunity-list
```

9.9.2 BGP Extended Communities in Route Map

match extcommunity *word* — Route Map
set extcommunity rt *extcommunity* — Route Map

This command set Route Target value.

set extcommunity soo *extcommunity* — Route Map

This command set Site of Origin value.

9.10 Displaying BGP Routes

9.10.1 Show IP BGP

show ip bgp — Command
show ip bgp *A.B.C.D* — Command
show ip bgp *X:X::X:X* — Command

This command displays BGP routes. When no route is specified it display all of IPv4 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
```

9.10.2 More Show IP BGP

show ip bgp regexp *line* — Command

This command display BGP routes using AS path regular expression (see Section 9.7.2 [Display BGP Routes by AS Path], page 48).

show ip bgp community *community* Command
show ip bgp community *community* **exact-match** Command

This command display BGP routes using *community* (see Section 9.8.4 [Display BGP Routes by Community], page 52).

show ip bgp community-list *word* Command
show ip bgp community-list *word* **exact-match** Command

This command display BGP routes using community list (see Section 9.8.4 [Display BGP Routes by Community], page 52).

show ip bgp summary Command
show ip bgp neighbor [*peer*] Command
clear ip bgp *peer* Command

Clear peers which have addresses of X.X.X.X

clear ip bgp *peer* **soft in** Command

Clear peer using soft reconfiguration.

show debug Command
debug event Command
debug update Command
debug keepalive Command
no debug event Command
no debug update Command
no debug keepalive Command

9.11 Capability Negotiation

When adding IPv6 routing information exchange feature to BGP. There were some proposals. IETF IDR working group finally take a proposal called Multiprotocol Extension for BGP. The specification is described in RFC2283. 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 remote peer supports the protocol, `bgpd` can exchange IPv6 and/or multicast routing information.

Traditional BGP does not have the feature to detect remote peer's capability whether it can handle other than IPv4 unicast routes. This is a big problem using Multiprotocol Extension for BGP to operational network. *draft-ietf-idr-bgp4-cap-neg-04.txt* is proposing a feature called Capability Negotiation. `bgpd` use this Capability Negotiation to detect remote peer's capabilities. If the peer is only configured as IPv4 unicast neighbor, `bgpd` does not send these Capability Negotiation packets.

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

If you want to completely match capabilities with remote peer. Please use `strict-capability-match` command.

neighbor *peer* **strict-capability-match** BGP
no neighbor *peer* **strict-capability-match** BGP

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** BGP
no neighbor *peer* **dont-capability-negotiate** BGP

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.

neighbor *peer* **override-capability** BGP
no neighbor *peer* **override-capability** BGP

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

9.12 Route Reflector

bgp cluster-id *a.b.c.d* BGP
neighbor *peer* **route-reflector-client** BGP
no neighbor *peer* **route-reflector-client** BGP

9.13 Route Server

At an Internet Exchange point, many ISPs are connected to each other by external BGP peering. Normally these external BGP connection are done by `full mesh` method. As with internal BGP full mesh formation, this method has a scaling problem.

This scaling problem is well known. Route Server is a method to resolve the problem. Each ISP's BGP router only peers to Route Server. Route Server serves as BGP information exchange to other BGP routers. By applying this method, numbers of BGP connections is reduced from O(n*(n-1)/2) to O(n).

Unlike normal BGP router, Route Server must have several routing tables for managing different routing policies for each BGP speaker. We call the routing tables as different `views`. `bgpd` can work as normal BGP router or Route Server or both at the same time.

9.13.1 Multiple instance

To enable multiple view function of `bgpd`, you must turn on multiple instance feature beforehand.

bgp multiple-instance Command

Enable BGP multiple instance feature. After this feature is enabled, you can make multiple BGP instances or multiple BGP views.

no bgp multiple-instance Command

Disable BGP multiple instance feature. You can not disable this feature when BGP multiple instances or views exist.

When you want to make configuration more Cisco like one,

bgp config-type cisco Command

Cisco compatible BGP configuration output.

When bgp config-type cisco is specified,

“no synchronization” is displayed. “no auto-summary” is desplayed.

“network” and “aggregate-address” argument is displayed as “A.B.C.D M.M.M.M”

Zebra: network 10.0.0.0/8 Cisco: network 10.0.0.0

Zebra: aggregate-address 192.168.0.0/24 Cisco: aggregate-address 192.168.0.0 255.255.255.0

Community attribute handling is also different. If there is no configuration is specified community attribute and extended community attribute are sent to neighbor. When user manually disable the feature community attribute is not sent to the neighbor. In case of “bgp config-type cisco” is specified, community attribute is not sent to the neighbor by default. To send community attribute user has to specify “neighbor A.B.C.D send-community” command.

! router bgp 1 neighbor 10.0.0.1 remote-as 1 no neighbor 10.0.0.1 send-community !

! router bgp 1 neighbor 10.0.0.1 remote-as 1 neighbor 10.0.0.1 send-community !

bgp config-type zebra Command

Zebra style BGP configuration. This is default.

9.13.2 BGP instance and view

BGP instance is a normal BGP process. The result of route selection goes to the kernel routing table. You can setup different AS at the same time when BGP multiple instance feature is enabled.

router bgp *as-number* Command

Make a new BGP instance. You can use arbitrary word for the *name*.

```
bgp multiple-instance
!
router bgp 1
 neighbor 10.0.0.1 remote-as 2
 neighbor 10.0.0.2 remote-as 3
!
router bgp 2
 neighbor 10.0.0.3 remote-as 4
 neighbor 10.0.0.4 remote-as 5
```

BGP view is almost same as normal BGP process. The result of route selection does not go to the kernel routing table. BGP view is only for exchanging BGP routing information.

router bgp *as-number* **view** *name* Command

Make a new BGP view. You can use arbitrary word for the *name*. This view's route selection result does not go to the kernel routing table.

With this command, you can setup Route Server like below.

```
bgp multiple-instance
!
router bgp 1 view 1
 neighbor 10.0.0.1 remote-as 2
 neighbor 10.0.0.2 remote-as 3
!
router bgp 2 view 2
 neighbor 10.0.0.3 remote-as 4
 neighbor 10.0.0.4 remote-as 5
```

9.13.3 Routing policy

You can set different routing policy for a peer. For example, you can set different filter for a peer.

```
bgp multiple-instance
!
router bgp 1 view 1
 neighbor 10.0.0.1 remote-as 2
 neighbor 10.0.0.1 distribute-list 1 in
!
router bgp 1 view 2
 neighbor 10.0.0.1 remote-as 2
 neighbor 10.0.0.1 distribute-list 2 in
```

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.

9.13.4 Viewing the view

To display routing table of BGP view, you must specify view name.

show ip bgp view *name* Command

Display routing table of BGP view *name*.

9.14 How to set up a 6-Bone connection

```
zebra configuration
===================
!
! Actually there is no need to configure zebra
!

bgpd configuration
==================
!
! This means that routes go through zebra and into the kernel.
!
router zebra
!
! 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
!
! logfile FILENAME is obsolete.  Please use log file FILENAME
!
log file bgpd.log
!
```

9.15 Dump BGP packets and table

dump bgp all *path* Command
dump bgp all *path interval* Command

Dump all BGP packet and events to *path* file.

dump bgp updates *path* Command
dump bgp updates *path interval* Command

Dump BGP updates to *path* file.

dump bgp routes *path* Command
dump bgp routes *path* Command

Dump whole BGP routing table to *path*. This is heavy process.

10 VTY shell

`vtysh` is integrated shell of Zebra software.

To use vtysh please specify —enable-vtysh to configure script. To use PAM for authentication use —with-libpam option to configure script.

vtysh only searches /usr/local/etc path for vtysh.conf which is the vtysh configuration file. Vtysh does not search current directory for configuration file because the file includes user authentication settings.

Currently, vtysh.conf has only one command.

```
!
username foo nopassword
!
```

With this set, user foo does not need password authentication for user vtysh. With PAM vtysh uses PAM authentication mechanism.

If vtysh is compiled without PAM authentication, every user can use vtysh without authentication.

11 Filtering

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

11.0.1 IP Access List

access-list *name* **permit** *ipv4-network* Command
access-list *name* **deny** *ipv4-network* Command

Basic filtering is done by `access-list` as shown in the following example.

```
access-list filter deny 10.0.0.0/9
access-list filter permit 10.0.0.0/8
```

11.0.2 IP Prefix List

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

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

ip prefix-list *name* **(permit|deny)** *prefix* [**le** *len*] [**ge** *len*] Command
ip prefix-list *name* **seq** *number* **(permit|deny)** *prefix* [**le** *len*] [**ge** *len*] Command

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

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

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

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

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

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

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

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

In the case of no le or ge command,

Version 0.85: the matching rule will apply to all prefix lengths that matched the prefix list.

Version 0.86 or later: In the case of no le or ge command, the prefix length must match exactly the length specified in the prefix list.

no ip prefix-list *name* Command

11.0.2.1 ip prefix-list description

ip prefix-list *name* **description** *desc* Command

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

no ip prefix-list *name* **description** [*desc*] Command

Deletes the description from a prefix list. It is possible to use the command without the full description.

11.0.2.2 ip prefix-list sequential number control

ip prefix-list sequence-number Command

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

no ip prefix-list sequence-number Command

With this command, the IP prefix list sequential number is not displayed.

11.0.2.3 Showing ip prefix-list

show ip prefix-list Command

Display all IP prefix lists.

show ip prefix-list *name* Command

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

show ip prefix-list *name* **seq** *num* Command

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

show ip prefix-list *name a.b.c.d/m* Command

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

show ip prefix-list *name a.b.c.d/m* **longer** Command
show ip prefix-list *name a.b.c.d/m* **first-match** Command
show ip prefix-list summary Command
show ip prefix-list summary *name* Command
show ip prefix-list detail Command
show ip prefix-list detail *name* Command

11.0.2.4 Clear counter of ip prefix-list

clear ip prefix-list Command

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

clear ip prefix-list *name* Command
clear ip prefix-list *name a.b.c.d/m* Command

12 Route Map

Route map is a very useful function in zebra. There is a match and set statement permitted in 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.

12.0.1 Route Map Command

route-map *route-map-name* **permit** *priority* Command

12.0.2 Route Map Match Command

match ip address *access_list* Route-map Command

Matches the specified *access_list*

match ip next-hop *ipv4_addr* Route-map Command

Matches the specified *ipv4_addr*.

match aspath *as_path* Route-map Command

Matches the specified *as_path*.

match metric *metric* Route-map Command

Matches the specified *metric*.

match community *community_list* Route-map Command

Matches the specified *community_list*

12.0.3 Route Map Set Command

set ip next-hop *ipv4_address* Route-map Command

Set the BGP nexthop address.

set local-preference *local_pref* Route-map Command

Set the BGP local preference.

set weight *weight* Route-map Command

Set the route's weight.

set metric *metric* Route-map Command

Set the BGP attribute MED.

set as-path prepend *as_path* Route-map Command

Set the BGP AS path to prepend.

set community *community* Route-map Command

Set the BGP community attribute.

set ipv6 next-hop global *ipv6_address* Route-map Command

Set the BGP-4+ global IPv6 nexthop address.

set ipv6 next-hop local *ipv6_address* Route-map Command

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

13 IPv6 Support

Zebra fully supports IPv6 routing. As described so far, Zebra supports RIPng, OSPFv3 and BGP-4+. You can give IPv6 addresses to an interface and configure static IPv6 routing information. Zebra-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.

13.1 Router Advertisement

ipv6 nd send-ra Interface Command

ipv6 nd prefix-advertisement *ipv6prefix* Interface Command

```
interface eth0
 ipv6 nd send-ra
 ipv6 nd prefix-advertisement 3ffe:506:5009::/64
```

14 Kernel Interface

There are several different methods for reading kernel routing table information, updating kernel routing tables, and for looking up interfaces.

‘`ioctl`’
: The ‘`ioctl`’ 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`’
: ‘`sysctl`’ 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`’
: ‘`proc filesystem`’ provides an easy way of getting kernel information.

‘`routing socket`’

‘`netlink`’
: On recent Linux kernels (2.0.x and 2.2.x), there is a kernel/user communication support called `netlink`. It makes asynchronous communication between kernel and Zebra possible, similar to a routing socket on BSD systems.

  Before you use this feature, be sure to select (in kernel configuration) the kernel/netlink support option 'Kernel/User network link driver' and 'Routing messages'.

  Today, the /dev/route special device file is obsolete. Netlink communication is done by reading/writing over netlink socket.

  After the kernel configuration, please reconfigure and rebuild Zebra. You can use netlink as a dynamic routing update channel between Zebra and the kernel.

15 SNMP Support

SNMP (Simple Network Managing Protocol) is widely implemented feature for collecting network information from router and/or host. Zebra itself does not support SNMP agent functionality. But conjuction with SNMP agent, Zebra provides routing protocol MIBs.

Zebra uses SMUX protocol (RFC1227) for making communication with SNMP agent. There are several SNMP agent which support SMUX. We recommend to use the latest `ucd-snmp` software.

15.1 How to get ucd-snmp

ucd-snmp is a free software which distributed so called "as is" software license. Please check the license which comes with distribution of `ucd-snmp`. The authors of ucd-snmp are the University of California, the University of California at Davis, and the Electrical Engineering department at the University of California at Davis.

You can get ucd-snmp from `ftp://ucd-snmp.ucdavis.edu/`. As of this writing we are testing with `ucd-snmp-4.1.pre1.tar.gz`.

To enable SMUX protocol support, please configure `ucd-snmp` like below.

```
% configure --with-mib-modules=smux
```

After compile and install `ucd-snmp`, you will need to configure smuxpeer. I'm now using configuration shown below. This means SMUX client connects to MIB 1.3.6.1.6.3.1 with password test.

```
/usr/local/share/snmp/snmpd.conf
================================
smuxpeer 1.3.6.1.6.3.1 test
```

15.2 SMUX configuration

To enable SNMP support of Zebra, you have to configure Zebra with `--enable-snmp` (see Section 2.1 [Configure the Software], page 7).

smux peer *oid* Command
no smux peer *oid* Command
smux peer *oid password* Command
no smux peer *oid password* Command

```
!
smux peer .1.3.6.1.6.3.1 test
!
```

Appendix A Zebra Protocol

Zebra Protocol is a protocol which is used between protocol daemon and zebra. Each protocol daemon sends selected routes to zebra daemon. Then zebra manages which route is installed into the forwarding table.

Zebra Protocol is a TCP-based protocol. Below is common header of Zebra Protocol.

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Length (2)           |   Command (1) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Length is total packet length including this header length. So minimum length is three. Command is Zebra Protocol command.

ZEBRA_INTERFACE_ADD	1
ZEBRA_INTERFACE_DELETE	2
ZEBRA_INTERFACE_ADDRESS_ADD	3
ZEBRA_INTERFACE_ADDRESS_DELETE	4
ZEBRA_INTERFACE_UP	5
ZEBRA_INTERFACE_DOWN	6
ZEBRA_IPV4_ROUTE_ADD	7
ZEBRA_IPV4_ROUTE_DELETE	8
ZEBRA_IPV6_ROUTE_ADD	9
ZEBRA_IPV6_ROUTE_DELETE	10
ZEBRA_REDISTRIBUTE_ADD	11
ZEBRA_REDISTRIBUTE_DELETE	12
ZEBRA_REDISTRIBUTE_DEFAULT_ADD	13
ZEBRA_REDISTRIBUTE_DEFAULT_DELETE	14
ZEBRA_IPV4_NEXTHOP_LOOKUP	15
ZEBRA_IPV6_NEXTHOP_LOOKUP	16

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Type               |             Flags             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Appendix B Packet Binary Dump Format

Zebra can dump routing protocol packet into file with a binary format (see Section 9.15 [Dump BGP packets and table], page 62).

It seems to be better that we share the MRT's header format for backward compatibility with MRT's dump logs. We should also define the binary format excluding the header, because we must support both IP v4 and v6 addresses as socket addresses and / or routing entries.

In the last meeting, we discussed to have a version field in the header. But Masaki told us that we can define new 'type' value rather than having a 'version' field, and it seems to be better because we don't need to change header format.

Here is the common header format. This is same as that of MRT.

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              Time                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Type              |            Subtype            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             Length                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

If 'type' is PROTOCOL_BGP4MP, 'subtype' is BGP4MP_STATE_CHANGE, and Address Family == IP (version 4)

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Source AS number       |     Destination AS number     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Interface Index        |       Address Family          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source IP address                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Destination IP address                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Old State          |           New State           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Where State is the value defined in RFC1771.

If 'type' is PROTOCOL_BGP4MP, 'subtype' is BGP4MP_STATE_CHANGE, and Address Family == IP version 6

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Source AS number      |     Destination AS number     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Interface Index       |      Address Family           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source IP address                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source IP address (Cont'd)              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source IP address (Cont'd)              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source IP address (Cont'd)              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination IP address                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination IP address (Cont'd)            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination IP address (Cont'd)            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination IP address (Cont'd)            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Old State         |           New State           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

If ‘type’ is PROTOCOL_BGP4MP, ‘subtype’ is BGP4MP_MESSAGE, and Address Family == IP (version 4)

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Source AS number      |     Destination AS number     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Interface Index       |      Address Family           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source IP address                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination IP address                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      BGP Message Packet                       |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Where BGP Message Packet is the whole contents of the BGP4 message including header portion.

If ‘type’ is PROTOCOL_BGP4MP, ‘subtype’ is BGP4MP_MESSAGE, and Address Family == IP version 6

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Source AS number      |     Destination AS number     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Interface Index       |      Address Family           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source IP address                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source IP address (Cont'd)              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source IP address (Cont'd)              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source IP address (Cont'd)              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination IP address                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination IP address (Cont'd)            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination IP address (Cont'd)            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination IP address (Cont'd)            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      BGP Message Packet                       |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

If ‘type’ is PROTOCOL_BGP4MP, ‘subtype’ is BGP4MP_ENTRY, and Address Family == IP (version 4)

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             View #            |             Status            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Time Last Change                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Address Family          |    SAFI       | Next-Hop-Len  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Next Hop Address                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Prefix Length |             Address Prefix [variable]         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Attribute Length        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      BGP Attribute [variable length]     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

If ‘type’ is PROTOCOL_BGP4MP, ‘subtype’ is BGP4MP_ENTRY, and Address Family == IP version 6

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            View #             |            Status             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Time Last Change                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Address Family          |    SAFI       | Next-Hop-Len  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Next Hop Address                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Next Hop Address (Cont'd)              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Next Hop Address (Cont'd)              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Next Hop Address (Cont'd)              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Prefix Length |             Address Prefix [variable]         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Address Prefix (cont'd) [variable]        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Attribute Length        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      BGP Attribute [variable length]         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

BGP4 Attribute must not contain MP_UNREACH_NLRI. If BGP Attribute has MP_REACH_NLRI field, it must has zero length NLRI, e.g., MP_REACH_NLRI has only Address Family, SAFI and next-hop values.

If 'type' is PROTOCOL_BGP4MP and 'subtype' is BGP4MP_SNAPSHOT,

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           View #              |       File Name [variable]    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The file specified in "File Name" contains all routing entries, which are in the format of "subtype == BGP4MP_ENTRY".

```
Constants:
  /* type value */
  #define MSG_PROTOCOL_BGP4MP 16
  /* subtype value */
  #define BGP4MP_STATE_CHANGE 0
  #define BGP4MP_MESSAGE 1
  #define BGP4MP_ENTRY 2
  #define BGP4MP_SNAPSHOT 3
```

Command Index

VTY Key Index

Short Contents

Table of Contents