InfiNet Wireless Router R5000

WANFlex OS User Manual

Updated**: 5 March 2008**

OS Version**: V4.29**

Table of contents

I. Introduction

1. General notes

This manual lists the commands of the WANFleX operating system used in the InfiNet Wireless Router.

For device's management and configuration a Unix-like command line interface is used. Every command is having power right after Enter key is pressed. However, each command lifetime duration is limited within one configuration session. In order to save a current configuration "**config save**" command is used.

Several commands can be grouped in one line using "**;**" character. If a wrong-syntax line is met in the group, the rest of the string is checked anyway and the wrong command is ignored. Command name can be shortened unless the ambiguity occurs.

If your terminal supports VT100 or ANSI standard you can move around the list of recently executed commands using cursor keys. Numbered list of these commands can be reviewed by "**!h**" command. Any command from this list can be available using "**!<NUMBER>**" command. TAB key performs substring search of recently executed commands.

Ctrl/R combination refreshes the command string if its content was disturbed by system messages.

The command executed with no arguments prints a short hint about its keys, parameters and syntax.

Context help can be obtained by printing "**?**" in any position of the line.

2. IP-address format

Many commands of the operating system require specification of IP-addresses.

In OS WANFleX, the IP-addressees may be specified in traditional numeric format. Optionally, the mask may be specified either by its bit length (the specified number of leading bits in the mask are set to 1, the remaining bits are reset to 0) or numeric value. The IP-address 0/0 denotes all possible IP-addresses.

Therefore, the possible formats to specify IP-addresses are:

nn.nn.nn.nn (no mask is used)

nn.nn.nn.nn/N (N is the bit length of the mask)

nn.nn.nn.nn:xxx.xxx.xxx.xxx (xxx.xxx.xxx.xxx is the numerical value of the mask)

Example:

The 192.168.9.0/24 address describes the network address 192.168.9.0 and the mask with leading 24 bits on.

The same set of addresses may be denoted as 192.168.9.0:255.255.255.0.

II.General Purpose Command Set

1. Help command

The command displays system commands information.

Syntax:

help

Description:

Displays the list of all router commands. Executed automatically, if the user types an unknown command.

2. System command

The command is used to review and update system parameters.

Syntax:

system [arguments]

Command arguments:

- **system name [system_name]**

  Assigns to the system a new name specified by system_name parameter. If the parameter is not specified, the current system name will be displayed.

  Example:

  system name revolution

- **system location [string_describing_system_location]**

  Optional character string describing the system location; used in SNMP protocol.

  Example:

  sys location On the Carlsson's rooftop

- **system user user-name**

  Assigns a name under which the system administrator enters the router from the console or remotely, using telnet/http.

  Example:

  system user root

- **system password password**

  Sets the system administrator's password.

  Example:

  system password qwerty

- **system gpsxy [E|W]XX.XXXXX [N|S]YY.YYYYY**

  Sets the geographical coordinates of the device (longitude, latitude). This parameter can be used for automatical radiomodule parameters correction when connecting to the network.

  Example:

  system gpsxy 60.40056 56.82857

- **system guest key-word-or-phrase**

  Specifies a keyword for entering a guest mode. The keyword is entered as a login, any password may be used. While in the guest mode, you cannot modify the router's configuration parameters, neither security-related parameters.

  Example:

  system guest for_members_only

- **system prompt any-word**

  Replaces the prompt on the screen with the given any-word of amaximum length of 16 characters. The resulting prompt will look as "Prompt#ttyN>".

  Example:

  system prompt MyHost

- **system [no]fastroute**

  Enables/disables the fast routing mode. In this mode the router becomes invisible for traceroute network tracing procedures, while still performing all routing functions. It is not recommended to enable the fast routing mode simultaneously on several devices connected to the same cable Ethernet segment, because this may produce a tempest of IP packets.

- **system uptime**

  Displays the duration of time elapsed since the system's last reset.

- **system version**

  Displays the software version.

- **system log [on|off][IP_address][-][show][clear]**

  Manages the system log operation. The optional IP_address parameter specifies the UNIX host where the system log is located to which messages are directed under the standard syslog protocol. The command has the following options:

  - **on** - display messages on the current console
  - **off** - stop displaying messages on the console
  - **"-"** - disable logging on the remote host
  - **show** - show the system log listing (the latest message displayed on the bottom line; message time for every message expressed in seconds/milliseconds back from the current time)
  - **clear** - clear the system log
  - **[no]filter** – this option removes neighboring identical lines from system log leaving only one copy of each message and counts their recurrence (enabled by default)

- **system icmplimit XX**

  Sets a limit to the number of outcoming ICMP packets per second (200 by default). Helps to avoid the device rebooting while network scanning programs implementation. Being set to 0 all limitations are turned off

- **system [no]sendredirects**

  Enables (disables) the system to send *icmp redirect* messages for the packets source suppression if routing is incorrectly configured.

- **system [no]dropredirects**

Enables (disables) the system to send *icmp redirect* messages for routing tables updating if routing is incorrectly configured.

- **system cpu**

  Indicates current CPU load (in percent)

- **system OfficialAddress IP-address**

  Sets the IP-address which will be used as a source IP-address in all outgoing connections of the unit.

- **system search [seconds]**

  This command forces all indication to blink for the one type devices search. By default, this mode turns off after 10 seconds.

- **system serialCD [no]log [no]trap**

  Sets the reaction to appearance/disappearance of Carrier Detect (CD) signal on the diagnostics port (console).

  "**log**" option enables this event to be written into the system log. "**trap**" option enables SNMP-trap messages generation
  The message of CD signal disappearance has an oid of 1.3.6.1.4.1.3942.0.103
  The message of CD signal appearance has an oid of 1.3.6.1.4.1.3942.0.104
  In order to send SNMP messages you need to run trapd service using trapd command (see **trapd** command).

The current remote log IP-address, if any, may be saved in the configuration file using the **config save** command.

3. Set command (time zone settings)

The command is used for time zone settings manipulations. Starting from **3.36** version of firmware.

Syntax:

set TZ TIMEZONE

Example:

set TZ EST+5EDT,M4.1.0/2,M10.5.0/2

set TZ EKT+5

For more details please visit: http://www.rt.com/man/tzset.3.html

4. Config command (configuration manipulations)

This command is used to view, save, export, and import the router configuration.

Syntax:

config show [patterns]

config save

config export|import login:password@host/file

Description:

- **show**

  Displays the current configuration of the system. Any change of the system parameters may be immediately viewed using the config show

command. The optional parameter may contain a selection of WanFleX commands (abbreviated to their initial letters), as shown in the following examples; only those system parameters will then be displayed which relate to the commands selected.

Examples:

co show rma rip

will display parameter configuration for RMA and RIP protocols;

co show r !rip

will display parameter configuration for all commands starting with "r" except "rip".

- **save**

  Saves the current system configuration in the router's flash memory for subsequent permanent use. All modifications to the system parameters, if not saved by this command, are valid only during the current session (until the system reset).

- **export, import**

  Saves the router configuration on a remote server and reloads it from a remote server. The information is transferred using FTP. The name of the file to or from which the information is transferred. The file name shall be specified in full, in the format of the remote server's file system.

  Example:

config export user:secret@192.168.1.1/var/conf/test.cfg

5. Flashnet command (firmware uploading)

This command uploads a new version of software.

Syntax:

flashnet get login[:password]@IP-address/file-name

Description:

Loads a new software version into the router from a remote server using FTP. **File-name** is the name of the file containing the information transferred. The file name shall be specified in full, in the format of the remote server's file system. **IP-address** the IP_address of the remote server.

Example:

flashnet get upgrade@192.168.1.1/conf/rwr/rwr300.bin

Loading consists of two phases:

- Reading the file from the remote server
- Loading the system image in the router memory

The second phase is shown on the screen by alternate signs "O" and ".".

Do not interrupt this process!! If you do, the device would become unusable and could be put back to work only by the manufacturer.

6. Restart command

The command performs soft router reset.

Syntax:

restart [y]

restart SECONDS

restart stop

Description:

Full reset and reinitialization of a router. Equivalent to toggling the power switch off and on. May be used to restore initial configuration after a number of unsuccessful attempts to understand what exactly is done wrong, and after loading a new version of software. With the "**y**" option, RESTART command is executed immediately, without asking the operator for confirmation.

This command can be used for the postponed reinitialization (after certain number of seconds, e.g. **restart 300**). This option can be useful in case of dangerous manipulations with device's configuration when there is a risk to loose control over the device. The system will periodically inform the user about the time left to reinitialization by putting the corresponding message to the system log. Repeated call of this command will start the countdown from the beginning. **Restart stop** command will cancel a postponed reinitialization.

7. Ping command

The command sends test packets.

Syntax:

ping IP-address [size size] [count count]

Description:

Sends test packets (ICMP_ECHO_REQUEST) to the given IP-address. Enables to estimate attainability of a host and the destination response time. The command has the following parameters:

- **IP-address** - the IP-address of the tested host;
- **size** - the test packet length within the range of 10 to 8000 bytes (optional, 64 by default);
- **count** - the number of the test packets (optional, 5 by default).

Example:

ping 192.168.1.1 size 20 count 7

8. Telnet command

Use telnet protocol to enter a remote host.

Syntax:

telnet IP-address

Description:

Sets up a connection with a remote host specified by the **IP-address** in the terminal emulation mode. The **telnet** command uses transparent symbols stream without any intermediate interpretation; therefore, the terminal type is defined by the terminal from which the command has been executed. To interrupt the terminal emulation session, press Ctrl/D.

9. Tracert command

The command trace attainability of an IP-node.

Syntax:

tracert [-s SourceAddress] HostAddress

Description:

Traces the packet transmission path up to the IP node (host), specified by the HostAddress parameter. The command sends packets to the specified host, assigning different values to the "time to live" field in their IP headers, and analyzes "ICMP TIME_EXCEEDED" indications coming from different routers along the path to that host.

By default, the sending interface's address is put in the "source address" field of the packets. Using the **-s** option, any other IP address (SourceAddress) may be substituted for this default address.

Tracing is limited to a path with maximum 30 intermediate IP nodes. Trace packets are 36 bytes long. The trace procedure makes 3 attempts for every intermediate node.

Every trace result contains the IP-address of an intermediate node and the response time (in milliseconds) of every attempt. In addition, it may contain some special symbols corresponding to specific reply codes of the ICMP protocol:

- ! - Port unattainable
- !N - Network unattainable
- !H - Node (host) unattainable
- !P - Inappropriate protocol
- !F - Too long packet
- !X - Access to a node is administratively restricted (filter, proxy etc.)
- * - No reply

10.Rshd command (Remote Shell)

RSH (remote shell) protocol support module.

Syntax:

rshd enable | disable RemoteUSER RemoteHOST LocalUSER

rshd start | stop | flush

Description:

The built-in RSH server makes it possible remote command execution using the rsh program. Identification is based on using privileged TCP ports and a list of authorized hosts.

By default, the RSH server is disabled. To start and stop the server, the commands **rshd start** and **rshd stop** are executed. When started, the server ignores requests for command execution until at least one valid system entry is enabled.
A system entry is specified by an rshd enable command with three parameters:

- **RemoteUSER** - the name of a remote user (up to 16 symbols)
- **RemoteHOST** – IP-address of a remote host
- **LocalUSER** - the name of a local user (up to 16 symbols)

A request for command execution is serviced only if for all three parameters it specifies the values corresponding to a valid entry.

Up to 6 independent entries may be defined.

The name of a local user is in no relation with the WANFleX main authorization system; it may be considered simply as a keyword.

To disable an entry, an **rshd disable** command is executed with parameters defining that entry.

The **rshd flush** command clears the rsh server configuration.

The RSH server may be conveniently used e.g. for periodic reading of a router statistics:

rsh -l mysecretuser RWR.domain.ru ipstat get

Example:

rshd enable admin 195.38.44.1 mysecretuser

rshd enable root 195.38.45.123 mysecret2

rshd start

11.Ipstat command (IP-statistics)

IP statistics gathering module.

Syntax:

ipstat enable [items] | disable

ipstat clear | getclear

ipstat fixit | fixget | fixclear

ipstat strict | -strict

ipstat add [ifname] rule...

ipstat del N

ipstat traffic

Description:

The IP statistics gathering module provides for collecting information on dataflows traversing the router, for further analysis and/or for accounting.

Information is accumulated in the router's RAM memory as a series of records having three fields: source address, destination address, number of bytes transferred. By default, only outgoing packets are counted, at the moment they are sent to a physical interface. One record takes 12 bytes.

The maximum number of records is specified by the items numeric parameter of an ipstat enable items command; it shall not exceed the size of memory available. By default the number of records is 1000; typically it's sufficient for recording 15 to 20 minutes of operation of a client unit.

Accumulated information is displayed on the current terminal (or rsh session) using the following commands:

- **ipstat getclear** - show and clear accumulated statistical info
- **ipstat clear** - clear accumulated statistical info

A more reliable method of remotely getting statistical info consists in using the following commands:

- **ipstat fixit** - dumps the currently collected info from the router's memory into an intermediate buffer. The memory is cleared, and continues receiving info over again.
- **ipstat fixget** - shows the content of the dump buffer. This command may be executed any number of times, with no damage to the dumped statistical info.
- **ipstat fixclear** - clears the temporary dump buffer

The listing of statistical info provides:

- time elapsed since the previous "clear" operation
- number of records effectively used, and total record space available
- number of bytes lost due to record memory overflow list of all records.

If the record table in the router memory overflows, or if there is not enough memory currently available, an appropriate warning is written into the system log, and further statistical data are discarded. If enable strict option has been specified, then at the overflow condition the transit routing is disabled, but the router still responds to any protocol.

The **ipstat add [ifname] rule** command makes it possible to filter packets for statistic gathering, taking into account only those packets which satisfy the rule. The syntax of the "**rule**" parameter is the same as defined in the **ipfw** command description.
The **ipstat del N** command deletes the N-th rule from the list of rules. The **ipstat traffic [XX]** allows for visually inspecting statistics collection process in real time. Maximum XX lines are displayed (23 by default), sorted according to data rates of individual dataflows in decreasing order.

This is the script for the reliable device statistics receiving with **rsh** command usage:

```
#!/usr/bin/perl -w
for(;;)
{
   my $stat;
   do
   {
      $stat = system("rsh -t 30 -n -l root IWR_IP
 ips fixit >/dev/null");
      if(int($stat) != 0) { sleep(5); }
   } while (int($stat) != 0);
   do
   {
      $stat = system("rsh -t 30 -n -l root IWR_IP
 ips fixget >stat.tmp");
      if(int($stat) != 0) { sleep(5); }
   } while (int($stat) != 0);
   do
   {
      $stat = system("rsh -t 30 -n -l root IWR_IP
 ips fixclear >/dev/null");
      if(int($stat) != 0) { sleep(5); }
```

```
   } while (int($stat) != 0);

   system("cat stat.tmp >>stat.txt");

   sleep(300);
}
```

12.Acl command (Access Control Lists)

Access Control Lists.

Syntax:

acl add $NAME TYPE params...

acl del $NAME [params...]

acl ren $NAME1 $NAME2

acl flush

Command description

While network planning you may often need to group similar parameters in lists which can be used for different filters (e.g. **ipfw**, **qm**, **ipstat**). Access control lists (ACL) can effectively solve this problem.

acl add command creates an access list of NAME title and TYPE type. Lists names MUST start with $ symbol and can include up to 7 letters, digits and other symols excluding spaces and semicolon. At the same time the command can contain several parameters of TYPE type which will be included in the list. If the list with this name has been already created listed parameters will be attached to this list.

acl del command deletes specified parameters from the NAME list. If none of parameters are mentioned all list will be deleted.

acl rename command changes list's name from NAME1 to NAME2.

acl flush command deletes all lists

Accepted list types (TYPE):

net - contains network addresses in dot format.

xxx.xxx.xxx.xxx or **xxx.xxx.xxx.xxx/MASK_LENGTH** or

xxx.xxx.xxx.xxx/xxx.xxx.xxx.xxx

Lists of net type optimize their parameters by excluding duplicates and by having the feature that enables bigger networks include smaller networks. For example, if the list contained 1.1.1.1 parameter, when you include 1.1.1.0/24 parameter in the list 1.1.1.1 will be excluded.

Example:

acl add $LIST1 net 10.0.0.0/8 192.168.0.0/16 5.5.5.5

acl del $LIST1 5.5.5.5

13.Sntp command

SNTP parameters management.

SNTP support developed in WANFlex lets the system to synchronize the time with configured NTP server using fourth version of SNTP protocol **RFC 2030**.

Client works in unicast server request mode in certain time range.

Syntax:

sntp [options] [command]

Commands are the following:

- **start** - start service
- **stop** - stop service

Options are the following:

- -**server**={ipaddr} - set sntp server address
- -**interval**={seconds} - specify poll interval in seconds
- -**debug**={on|off} - enable/disable debug information

Example:

sntp -interval=3600 -debug=on

sntp -server=9.1.1.1 start

Commands:

- **start**

  Make the process of time synchronization active.

  Example:

  sntp start

- **stop**

  Stop the process of time synchronization.

  Example:

  sntp stop

Parameters:

The parameters can be set using any sequence with or without the command itself.

- **server**

  Using **server** parameter you can set the IP-address of your NTP server.

  Example:

  sntp -server=9.1.1.1

- **interval**

  Using **interval** parameter one can set the time value (in seconds) defining client's periodicity of NTP server requesting. 3600 by deafult.

  Example:

  console> sntp -interval=5000

- **debug**

  This parameter enables/disables printing of debugging information (packets) in the system log of WANFlex OS.

Example:

sntp -debug=on

sntp -debug=off

14.Date command

Date and time management.

This command shows or sets the date and time in WANFlex system. While setting the date and time not only kernel clock is being changed but hardware clock changes its value either (if the device supports this feature).

Syntax:

date [[[[[cc]yy]mm]dd]HH]MM[.ss]]

cc 19th or 20th century (is added before Year)

yy Year in abbrevated form (i.e. 89 for 1989, 05 for 2005)

mm Month in numeric form (1 to 12)

dd Day (1 to 31)

HH Hour (0 to 23)

MM Minute (0 to 59)

ss Second (0 to 61 - 59 plus maximum two leap seconds)

Example:

date 20040210053004

Tue Feb 10 05:30:04 2004

date

Tue Feb 10 05:30:10 2004

III.Layer 2 commands set – PHY and MAC

1. Rfconfig command (Radiointerface configuration)

The command is used to configure a radio module.

Syntax:

rfconfig if-name[sid identifier] [bitr speed] [freq frequency]

[[-]burst]

[[-]wocd]

[pwr power]

[[-]pwrctl]

[mod ofdm | CCK]

[rtsthreshold XXX]

[statistics]

[band full|half|quarter]

[capabilities]

[chntime XXX]

[[-]long]

[distance XXX | auto]

[[-]bcsid]

[txrt XXX]

[txvrt XXX]

[noise XXX]

Attention: not all of radio interfaces have the same set of parameters and options because it depends on the type and standard of the radio module. A complete list of parameters available for the specific interface can be displayed using "**rf ifname ?**" command. Radio module type and its features list are displayed by "**rf ifname cap**" command.

Description:

Sets parameters of a radio module specified by the **if-name** (name of the radiointerface) parameter, or displays them if executed without any optional parameter. Optional parameters are as follows:

- **sid**: system identifier of the router, a hexadecimal number in the range of 1H to FFFFFFH. All routers that are supposed to see each other on the same radio link must have the same identifier.
- **bitr**: the bit transfer rate (in Kbit/s) of the radio link. Allowed values are:
  - For CKK mode: 11000, 5500, 2000, 1000 Kbit/s
  - For OFDM mode: 6000, 9000, 12000, 18000, 24000, 36000, 48000, 54000 Kbit/s.

  The maximum bitrate is limited by the specific model.

  If the router is configured as a BS and **autobitrate** mode is turned on, ` this parameter will specify the maximum transmitting speed for the BS.
- **freq**: the radio link frequency (in MHz) The list of allowed frequencies can be obtained by executing "**rf if-name cap**" command.

- **burst**: enables the BURST protocol. "**-burst**" disables the BURST protocol support. Initial setting is "**-burst**".

  BURST protocol consists in grouping several short packets with the same destination address on a radio link into larger packets, thus cardinally decreasing the response time for applications generating streams of short packets. Burst enabling relates to a radio interface as a whole, and means only that you want to use this mode in this device; but the BURST protocol can only work for destinations where it is also enabled at the other end, and only if the RMA protocol is used at both sides.

  Burst enabling does not induce any changes in the work of other devices in the network. The BURST protocol reaches its maximal efficiency on high throughput point-to-point "backbone" links with ptp mode enabled.

  Some statistics about BURST protocol can be viewed by using the **muf stat** command.

- **wocd**: disables carrier sense on the radio interface when sending data packets (for 2 Mbit/s radio modules only). May only be used on a base station working in WMA (Wireless Marker Access) mode. Reduces the number of retransmitted packets, thus increasing effective throughput of a multipoint radio link. "-vocd" re-enables the standard mechanism of permanent carrier sense. Initial setting is "-vocd" (carrier sense enabled).

- **pwr**: sets the emitting power of a transmitter (in milliwatts). The values of acceptable trasmit power levels can vary depending on the type of radio module installed. The full list of acceptable trasmit power levels is available by using "**rf if-name cap**abilitites" command. If the value entered is invalid the system will automatically set a minimal adjacent value.

- **band {full|half|quarter}**

  This option allows choosing the channel width for transmitting. **Full** means 20 MHz, **half** – 10 MHz, **quarter** – 5 MHz. Units in one PTP or PTM connection must have the same channel width. Example:

  rf rf4.0 band half

- **pwrctl**: enables authomatic management of transmitter's emitting power. This mode is recommended for use in subscriber units with Cisco 350 modules (with extended range of power management), communicating with a base station (BS) having two antennas (RWR3513) and the **antenna diversity** option enabled. In such a case, the SM will try to maintain optimal transmitting power (delivering amplitude values 4 to 7at the receiving BS), thus providing a reliable reception of the SM's packets by the BS, while excluding spurious actions of BS antenna selection at the SM. This mechanism **will not work** properly if using external amplifiers.

- **chntime**. Channel Burst Time. The time for the unit to occupy the radio channel. By default – 0 (dynamic, is not displayed in the configuration). Set in microseconds. Recommended value for PTP links – 5000 (20 MHz channel width)

- **mod**: sets the modulation technique for the radio. For 5-6GHz radios OFDM mode is always turned on. For 2.4GHz radios both **cck** and **ofdm** modes are available:

  - **cck**
  - **ofdm**

  Devices that have different modulation cannot work with each other.

- **rtsthreshold**: sets the maximum packet length (in bytes), after which the RTS/CTS Virtual Carrier Sense mechanism (i.e. radio channel reservation) is to be used. Default value is 2048.
- **statistics**: displays current values of the radio module's statistics with 1 sec interval.

Below tables show "**rfconfig stat**" command output for 5GHz and 2.4GHz devices

"rf stat" output for 5GHz devices description	
Parameter	**Description**
Broadcast rate	Current bitrate value for Broadcast and Multicast packets on the BS; depends upon the speed of the slowest CPE
Voice Mode	ON/OFF value. If turned ON, the mode of their prioritized processing is turned on
Bytes Received	Number of received bytes including headers
Bytes Transmitted	Number of transmitted bytes including headers
Packets Received OK	Number of correctly received packets
Packets Transmitted OK	Number of correctly transmitted packets
Duplicate Received	Number of duplicate packets received due to protocol excesses
Total Retries	Total number of retries
FIFO Overrun	Number of FIFO queues overruns in the radio when receiving
FIFO Underrun	Number of FIFO queues underruns in the radio while transmitting
CRC Errors	Number of received packets with CRC errors
Excessive Retries	Number of packets which were not transmitted with maximal number of retries
Noise Floor	Input noise level. Measurement cycle –10 seconds
Noise Floor Threshold	Noise Floor Threshold for Carrier Detect

"rf stat" command output for 2.4GHz devices	
Parameter	**Description**
BeaconsReceived	number of received service packets (not used)
BeaconsTransmitted	Number of transmitted service packets (not used)
AckPackets Transmitted	Number of transmitted acknowledgement packets
RTS Packets Transmitted	Number of RTS packets transmitted
CTS PacketsTransmitted	Number of CTS packets transmitted
PLCP CRCErrors	CRC errors counter

Single Collisions	Single collisions counter
PLCP FormatErrors	PLCP format errors number
Polling cache aged	Expired cache lifetime packets number
PLCP LengthErrors	PLCP length errors number
NoDeferral	Non-deferred packets number
MAC CRCErrors	CRC check errors number
Deferred Protocol	Protocol deferred packets number
PartialReceived	Number of partially received packets
Deferred EnergyDetect	Number of deferred packets due to media unavailability
SSIDMismatches	Number of SSID mismatches (useless in our case)
RetryLong	Number of retries for the packets which exceeded RTC threshold
APMismatches	Access Point mismatches (useless in our case)
RetryShort	Number of retries for the packets which were lower than RTC threshold
Data RateMismatches	Data rate mismatches number
AuthenticationRejects	Authentication rejects number
AckReceived	Number of received acknowledgements for packets transmitting
AuthenticationT/O	Number of authentication timeouts
No AckReceived	Number of not acknowledged packets when transmitting
AssociationRejects	Association rejects number
CTSReceived	Number of received CTS packets
AssociationT/O	Number of association timeouts
No CTSReceived	Number of not received CTS packets
PacketsAged	Number of expired lifetime packets after receiving

PLCP - physical layer convergence protocol

- **capabilities**: displays the radio module's internal information on its operating features including acceptable trasmit power levels, frequencies etc.
- **distance**: this parameter is used as an alternative method of link range configuring in order to set the exact distance value between two devices (in kilometers). This parameter changes time values for some delays and time-outs of thus making possible to work on longer distances with smooth adjustment.

  There are several ways to manage this parameter:

  - if you set an exact value, this value is used no matter what the connection method is used

- If the CPE has auto value instead of a number (by default), the CPE will configure its parameters using Base Station commands. It is enough to set a numeric value on a Base Station (the distance to the remotest CPE); all other CPEs will automatically adjust their work. While configuration showing, there might be the current distance value after **auto** parameter: **auto (XX)**
- when knowing exact device's geographical coordinates (e.g. using GPS) you can specify their values in **sys gpsxy** command and distance parameter set as auto on all devices including the Base Station. In this case devices will automatically adjust their settings selecting an optimal value for the **distance** parameter. Base Station will calculate a distance to the remotest subscriber, and subscriber will calculate a distance to the base station. If the CPE has a link coordinates information it will use this information, otherwise it will use the **distance** parameter value got from the base station.

If **distance** parameter is set to 0 radio module will use default settings.

- **long**: this parameter is used on the long distances. Should be used with a distance more than 23 km. This mode is to be turned on at the BS, CPEs will automatically switch to it.
- **bcsid**: enables SID broadcasting (in beacon service packets which are regulary sent by the radio module). This feature is a potential weak place in the security because it lets the device to reply to subscriber's cards requests having no SID value (ANY). Though RMA and MINT protocols exclude the possibility of unauthorized network connection, such device's behaviour leads to its unstable work, failures and communication link delays.
- **txrt** (transmit retries): maximum number of repeat requests when sending unicast packets. 16 by default
- **txvrt** (transmit voice retries): maximum number of repeat requests for data packets (excluding voice packets) in voice mode. 5 by default. Voice mode turns on automatically when IP-traffic appears. Maximum value - 64.
- **noise** sets Noise Floor Threshold for radio interface. Measured in decibel. By default Noise Floor Threshold is 25 db. Noise Floor Threshold is defined as a positive shift relative to the current level of noise which is measured by a device. The unit begins data transmission only when there are no signals in the air that have signal level higher than Noise Floor Threshold. See Noise Floor and Noise floor Threshold values with "*rf IFNAME stat*" command.

Examples :

rfconfig rf3.0 sid 1 bitr 2000 freq 2427 burst

rfconfig rf4.0 bitr 18000 freq 5280 sid 01020304 burst

rfconfig rf4.0 pwr 63 distance 1

2. RMA command (Routed Multiple Access configuration)

RMA command is used for Router Multiple Access protocol configuration.

General Description

Every wireless router can be configured as a client unit or as a base station. Each link between a base station and a client unit is described as a point-to-point connection (or a subnetwork consisting of two units).

From external networks each client unit will be seen as a subnetwork with the mask length of 30, and the information on attainability of this subnetwork and of all subnetworks connected to the client unit will be spread by all base stations, where this client unit is described.

The RMA protocol provides a mechanism enabling to direct routing information relating to a client unit in such a way that at any moment of time it comes out only from that particular base station, to which this client unit is actually connected.

The routing information can be processed using any suitable routing protocol (RIP II in this case).

RMA-related commands may be classified as follows:

rma ab ...	Base station configuration
rma bs ...	Client unit configuration
rma IFNAME [-]noab rma IFNAME [-]nobs	Disables client's registration and/or base station search on the specified interface
rma IFname test | auto | autobs	Test regimes management
rma IFname loamp=XX	Set the minimal signal amplitude from client station having which base station allows client's registration. Range: 0 to 16. 0 by default
rma IFNAME [-]autobitrate	turn on the mode of automatic bitrate management
rma IFname [-]defroutebs [metric N]	set default route to the active base station
rma start | stop | show	RMA protocol operation management
rma IFname poll | poll2 | -poll	WMA/CD (polling) regime management
rma key XXX	Personal device secret key device which is used for the electronic signatures (for service packets) while base station authentification process (including registration via RAPS) and transparent encoding of data sent (optional). If this parameter is set at the client's side it should be known to the base station (through RAPS or through rma ab key=XXX). Key is a string of 32 characters maximum without spaces.
rma bckey XXX	This key is used for signatures and encoding of the broadcast packets (optional). It is commom for all the network. Key is a string of 32 characters maximum without spaces.
rma IFNAME [-]crypt	Enables data transparent encoding mode (should be ordered separately). If used must be turned on on all devices.
rma id XXXX	RAPS (Remote Access Permission

rma server IP[:PORT] Service) commands

Subscriber unit configuration

Syntax:

rma bs if-name sid[/speed/minspeed] freq [,freq...] [-del]

Description:

Configuring a client unit by providing it with parameters of one or several base stations with which the client unit may potentially be allowed to interact. When executed without parameters, the **RMA BS** command displays the status of the base station selected and the current connection quality.

The command has the following parameters:

- **if-name** - the name of an interface;
- **sid** - a network identifier;
- **speed** - requested connection speed in KBit/s (if autobitrate mode is turned on, this parameter sets the highest possible speed). The list of available speeds can be displayed by "rf if-name bitr" command and depends on radio module type. In **autobitrate** mode sets the maximum value for the transmitting speed parameter from CPE.
- **minspeed** – the minimal speed for autobitrate mechanism. This parameter is optional. This parameter sets a symmetrical limitation for the transmitting speed both for BS and CPE in **autobitrate** mode.
- **freq** - the base station operating frequencies; any number of frequencies may be specified, as a list of individual frequencies and/or of frequency ranges f1-f2, separated by commas.

The **-DEL** option is used to remove the specified parameters from the subscriber unit's configuration table.

Base Station configuration

Syntax:

rma ab if-name mac myip [name] [options]

rma ab if-name mac [name] [options]

rma abN if-name [options]

Options are the following: **-disable** | **-enable** | **-del** | **-dumb** | **-nodumb**

Description:

Configures a router as a base station by providing it with a table of client units which are allowed to work with this base station. As it was already mentioned, each link with a client unit is described as a separate point-to-point subnetwork with the mask length of 30. In addition, MAC-addresses of each client unit shall be specified.

The command has the following parameters:

- **if-name** - the name of an interface
- **mac** – MAC-address of a client unit.

- **myip** – IP-address of the connection as seen from the base station (mask length equals 30).
- **name** - subscriber's mnemonic name (up to 20 symbols without spaces).

Possible options:

-disable|-enable: disables and enables the RMA mechanism.
-del: removes the specified parameters from the BS configuration table.

When executed without optional parameters, the **rma ab** command shows the status of all client units registered on the base station and current connection quality, as shown on the following example:

1	2	3	4	5	6	7
ab1	00028ae1d72a	NewRWR.0.1	9.9.0.1->2	11/8	A/B/P	<18/36>
ab2	00028ae1d795	NewRWR.0.5	9.9.0.5			

Fields description:

Field#	Description
1	System assigned device's name. Can be used instead of a MAC-address while configuring
2	Subscriber unit MAC address.
3	Unit mnemonic name.
4	IP-addresses: (base station)>(subscriber unit). The address of a subscriber unit is included into the table after successful registration of the unit.
5	Current connection quality measured in relative units from 0 to 16. 0 indicates the weakest signal quality, 16 – the strongest. <signal level from CPE>/<signal level from BS>
6	Status field. A – active B – burst mode on P – polling mode on
7	<transmitting speed>/<receiving speed>

Example:

rma ab rf4.0 00:00:00:00:00:11 10.10.1.1 Subscriber1

rma ab rf4.0 00:00:00:00:00:55 10.10.1.5 Subscriber2

rma ab rf4.0 00:00:00:00:00:99 10.10.1.9 Subscriber3

rma start

rip start

Test regimes

The InfiNet Wireless Routers support three test regimes:

1. **Manual test** of a link to a client unit or base station with specified MAC address. Any device can be tested in this mode, and testing results can be seen on the console and on the front panel LED indicators.

Syntax:

rma if-name test mac [len] [mcast]

Here:

- **mac**: MAC-address of the unit tested; instead, you can specify a keyword abN or bs.
- **len**: an optional parameter, allowing you to specify the test packet length. Default length is 1024 bytes.
- **mcast**: this option enables the user to carry out testing with broadcast packets. In this case the results may be much worse than in the unicast mode.

One can use keywords abN or bs instead of the MAC-address, but this command interpretation will work only if the connection is already established (a client is registered on the base station)

2. **Automatic test** of a link to a client station **with specified MAC-address**. This is a "deferred" test regime.MAC address of the unit to be tested is stored in the router memory, and the test is executed immediately after the power is switched on. The command shall be followed by the **config save** command.

Syntax:

rma if-name auto mac [len] [mcast]

config save

The meaning of parameters is as for the **rma test** command above.

Example:

rma rf4.0 auto 00:00:00:00:00:55 2048 mcast

config save

3. **Automatic test** of a link to all client units **registered at the base station**. The test is preceded by automatic search of the unit stations. MAC addresses of the units to be tested are saved in the router memory, and the test is executed immediately after the power is switched on. The command is to be followed by the **config save** command.

Syntax:

rma if-name autobs [len] [mcast]

config save

The meaning of parameters is as for the **rma test** command above.

Example:

rma rf4.0 autobs 2048 mcast

config save

Disabling automatic test regimes

Syntax:

rma IFNAME -autotest

config save

Packet length is 256 bytes by default.

The testing protocol is always available regardless of the router's work mode (including RMA). However if a client is searching the base station and constantly changing radio module parameters, you will not get any positive results. Therefore it is reasonable to run the test on the client's side when there is an established connection with the base station or one should turn RMA mode off (do not forget to check current radio module parameters). It is not recommended to run two tests simultaniously.

RMA protocol operation management

Syntax:

rma start

rma stop

These commands enable/disable the RMA protocol support. Current process status can be seen using **config show** command.

rma clear command clears all RMA configuration

Automatic speed management (autobitrate)

Syntax:

rma IFname [-]autobitrate

Enables/disables an automatic speed management mode.

Traditional cases of InfiNet router's usage mean a fixed connection speed between a base station and a client. A client while searching and registering on a base station chooses an optimal, available in configuration, mode and afterwards does not change connection parameters even if radio channel characteristics (signal levels, noise level) are changing. The speed in both directions is set as it is configured on the client's side. If the conditions worsen so much that further work is impossible the connection breaks and a new base station search is run or new work parameters are being chosen. This way of configuration is logical when one needs to get a reliable channel with previously known characteristics and when all radio channel parameters are thoroughly designed.

However in some cases when there is a need in fast channel making (if it has non-critical parameters) it is more convinient to switch to an automatic mode. The automatic speed control mode solves this problem. In this mode every device controls the connection parametrs independently (amplitude of the received signal, number of ARQs on transmitting, erros, SNR on the opposite side etc) and chooses such transmittion speed which provides necessary conditions for a reliable work with minimum number of ARQs and losses. Speed values can be different for each direction but it will be optimal.

If this mode is turned on on the client's side the base station will turn it on also for this particular client. In this case the base station search is performed on the minimal for the client speed and then the speed is increased to its maximum value (it can be limited by the speed specified in rma bs command or by limitations for the specific device model). If autobitrate mode is turned on on the base station all client devices that work with it will automatically switch to this mode.

It is strongly recommended to turn on autobitrate mechanism on each CPE separately. This will increase system stability greatly.

It is important to remeber that turning autobitrate on can decrease overall network throughput because an average speed having the base station heavily loaded will long for the slowest client's speed.

WMA/CD (polling) regime management

Syntax:

rma if-name poll [MI MP MT]

rma if-name –poll

Enables/disables polling mode.

rma if-name poll stat [clear]

Shows protocol operation statistics

The WMA/CD (polling) regime is a method of accessing common radio channel under base station control, which consists in centralized distribution of transmission authorization markers by a base station to client units. This regime greatly improves operational stability and throughput of base stations under conditions of heavy load and signal level disbalance between different client units. It is particularly useful when client units are at long range from a base station and not in the direct visibility of each other, so that they cannot avoid mutual collisions in the radio channel by listening each other's transmission. The polling regime makes it possible to establish reliable communication between subscribers when the ordinary CSMA/CA wireless access method does not work at all.

Despite a slight decrease in the maximum transmission speed, the polling regime substantially increases the total throughput of a base station and provides for its fair distribution between client units. The polling algorithm is so designed as to minimize the protocol overhead while maintaining high efficiency and robustness.

Among the drawbacks of wireless polling method one can mention comparatively greater initial delay and greater response delay variations for series of short packets, which is adversely affecting such applications as telemetry and on-line games. Furthermore, two closely located WMA-operating base stations using the same or neighboring frequencies would make each other completely ineffective.

The polling regime is enabled on the base station only. Configuration of client units needs not to be modified. All those units, however, shall have RMA protocol enabled.

A base station with polling regime enabled may simultaneously serve both polling and non-polling client units; presence of non-polling units, however, substantially reduces the whole system's effectiveness.

For fine tuning of polling regime there are 4 optional parameters which can be set using the following syntax:

rma IFname poll [it=XX] [mi=XX] [ub=XX] [mt=XX]

- **IT** - input threshold number of packets per seconds. when reached a polling regime is turned on
- **MI** - Marker Interval. The primary time unit used in calculating the marker sending frequency. The approximate value is a half of the round-trip delay for the interface. Values are given in milliseconds, from 4 to 20.

- **MP** - Minimal priority (this parameter is obsolete, not used in new versions)
  The minimal priority level defined for inactive client units (higher the number, less the priority level). During the system operation, marker sending frequency dynamically varies for each client unit as a function of the unit's activity level. At each moment of time, the marker sending interval for a given client unit is defined as MI times current priority of the unit. When a client unit becomes inactive, its priority level gradually decreases, up to becoming equal to MP, and then starts increasing only when the unit manifests some packet-sending activity. The MP*MI product defines a maximal marker-sending interval for inactive client units; it shall be within 200 to 900 ms limits. By default, MP=60.
- **UB** - Upper bound. Marker sending interval upper bound. This value provides a minimal guaranteed marker sending frequency. This parameter is chosen based on the compromise between the total number of markers loading up the channel for no purpose and the responce time for the first key pressed in telnet program. The value is within 50 and 1000 ms. 800ms by default.
- **MT** - Marker Timeout. The maximum waiting time for a client unit response to a marker or data packet. Expressed in milliseconds; default value is 120 ms.

Do not modify default values of these parameters unless you really need it. If the command is executed without any parameters default values are used.

To evaluate effectiveness of the polling protocol operation, you can launch WMA statistics gathering regime by the **rma if-name poll stat** command. Information will then be displayed as on the example hereafter:

MI=9, MP=60, MT=120

md=143234, ma=123212, mt=36

pd=312762, pa=22343, pt=11

---

where:

- **md** - the number of markers, responded with a data packet
- **ma** - the number of markers, responded with a "no data" packet
- **mt** - the number of non-responded markers (timeouts)
- **pd** - the number of data packets, responded with a data packet
- **pa** - the number of data packets, responded with a "no data" packet
- **pt** - the number of non-responded data packets (timeouts)

In normal operation, the number of timeouts should not be greater than 5% of the total number of packets (markers and data packets) sent by the base station, i.e. the inequation

mt+pt < (md+ma+pd+pa)/20

should be permanently true. If the timeouts proportion becomes greater, you can try to gradually increase the value of the MT parameter in 20 ms steps, within reasonable limits.

When disabling the polling regime, the device enters the standard CSMA/CA regime, possibly with repeated registration of client units.

The polling regime cannot be enabled on a client unit.

---

3. Muffer command (Environment analyzer)

The muffer module is used to analyze the electromagnetic environment.

Syntax:

muffer [-tXX] [-lXX] IFname review [F1..F7] |sid|mac|mac2|mynet|scan

muffer stat [clear]

Description:

The **muffer** module makes it possible to rapidly test the electromagnetic environment, visually estimate the efficiency of the utilization of the air links, reveal sources of interference, and estimate their power.

Several operating regimes of the **muffer** module provide for different levels of details in test results.

On an 11Mb radio module, any test regime of the muffer module interrupts normal operation of the radio interface. Furthermore, only received packets are analyzed. Testing any external spread spectrum radiation sources is not yet implemented.

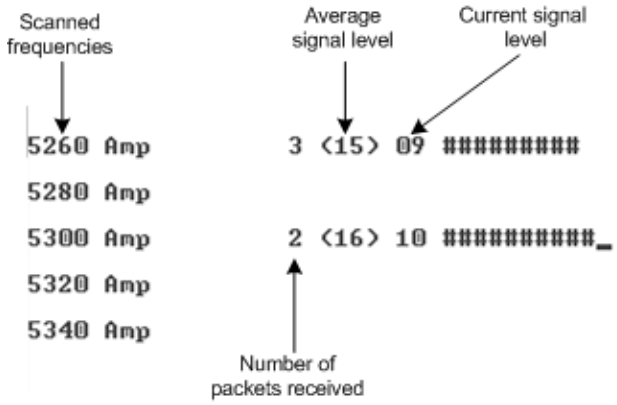
Review mode

This regime is enabled by the **review** option. It makes possible to have a general estimation of emissions and interference within specified frequency range. Up to 7 frequencies (separated by spaces) subject to analysis may be specified as parameters [F1..F7]. Only two last digits of the frequency values shall be given.

Normal operation of a radio module is impossible when this regime is enabled.

Example :

muffer rf4.0 review



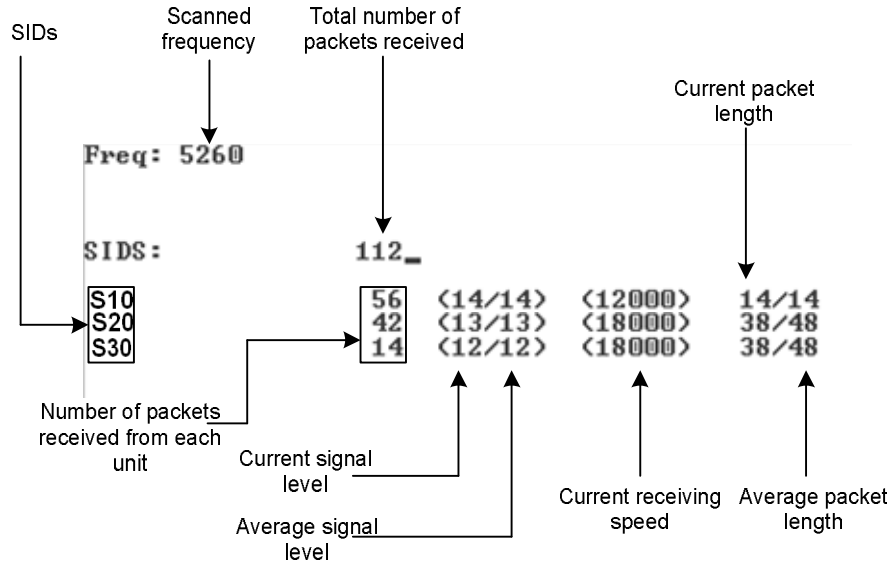
SID mode

The **sid** regime allows estimating the number of currently operating client groups having different identifiers (SID), and the efficiency of air links utilization. The analysis is carried out for all network identifiers at the frequency previously specified for the radio module by **rfconfig** command.

This regime disturbs normal operation of an 11 Mb radio module.

Example:

muffer rf4.0 sid



MAC|MAC2|MYNET modes

These regimes perform MAC-addresses analysis to estimate the number of clients with different MAC-addresses and the efficiency of their utilization of the air link. The analysis is carried out for all MAC-addresses at the frequency previously specified by rfconfig command. The **mac** regime checks only data packets, while in the **mac2** regime the link-level ACK messages sent by protocol support devices are also taken into account whenever possible.

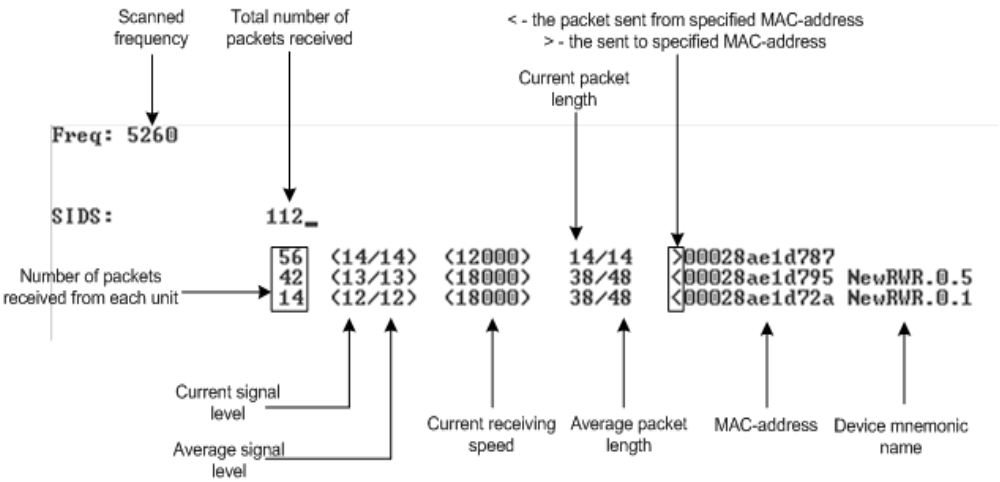
*This **mac** and **mac2** regimes disturb normal operation of an 11 Mb radio module.*

Finally, the **mynet** regime performs the radio testing without disturbing radio module's normal operation, but taking into account only packets from within the given network.

Example:

muffer rf4.0 mac2

The picture below shows the output **mac2** regime.



Scan mode

The scanning regime is enabled by a **muf scan** command, and provides for deep analysis of radio emission sources within the given network's territory. In this regime, the device scans the radio spectrum on all frequencies and for all modulation types.

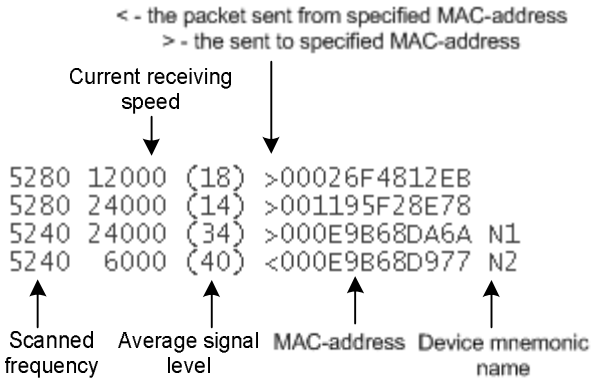
Information is displayed on any source of irregular (non-repetitive) radio signals.

To obtain information as complete as possible, the scanning process may take significant time.

Normal operation of the radio module is impossible when the scan regime is enabled.

Example:

muffer rf4.0 scan



Supplementary options for all the above regimes:

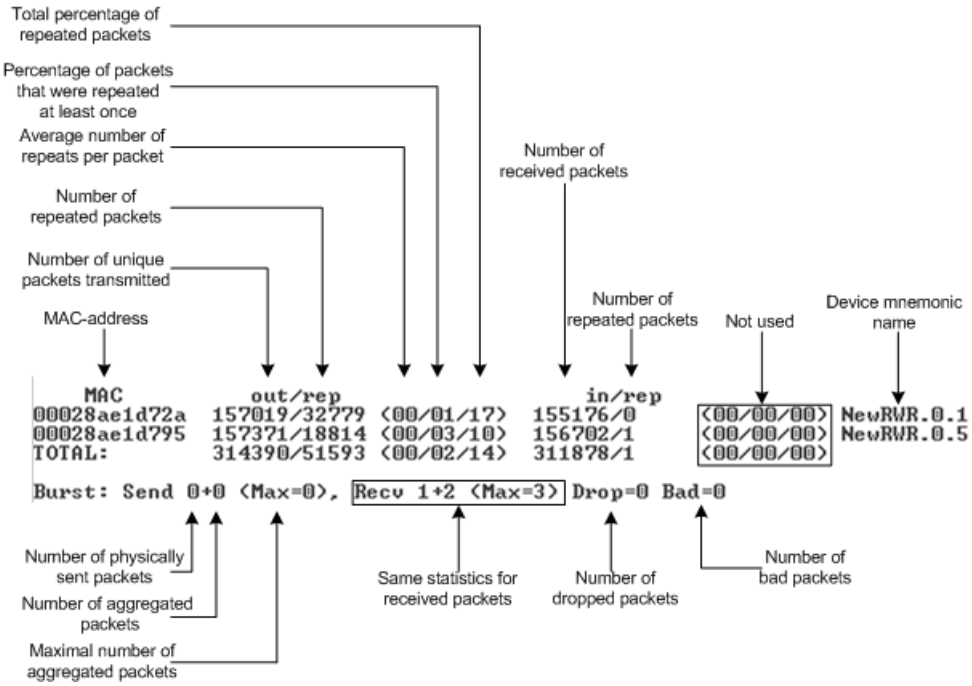
- **-tXX** specifies the duration of time, in seconds, for which the test regime is enabled (2 minutes by default). The value 0 in this field enables a test regime for illimited time.
- **-lXX** specifies the number of lines on the screen for displaying test results (24 lines by default)

To terminate the analyzer operation in any of the above regimes, press **<ESC>** or **<Ctrl/C>**.

Statistics module

The statistics gathering is used for estimating link load intensity and per client. The amount of packets sent and received, and the number of retransmissions is shown for each MAC-address participating in the data exchange.

The statistics output is presented in the picture below.



The following decisions can be made by analyzing the outputted parameters:

- If the number of repeated packets is comparable with total number of packets that means that you might have an interference source on the selected frequency. For normally operating link the percentage of repeated packets should not exceed 10%. It is extremely important to obtain a permanent zero value for the average number of repeats per packet. If the value is not zero that means that the link is NOT working properly and requires further improvement
- If total percentage of repeated packets and the percentage of packets that were repeated at least once are close to each other that might mean that you have got a permanent source of interference. Otherwise, it means that a strong interference source appears from time to time breaking your signal
- Concerning the fact that statistics module outputs the information for each MAC-address separately, you can reveal the problem for some specific unit on the wireless network

The "**muffer stat**" command shows the statistics only from registered devices.

To view **statistics** type the following command:

```
muffer stat
```

To reset all counters please type

```
muffer stat clear
```

4. Macf command (addresses mapping)

The command is used to map IP-addresses onto Ethernet MAC-addresses

Syntax:

macf MAC-address IP-address Comment

macf del N

macf [-]dhcp [-]strict | [-]reverse | [-]simple [-quiet]

macf show | clear

Description:

The **macf** command performs static mapping of IP-addresses to MAC-addresses in an Ethernet network. It may be useful for service providers when they connect to their network a group of clients (such as individual users in an appartment block) via one common access unit. In this case, clients may be tempted to change their IP-address to that of a neighbor, thus deceiving provider's accounting system. Although it is almost impossible to definitely resolve this issue, you can make however your life easier by directly mapping the client's assigned IP-address to his/her MAC-address, because surreptitiously modifying a MAC-address is much more difficult for an average user.

Every **macf** MAC-address IP-address command adds a new address pair to the address mapping table:

macf 102030405060 1.1.1.1 Room123

macf 203040506070 2.2.2.2 Room125

The comment parameter is a simple description string with no syntax restrictions.

The current state of the mapping table may be displayed by a **macf show** or **co show** command:

macf show

The output is the following:

macf 1 0020af915099 192.78.64.99 Server

macf 2 0020af9150a3 192.78.64.194 Room94

macf 3 0020af9150a4 192.78.64.134 Room57

macf 4 0020af9150a7 192.78.64.174 Admin

The second column in the above table contains automatically assigned internal numbers, which may be used to delete any specific line from the table by a **macf del N** command.

The **macf clear** command clears the mapping table altogether.

The mapping filter may operate in two different regimes:

In the **normal regime**, all client units whose address pairs are not explicitly specified in the mapping table are treated as usual, without any restrictions. This is default regime.

In the **strict regime** (which is enabled by **macf strict** and disabled by **macf -strict**), all packets received from units not described in the mapping table will be discarded.

In **quiet** mode logging into system log is disabled.

If you are remotely configuring a router using telnet, make sure, when enabling the strict regime, that your own workstation is already cited correctly in the mapping table. Otherwise you lose control over the router, and disabling the strict regime will be only possible through the router's diagnostics port.

In both regimes, when a packet is discarded by the filter, this fact is accompanied by a warning message on the screen and registered in **syslog**. To prevent an avalanche of faulty registration packets, only the first attempt to deceive the system from the group of similar ones is registered.

The MAC filter algorithm consists of two steps. In the normal mode (**default**):
1. First, the table is searched for the MAC-address of a packet being checked.

2. If the MAC-address is found, then the received IP-address is checked for correspondence with that found in the table.

The reverse mode (enabled by **reverse** option, and disabled by -reverse) swaps the above two steps:

1. First, the table is searched for the IP-address received

2. Then the MAC-address received is checked against that from the table.

In both regimes, the parameter with which the search starts is a search key and cannot appear in the table more than once.

If the **simple** option is enabled, only the first step of the above algorithm is executed. If the address searched for is found in the mapping table, then the packed is normally handled by the router. Otherwise, the packet will be discarded, regardless of whether the **strict** option is enabled or not (the second address is not checked).

With **dhcp** option enabled, macf filter is automatically supplemented with addresses issued by local DHCP server. These records are not stored in a permanent configuration and work until the given address is deleted by DHCP server.

Hereafter, some possible scenarios of using different filtering options:

1. Flat model. All workstations of the client's local network are directly connected to the Ethernet interface of the CU (client unit). In this case the simpliest filtering may be used, possibly with the strict option enabled:

macf MAC-address IP-address [strict]

2. If an intermediate router is installed between a client unit and the client's LAN, then **reverse strict** or **reverse simple** modes may be used, with IP-addresses of all authorized workstations being directly listed in the mapping table, while the MAC-address is always that of the intermediate router.

3. If several LANs are connected to the same client unit, each through an intermediate router, then the simple or reverse strict modes may become the most useful, with MAC-addresses of all those intermediate routers being listed in the mapping table.

5. Sppp command

PPP over synchronous links, maps E1 frames ports 0 (main), 1 (sub) onto network interfaces ppp0, ppp1.

Syntax:

sppp [-v] ifname [{enable/disable}] [options]

Description:

- **ifname** - name of network interface (ppp0, ppp1)
- **enable**/**disable** - attach or detach PPP-interface to E1 framer

SPPP protocol may work in the following modes:

- Cisco HDLC
- PPP (Point to Point Protocol according to RFC 1661)
- Frame Relay

Example:

sppp ppp0 enable mode=cisco

fconfig ppp0 10.0.0.1 10.0.0.2 up

where 10.0.0.1 - address of own PPP interface, 10.0.0.2 - remote side interface address

Options:

- **mode**

  Operation mode: **cisco**, **ppp**, **fr**.

- **keepalive**

  Used along with cisco and ppp modes. Value range: **on**, **off**.

  Enables/disables line state testing on the remote interface using control packets with 10 sec period. Makes sense only in cisco and ppp mode.

Example:

sppp *ppp0 enable mode=cisco* ***keepalive****=on*

sppp *ppp0* ***keepalive****=off*

sppp *ppp0* ***keepalive****=on*

PPP mode authorization options:

The below options are used only in ppp mode for selection of authentification protocol and setting of its parameters.

- **proto**

  Sets authentification protocol for both peers (myproto, hisproto). Allowed values: **none**, **pap**, **chap**.

  - **none** - disable authentification protocol;
  - **pap** - Password Authentification Protocol
  - **chap** - Challenge Handshake Authentification Protocol RFC 1994 with preliminary challenge coordination.

- **myproto**

  The same as proto, but only in case when far end requires authentification.

- **hisproto**

  The same as proto, but only in case when near end requires authentification.

- **myname**

  Sets username for authentification protocol (**chap**, **pap**).

- **mysecret**

  Sets password for authentification phase for near end.

- **hissecret**

  Sets password for authentification phase for far end.

Example:

sppp *ppp0 enable mode=****ppp*** *proto=****none***

Sets ppp mode without authentification. In the below example , ppp0 interface has authentification mode chap both for near end and far end and ppp1 interface has not authentification mode for far end.

sppp *ppp0 enable mode=**ppp** proto=**chap***

sppp *ppp0 myname=local mysecret=secret*

sppp *ppp0 hisname=remote hissecret=secret*

sppp *ppp1 enable mode=**ppp** myproto=chap hisproto=**none***

sppp *ppp1 myname=local mysecret=secret*

- **[no]rechallenge**

  Enables/disables repeated authentification mode during the operational phase. This feature is necessary for some incorrect chap protocol implementations.

  Example:

  sppp ppp0 norechallenge

6. Arp command (ARP protocol)

Implementation of Address Resolution Protocol.

Syntax:

arp view [IP]

arp add IP MAC|auto proxy

arp del IP|all [proxy]

arp [-]freeze

arp [-]proxyall [$ACL]

Description:

ARP protocol serves for IP to MAC-address mapping and vice versa. For example in Ethernet it allows to transform IP destination address into its 48-bit Ethernet address for packet forwarding over LAN.

In common case ARP works automatically making address resolution as it is necessary. But there are some cases when ARP tables should be corrected manually and **arp** command solves this problem.

The command has several forms:

arp view [IP]

Displays ARP records for IP-address. Displays all ARP records if address is not specified.

arp add IP MAC [proxy]

arp add IP auto proxy

Adds record into the ARP table. MAC-address mapped to IP-address. If keyword proxy specified then the system will announce this information as response to the requests from other stations, acting as proxy ARP server - even if this IP-address is not system own address. In this case instead of MAC-address one may specify auto keyword.

arp del IP | all [proxy]

Deletes the record for IP. Or all records in the system if keyword all specified. If optional parameter proxy specified, then only proxy IP-addresses will be deleted.

arp [-]freeze

Enables to freeze ARP table. No more automatically updates allowed. The command fixes only manual records and does not affect on the radiointerface with active protocol RMA or MINT. Be careful when entering this command via telnet.

arp [-]proxyall [$ACL]

In this mode the system will reply on all ARP requests, if respective IP target address resides in the routing tables and reachable via interface other than source MAC-address. I.e. if there is a route to the target IP then the system can be considered as a gateway.

In **proxyall** mode, you can specify an **$ACL** list of addresses/networks which limits replied ARP requests of the command with networks and addresses of the **$ACL** list.

Example:

arp add 10.10.10.10 00:11:22:33:44:55

arp add 192.168.5.1 5544332211 proxy

IV.Layer 3 Command Set – IP Networking

1. Ifconfig command (interfaces configuration)

The command is used to set and view configuration of network interfaces.

Syntax:

ifc*onfig interface[* ***inet*** *ADDRESS] [[* ***-*** *]* ***alias*** *]*

ifc*onfig interface [* ***up|down*** *] [* ***mtu*** *N] [* ***link0*** *|* ***link1*** *|* ***link2*** *]*

ifc*onfig interface media mediatype*

ifc*onfig vlan X vlan TAG [-] vlandev IFPARENT*

ifc*onfig -a*

Description:

This command allows setting and viewing the configuration of interfaces specified by their ID numbers.

The command has the following parameters and flags:

- **interface:** specifies the name of an interface (to see all interface names, an **ifconfig -a** or **netstat -i** command may be executed).
- **address:** specifies the IP-address assigned to the interface. May be specified as:
    - address/number of bits in the mask
    - address:mask
    - address proper

Example:

ifconfig eth0 inet 192.168.1.1/26

ifconfig eth0 inet 192.168.1.1:255.255.255.192

ifconfig eth0 inet 192.168.1.1

N: sets the desirable MTU (Maximum Transfer Unit) size for the interface. Usually the value of this parameter does not need to be changed, but in some cases decreasing the MTU value facilitates improving the work condition for a client with very low signal. In addition, it can be used to vary parameters of tunnel interfaces.

up|**down :** flags enabling/disabling the interface.

System limitations:

lo0 interface cannot be set to **down** state. Radiointerfaces states are not saved in the configuration (after rebooting all interfaces are in the **up** state)

Example:

ifconfig eth0 up

ifconfig eth0 1.1.1.1/24 up

ifconfig rf4.0 down

link0, link1, link2 flags specify the type of physical Ethernet transfer media:

link0: the type of the physical interface is AUI

link1: selects the BNC interface

link2: selects the UTP interface.

Example :

ifconfig eth0 inet 192.168.1.1 link0 up

If no flags are specified when configuring an Ethernet interface, then the physical interface is automatically selected between BNC and UTP.

To disable the flag previously set, an ifconfig command is executed with the same flag preceded by "-".

Example:

ifconfig eth0 -link0

Media parameter allows specifying physical interface eth0 10/100 properties.

Allowed values:

100BaseTX-fullduplex, 100BaseTX-halfduplex, 10BaseT-fullduplex, 10BaseT-halfduplex, auto

By default: **auto**

For **vlanX** (VLAN IEEE 802.1q) configuration one should use **vlan** and **vlandev** options in **ifconfig** command.

Vlan parameter sets VLAN tag for the current interface (1-4094). **Vlandev** parameter creates a connection with a physical interface which serves the media – eth0 in this case.

Example:

ifconfig vlan1 1.1.1.1/24 vlan 5 vlandev eth0 up

or

ifconfig vlan1 1.1.1.1/24 up

ifconfig vlan1 vlan 5 vlandev eth0

ifconfig -vlandev eth0

(last line in the example cancels the connection between vlan1 logical interface and physical device **eth**0)

Both additional parameters of **vlanX** interface should be entered in one line as it is shown in the example, and if needed one can add a new IP-address setup. For the normal vlanX interface functioning, a physical interface eth0 should be in the active state (**up** flag).

mtu N parameter sets a desired MTU (Maximum Transfer Unit) for the interface. Usually there is no need to change this parameter, but in some cases decreasing MTU parameter provides with more reliable work of CPEs with extremely low signal level. Moreover, it can be used to change parameters for tunnel interfaces.

alias: this flag indicates that several IP-addresses are assigned to one interface. Each new IP-address assigned to an interface (except the first, called primary) is considered an alias address and shall have the alias option set.

For example, after executing the commands:

ifconfig eth0 inet 193.124.189.1/27 up

ifconfig eth0 inet 10.0.0.1 alias

there will be two addresses from two different networks assigned to the same **eth0** interface.

To remove any address from an interface, an ifconfig command is executed with the **-alias** option following the address to be removed.

Example:

ifconfig eth0 inet 193.124.189.1/27 -alias

The **[-]alias** option may be put in any **ifconfig** command, that is, all addresses assigned to an interface are considered as equivalent aliases. If the first (primary) address is removed, the next (in the order of their assignments) becomes primary.

To display the current configuration of an interface, an **ifconfig** command may be executed with the interface name as the only parameter.

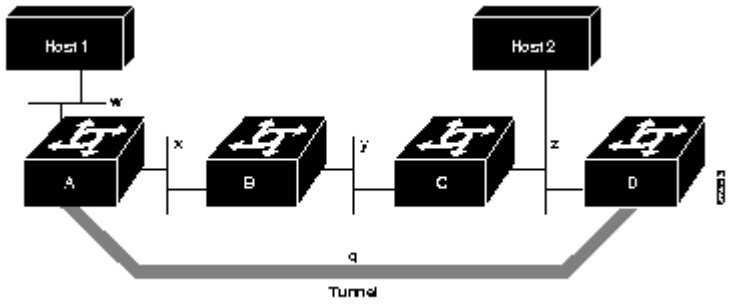
To see the configuration of all interfaces of the router, use the **ifconfig -a** command.

2. Tun command (tunnels building)

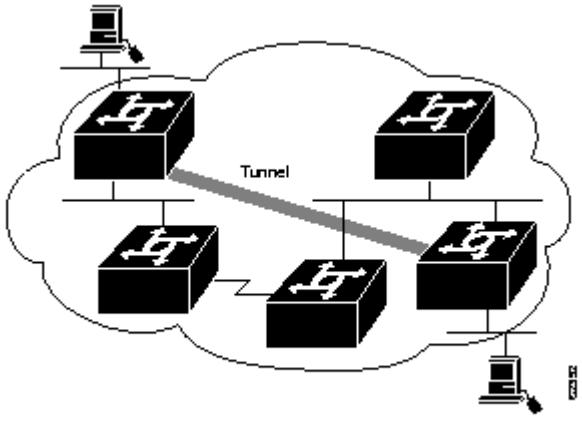
The command specifies the parameters of a software tunnel.

General Description:

Tunnels are used to merge two remote and physically not connected networks into one logical structure. Tunnels are widely used to create corporate networks or the so-called virtual private networks (VPN): several remote offices, connected to the network through the same or different providers, are connected to the company headquarters or to each other by tunnels, thus forming one corporate structure. Common IP address space and registration/accounting policy can be used throughout the whole VPN-based corporate network, independently of network provider(s) used.



Tunnels also solve the problem of using common transport media in a public network so that different clients could be provided with services by several providers. It means that a client can be connected by a tunnel to a specific provider, to be serviced by that provider, irrespective of the client's connection point to a common transport network.



There are several approaches to build tunnels. One of these, IP into IP Encapsulation (described in RFC 2003), is implemented in OS WANFleX. This technology is used, for example, in Cisco Systems routers, and is a subset of the IPSEC protocol supported by several operating systems.

Within this approach, tunnels are implemented as point-to-point (P2P) links between two endpoint routers. The whole data stream through such a link is encapsulated into IP packets at one end of a link and is delivered to its opposite end through the existing transport network.

Four parameters are necessary to configure a tunnel:

1. The internal IP-address of the local end of the P2P link.

2. The internal IP-address of the remote end of the P2P link.

3. Real source IP-address to be specified in the outgoing packets.

4. Real destination IP-address to be specified in the outgoing packets.

Internal IP-addresses of both ends of a P2P link are set using **ifconfig** command; all other parameters are specified by the **tun** command (see example below).

Syntax:

tun N src IP-addr dst IP-addr [mtu M] [clear].

Description:

Assigns the source (**src**) and destination (**dst**) real IP-addresses to a tunnel specified by its logical number **N** which has been created by an **ifconfig** command.

Outgoing packets are encapsulated into IP datagrams and sent to the **dst** address. The **src** address is inserted into the datagram as source address.

*The **dst** address shall also be attainable through an interface of the router different from that used to access the tunnel. This can be done, for example, by using explicit static routing (the route add command), or by prohibiting importation of some of the RIP protocol route descriptors arriving to that interface. If this condition is not satisfied, a looping may happen, when already encapsulated packets come back to the tunnel entrance, and so on, causing system overload. The system watches over such situations, and when discovering a loop, drops erroneous packets and writes a message **tunX: looping ...** into the system log.*

The **src** address must be a real IP-address for one of the router's interfaces; for the same reason, it shall be attainable from the router at the tunnel's remote end through the existing network (and not only through this tunnel).

On the remote site of the tunnel, the **src** and **dst** addresses swap their roles.

The **mtu** optional parameter allows the user to set the Maximum Transfer Unit size for packets going through the tunnel. Default value is 1480 bytes.

Disabling the tunnel number **N** may be done by executing the command:

tun N clear

Example:

ifconfig tun0 1.1.1.1 1.1.1.2

tun 0 src 195.23.23.23 dst 194.34.34.34

Here, the **ifconfig** command defines internal IP-addresses for both ends of a **tunnel #0** as addresses for an interface denoted as **tun0**; then, the tun command defines real IP-addresses for the **tunnel #0** extremities.

At the opposite side of the tunnel this would look as follows:

```
ifconfig tun0 1.1.1.2 1.1.1.1
tun 0 src 194.34.34.34 dst 195.23.23.23
```

If you use a Cisco Systems router at the remote end, you may configure it as follows:

```
interface Tunnel0
  ip address 1.1.1.2 255.255.255.252
  tunnel source 194.34.34.34
  tunnel destination 195.23.23.23
  tunnel mode ipip
!
```

3. Qm command (QoS configuration)

The command manages the "Quality-of-Service" (QoS) parameters

General description:

QoS manager is a convenient and flexible mechanism to manipulate data streams going through the router. The user can create up to 200 logical channels characterized by different properties (such as priority levels and data transfer rates), and then assign data streams to these logical channels according to special rules of assignment. Packets going through different channels are thus modifying their own properties as well as properties of their respective data flows.

With further development of our system, new properties would be added to the list of different properties characterizing a channel.

Syntax:

```
qm option {[-]voice [-]dot1p [-]tos [-]strict}
qm classL if=ifname | max=N
qm chN [max=X] [pri=P] [pps=T] [to=addr] [vlan=N|-1|-2] [dscp=N|-1|-2]
       [dot1p=N|-1|-2] [classL]
qm chN clear
qm stat [clear]
qm add [ifname] chN rule...
qm del R
qm mov R S
```

where

- **N,L,X,P,R,S,T** are integers;
- **addr** is an IP-address;
- **rule** is a packet filtering rule with the same syntax as in the **ipfw** command.

Parameter values shall be put after their keywords (if any) without blanks, as shown above; no blank may be put before or after "=" sign.

Description:

*qm classL if=ifname | **max**=N*

This command creates a service class **#L** assigning it to an interface specified by ifname. It is used for dynamic bandwidth allocation between different channels on the same interface. If the option "**max = N"** is used the total bandwidth of the class will be limited to a given value (thousands bps), otherwise for the total bandwidth limiting the present interface physical speed will be used.

To disable this parameter type:

qm classL if = ifname max = 0

qm chN [max=X] [pri=P] [to=addr]] [vlan=N|-1|-2] [dscp=N|-1|-2] [dot1p=N|-1|-2] [classL]

This command defines a logical channel #N (N=1…200)with properties specified by one or more command options as follows:

- **max=X** sets maximum data rate for the channel in Kbit/s. Value range: from 10 to 10000. If set to 0, cancels any speed limitation for the channel.
- **classL** assigns service class #L to the channel. This additional parameter relates to the above defined data rate limitation, making it flexible: when the total bandwidth of the interface having this service class is not fully used, the extra bandwidth may be granted to such channel, thus exceeding its predefined data rate limit, up to full load of the interface. When, however, there are several such channels competing for extra bandwidth, it is divided between them proportionally to their respective predefined speed limits. (See examples below)
- **pri=P** Sets priority level of the specified channel (0..16). Smaller values correspond to greater priority levels.
- **pps=T** Sets the limit for the packets per second for the specified channel
- **to=addr** redirects the whole stream to the specified IP-address irrespectively of the present routing conditions. The specified address shall be directly attainable through one of the router interfaces (without additional routing). This may be useful when the router serves as a network access unit, and two or more different clients want to access different providers through one unit.
- **vlan=N**, **dot1p=N**, **dscp=N** manipulates DSCP and/or 802.1p labels. Value "-2" deletes parameter from command line, value "-1" disables the field:
    - DSCP (valid values are 0-63) sets to 0 (zero).
    - 802.1p priority (valid values are 0-7) sets to 0 (zero) and, if VLAN ID isn't introduced, is deleted with VLAN header.
    - VLAN ID (valid values are 0-4095) is deleted with VLAN header regardless of 802.1p priority.

If several of the above parameters are specified in the same command, then speed limitation is applied first, then redirection, and only then priority.

qm chN clear

Cancels the **N**-th logical channel current specification, making its number free for another specification.

qm add [ifname] chN rule ...

Specifies one or more rules for accepting packets at the channel **#N**. Rules are specified using the same syntax as in the **ipfw** command.

Optional **ifname** parameter specifies the router's interface through which a packet shall arrive for being accepted at the specified channel.

All rules specified on a router constitute a numbered list; a rule is added at the end of this common list at the moment when it is specified for some channel, and then may be moved to another position (see below). To display the list of all rules with their numbers, use the config show command.

Each packet arriving to the router is checked against the set of rules in the order of their enumeration, until a rule is found which the packet satisfies, or until the end of the list of rules is encountered. Once such a rule is found, the packet is directed to the channel corresponding to the rule, without checking it against the remaining rules in the list. Therefore, the order of rules is very important for correct dispatching of packets among channels.

Optional "**pass**" parameter allows a packet to pass a rule executing the related actions of this rule and continue with other rules in the list.

qm stat command displays statistics of the specific channel (only for channels with speed limitation).

qm del R

This command deletes the R-th rule from the list.

qm mov R S

Moves the **R**-th rule to the **S**-th position.

Transparent packets prioritization is supported in MINT network. It is performed by using channels management in "qm" command. Administrator can put streams into different channels based on "qm/ipfw" rules as well as "tos" and "dscp" fields.

qm ch1 pri=12

qm add ch1 all from x/x to y/y

qm add ch1 dscp31 all from a to b

qm add ch1 dscp42

Each channel can be assigned a priority (0..16). Once assigned, a priority will be automatically recognized by every node inside MINT network. Priority scheme looks as follows:

QM_PRIO_NETCRIT	0
QM_PRIO_VOICE	1
QM_PRIO_RT1	2
QM_PRIO_VIDEO	3
QM_PRIO_RT2	4
QM_PRIO_QOS1	5
QM_PRIO_QOS2	6
QM_PRIO_QOS3	7
QM_PRIO_QOS4	8

QM_PRIO_BUSINESS1 9

QM_PRIO_BUSINESS2 10

QM_PRIO_BUSINESS3 11

QM_PRIO_BUSINESS4 12

QM_PRIO_BUSINESS5 13

QM_PRIO_BUSINESS6 14

QM_PRIO_BUSINESS7 15

QM_PRIO_BUSINESS8 16

Priorities "1" and "2" are additionally processed as "voice". Packets from which the priority is not clearly defined will be sent via common queue with "Best Effort".

The "**qm option**" allows automatic prioritization management of data flows in the device. Command options **[-]voice [-]dot1p [-]tos** enable/disable automatic prioritization of voice packets, packet labeled with IEEE 802.1p priority (below is a compliance scheme of MINT and IEEE 802.1p priorities), packets labeled with TOS. The **[-]strict** option means that "Strict Priority" policy is applied to all queues, otherwise (by default) "Weighted Fair Queuing" policy is used. "Strict Priority" policy is when packets from queue with lower priority are not processed before queue with higher priority is not empty. "Weighted Fair Queuing" policy is when even if higher priority queue is not empty packets from other queues will be processed in a distinct sequence relative to a higher priority queue. For example, 4 package from queue with priority 1 - 1 package from the queue with priority 2, 8 packages from queue priority 1 - 1 package from the queue with priority 3.

Compliance scheme of MINT and IEEE 802.1p priorities:

MINT	IEEE 802.1p
QM_PRIO_BUSINESS8	0 BE Best Effort
no priority	1 BK Background
no priority	2 Spare
QM_PRIO_BUSINESS1	3 EE Excellent Effort
QM_PRIO_QOS3	4 CL Controlled Load
QM_PRIO_VIDEO	5 VI Video
QM_PRIO_VOICE	6 VO Voice
QM_PRIO_NETCRIT	7 NC Network Control

For example, the unit is configured automatically prioritize packets labeled with IEEE 802.1p priority. The node receives a package labeled with IEEE 802.1p priority, "6 VO Voice". The node will assign him "QM_PRIO_VOICE" priority and in accordance with the priorities scheme, this package will be processed before packets with other priorities.

*Attention: Real prioritization within MINT network is conducted by priority, given by the option **pri=N**.*

DSCP label is transparently transmitted through MINT in any of its modes.

802.1p priority is transparently transmitted only in switch MINT mode.

*If necessary, when leaving MINT network **dot1p** and **dscp** parameters can be assigned by the operator.*

QoS Manager allows enough flexibility for prioritizing and remapping traffic (see examples below).

Examples:

qm ch1 max=64

qm add eth0 ch1 all from 0/0 to 0/0

When used on a client unit, sets the data rate for outgoing traffic at 64 Kbit/s limit.

qm ch1 pri=10 qm add ch1 all from 1.1.1.0/24 to 0/0

qm add ch1 all from 0/0 to 1.1.1.0/24

Establishes for the traffic from or to 1.1.1.0/24 network the highest priority over all other data flows.

qm ch1 pri=10

qm ch2 pri=20

qm add ch2 all from 1.1.1.0/24 to 0/0

qm add ch2 all from 0/0 to 1.1.1.0/24

qm add ch1 all from 0.0 to 0/0

The 1.1.1.0/24 network traffic will have the lowest priority as compared to other data flows. *Please note the order of rules in the above list. The last rule, which is satisfied by any packet, may only be at the end of the list.*

qm ch1 to=10.10.10.10

qm ch2 to=20.20.20.20

qm add ch1 all from 1.1.1.0/24 to 0/0

qm add ch2 all from 2.2.2.0/24 to 0/0

Subscribers of the 1.1.1.0/24 network will be serviced by the 10.10.10.10 provider, while the 2.2.2.0/24 subscribers will use the 20.20.20.20 provider.

In more complicated situations, when the routers of service providers are not directly accessible from the given node, one would better start with defining tunnels to those providers, and then redirect traffic to those tunnels.

```
qm option –voice tos
```

This command disables voice automatic prioritization and enables tos automatic prioritization.

Example of traffic prioritization and remapping:

Channel 1 disables DSCP labels and 802.1p priorities

qm ch1 dscp=0 dot1p=-1

Channel 2 sets flow priority QM_PRIO_BUSINESS1 and DSCP label 31

qm ch2 pri=9 dscp=31

Channel 3 sets flow priority QM_PRIO_VIDEO and DSCP label 11

qm ch2 pri=3 dscp=11

Channel 4 sets flow priority QM_PRIO_BUSINESS8 and DSCP label 51

qm ch4 pri=16 dscp=51

All the traffic is coming through channel 1 for setting all priorities to null

qm add ch1 pass all from 0/0 to 0/0

Some traffic is setting into channel 2

qm add ch2 tcp from X.X.X.0/24 to 0/0

Another part of traffic is setting into channel 3

qm add ch3 udp from X.X.X.0/24 PORT to 0/0

Other traffic will be processed as non-priority traffic or can be appointed with some priority by setting into channel 4

qm add ch4 all from 0/0 to 0/0

Channel 25 sets 802.1p packet priority. If there is no VLAN heading it will be added automatically.

qm ch25 dot1p=5

Channel 26 sets 802.1p priority and VLAN ID. If there is no VLAN heading it will be added automatically.

qm ch26 vlan=7 dot1p=4

Packets which are coming from MINT network through eth0 interface and having DSCP label 11 is put into channel 25.

qm addout eth0 ch25 dscp11 from 0/0 to 0/0

Packets which are coming from MINT network through eth0 interface and having DSCP label 13 is put into channel 26.

qm addout eth0 ch26 dscp13 from 0/0 to 0/0

4. Route command (static routes configuration)

The command is used to configure static routing tables.

Syntax:

route add address|default gateway

route delete address [gateway]

Description:

This command provides with manual management of system routing tables. In the normal mode, when a routing daemon is active, this command is not needed. However, in some cases it allows to achieve more precise, non-standard configuration.

Parameters:

- **add**: adds a route to a table
- **delete**: deletes a route from a table
- **address:** destination network IP-address or host address. The parameter can be specified in the following formats: network-address/mask length, or network-address:mask, or network-address.

- **gateway:** IP-address of the router through which the address is attainable.

It is possible to use the keyword **default** instead of explicitly specifying the 0/0 IP-address.

Examples:

route add default 195.38.44.129

route add 193.124.189.0/27 195.38.44.108

route add 193.124.189.0:255.255.255.224 195.38.44.108

All routes that are described using route add command are "pseudostatic". It means that this information will be immediately placed into the configuration and will be active until it is deleted using route delete command. However, actually described routes will be put into the system tables only when there is an interface with an address and a mask within the boundaries of the gateway address set. When this address is absent routes set will be automatically deleted from system tables but still will be present in the configuration.

5. Rip command (RIP-1 and RIP-2 configuration)

The command is used to configure the router for using RIP-1 and RIP-2 protocols

Syntax:

rip start|stop|restart|flush|[-]trace[LEVEL]|dump|show|[-]ridhosts|[-]keepstatic

rip IFNAME v1 v2 [-]in [-]out [-]v1in [-]v1out [-]v2in [-]v2out [-]ag [-]subag

rip IFNAME peer ADDR[/MASK | /MASKLEN] ... | del

rip [-]static NET[/MASK] GATEWAY

Export/import filters:

rip [INTERFACE] [no]export|[no]import NET[/MASK|/MASKLEN] [exact]|all| default [[+|-]metric N] [pref N] ...

rip [INTERFACE] [no]export |[no]import NET[/MASK]|[/MASKLEN]del

The routing module supports two versions of the Routing Information Protocol: RIP-1 and RIP-2. The **rip** command is used to set up the module for using these routing protocols.

Description:

rip start

rip stop

rip restart

Starts, terminates, restarts operation in RIP mode. To save the current state of the routing module in the flash memory, use the **config save** command.

rip flush

Flushes all import and export filters.

rip [-]trace [LEVEL]

Switches the RIP operation into the trace mode. The optional LEVEL parameter specifies the detalization level of debug information.

Allowed values are as follows:

1. minimum level of events tracing

2. tracing of received/sent packets

3. tracing of received/sent packets and their contents

4. tracing of changes in the kernel routing tables.

Default value for this parameter is 4.

rip dump

Displays the status of the routing module's internal routing tables and interfaces.

rip show

Displays current RIP setup parameters

rip [-]ridhosts

Prohibits the export of addresses of point-to-point local interfaces, when there is a network route going to the same network via the same interface. This is a specific case of aggregation and allows limiting the number of exported routes.

rip [-]keepstatic

Keeps saved static routes as default routes. Sometimes, it is useful to specify some routes statically using the route add command. This allows configuring the router for the "warm start" mode: the router starts operation immediately after switching power on, when dynamic routing tables have not been built yet. Typically, these static routes are overridden by dynamic routes built by RIP. The present command allows keeping static routes in the routing tables as by default routes even after starting RIP operation, when other sources of routing information become available. Exported metrics value of such routes will be equal to 1. If another value is needed, an appropriate export rule shall be specified by the **rip export** command.

After disabling this mode by a **rip -keepstatic** command, those static routes will be replaced by dynamic routes.

rip IFNAME v1 v2 [-]in [-]out [-]v1in [-]v1out [-]v2in [-]v2out

Group of options managing protocol version.

Allows specifying protocol versions used for import and for export, for each interface separately. By default, **RIP2** is enabled for import and export, and **RIP1** is fully disabled (rip IFNAME v2 -v1).

rip IFNAME peer ADDR[/MASK] ... | clear

Using this filter one can limit nodes number through which routing information is being exchanged via IFNAME interface. As a limiting parameter a range of addresses is set within which possible partners can be allocated. Routing information will be sent only though those interfaces which addresses correspond with range set. Received information will be filtered if source address does not match in the defined range.

Example:

rip rf4.0 peer 10.1.2.3 10.4.5.6 192.168.1.0/16

Routing information exchange is limited by 10.1.2.3, 10.4.5.6 nodes and everyone who matches 192.168.1.0/16 range.

rip IFNAME [-]ag [-]subag

Enables/disables aggregation of routing information.The command allows significantly decreasing the volume of routing information transmitted via the network. By default, this regime is disabled.

When **subag** option is enabled, the routing module tries to assemble (aggregate) in a bigger block several blocks of routing information pertaining to different small subnetworks and arriving from different sources to the same router interface.

When **ag** option is enabled, the same is done for class C natural networks.

Aggregation is extremely useful in routing nodes located between two independent parts of your network, or at the border with an external network. For example, a client's router on the border between the client's LAN and a provider network can use aggregation, if the client has been assigned a whole IP address block for his subnetworks.

You should be careful with the route aggregation. For example, it is better to avoid using it in ring networks where not all of the hosts support this mode or static routing is used on some of them. In this case, it may happen that one group of subnetworks will appear aggregated in one routing path, and separated in another. Naturally, when choosing the route, the shortest path will be used, which is not always correct.

rip [-]static NET[/MASK] GATEWAY [metric XX]

This command is eliminated starting from 3.31 version. One must use route add command instead. All rip static commands that exist in the configuration are automatically transformed to route add commands.

Export/import filters:

Enabling export/import filters allows the system administrator to limit the distribution of the routing information and force some changes in the properties of routes built.

Filters are combined in groups, each group consisting of 4 tables (Export, NoExport, Import, NoImport).There are 3 different groups of filters:

Filters for specific interface addresses. The INTERFACE parameter is of the form "int:ADDR". In this way it is possible to define a filter for any specific interface address, if there are several addresses (aliases) assigned to the interface.

Example:

rip int:10.2.3.4 export all

Filters for the whole interface. The INTERFACE parameter contains then the interface name. A filter applies to the whole interface, that is, to all its alias addresses.

Example:

rip eth0 export all

Filters for the protocol as a whole. The INTERFACE parameter is then not used, as such a filter applies to the whole protocol, on all interfaces.

Example:

rip export all

Export and NoExport tables list the networks that, respectively, must or must not be exported from the router.

Similarly, Import and NoImport tables list the networks that, respectively, must or must not be imported to the internal tables of the router.

When specifying filters, the following is to be noticed:

- Filters are ordered from less to more general. First are considered filters related to specific interface addresses; then, to specific interfaces; and then the general ones. Individual rules in the tables are ordered according to the same principle: from the smallest networks to the biggest ones, from the more detailed information to the more general.
- By default (when no filters are specified), all routes with their natural metrics are imported and exported.
- If at least one prohibiting (NOIMPORT/NOEXPORT) filter is enabled, all the rest is assumed permitted. If at least one permitting (IMPORT/EXPORT) filter is enabled, all the rest is assumed prohibited. Therefore, if you have started with a permitting filter,you must continue with permitting filters up to the end. Conversely, if you have prohibited something, only that particular thing will be prohibited.
- If for a given network a permitting and a prohibiting filter are simultaneously enabled, then the prohibiting filter will take priority.
- Only filters of the most prioritary group applicable to a particular network are effectively applied to that network.

The filter definition syntax is quite simple; it is better understood using examples hereafter.

Examples:

Permitting export of all routing information, except a narrow network and the default route:

rip noexport 192.168.9.0/24

rip noexport default

rip export all

Permitting the import of all routing information, except that of private networks from the address block 10.0.0.0/8:

rip noimport 10.0.0.0/255.0.0.0

rip import all

Several networks may be specified on the same command line:

rip noexport 192.168.9.0/24 192.168.10.0/24 192.168.20.0/24

A route metrics value may be explicitly specified in exporting/importing filters:

rip import 192.168.9.0/24 metric 5

rip export 192.168.9.0/24 metric 7

Furthermore, you can specify relative metrics change for a route when it traverses the given node. In the following example, original metrics values of all routes will be incremented by 2:

rip export all +metric 2

Example of decrementing metrics values:

rip import default -metric 1

When specifying relative metrics change, the resulting value shall never become less than 2 or more than13.

Normally, a network address specified in a filter defines the correspondent network with all its subnetworks (corresponding to longer masks). If however a filter needs precise address value, the exact keyword shall be used.

For example, the filter:

```
rip noimport 10.0.0.0/255.0.0.0
```

prohibits the import of the 10.0.0.0 network and of any its subnetwork (10.XXX.XXX.XXX), while the filter:

```
rip noimport 10.0.0.0/255.0.0.0 exact
```

prohibits the import of the 10.0.0.0 network only (the import of the subnetworks is permitted).

To delete a filter, the **del** keyword is added after specifying the addressing information:

```
rip noexport 192.168.9.0/24 del
```

6. OSPFv2 (dynamic routing protocol module)

Getting started

OSPF protocol is widely used routing protocol for IP networks. Basic principles that form a current version of protocol are outlined in RFC 2328. OSPF protocol is a classical Link-State protocol which delivers the following functionality:

- no limitation for the network size
- routes information update sending using multicast addresses
- high speed route definition
- using authentication procedure while routes updating
- classless routing support

Command language. Basic principles

OSPF has its own command shell (CS). To enter the shell, execute the following command:

```
#1> ospf
OSPF>
```

Commands entered in CS are not case-sensitive and can be shortened until ambiguity appears. To get a quick hint you can press "?" at any time:

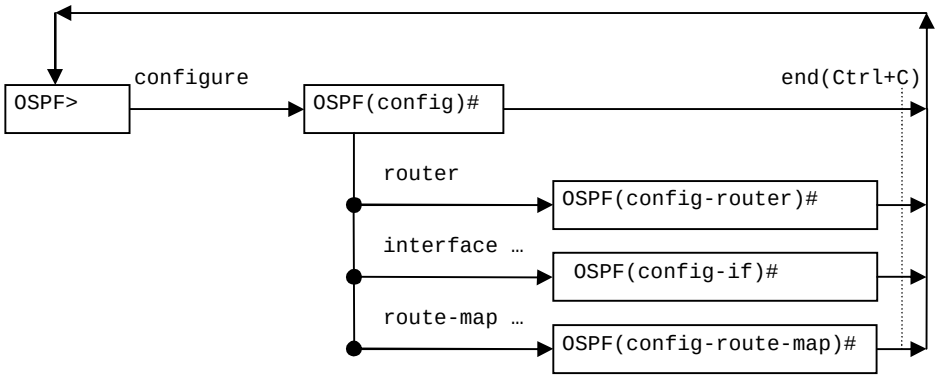
```
OSPF>?
  configure  Configuration from vty interface
  end        End current mode and change to root mode (CTRL+C).
  exit       Back to WANFlex command shell (CTRL+D).
  help       Print command list
  show       Show running system information
OSPF>
```

CS can work in different modes. Current mode is displayed along with command prefix as "OSPF(mode)#". For example, if **configure** command is entered, CS switches to *config* mode:

```
OSPF> configure
```

OSPF(config)#

The following figure shows the transition scheme between different modes of CS.



Every mode has its own set of commands. The following commands are available in any mode:

- Help – prints the list of commands for the current mode
- End – goes back from the current mode to the base mode
- Exit – exit to WANFlex CLI from OSPF CS

At the start, CS is in the base mode which has a set of commands to view current router state. In order to switch to the configuration mode you should have *superuser* rights. After entering a configuration mode, the configuration is being blocked and entering in this mode from other terminal (e.g. other telnet session) is prohibited. In order to avoid a "dead" block of the session, CS automatically quits the configuration mode after five minutes of no activity.

Context help is always available using "?". For example:

```
OSPF> config
OSPF(config)#?
  access-list  Add an access list entry
  clear        Reset functions
  end          End current mode and change to root mode (CTRL+C).
  exit         Back to WANFlex command shell (CTRL+D).
  help         Print command list
  interface    Select an interface to configure
  no           Negate a command or set its defaults
  prefix-list  Build a prefix list
  route-map    Create route-map or enter route-map command mode
  router       Enable a routing process
  show         Show running system information
  stop         stop
OSPF(config)# interface?
  IFNAME  Interface's name
OSPF(config)# interface eth0
OSPF(config-if)#?
  authentication       Enable authentication on this interface
  authentication-key   Authentication password (key)
  cost                 Interface cost
  dead-interval        Interval after which a neighbor is declared dead
```

```
  description          Interface specific description
  end                  End current mode and change to root mode (CTRL+C).
  exit                 Back to WANFlex command shell (CTRL+D).
  hello-interval       Time between HELLO packets
  help                 Print command list
  message-digest-key   Message digest authentication password (key)
  network              Network type
  no                   Negate a command or set its defaults
  priority             Router priority

  retransmit-interval  Time between retransmitting lost link state
  show                 Show running system information
  transmit-delay       Link state transmit delay
OSPF(config-if)#
```

After quitting CS using "exit" command (or Ctrl+D), CS stays in the last active mode.

Commands may have different parameters. Commands parameters are specified in several formats. Parameter's format is described in the context help or in the list of commands (**help** command) in the following way:

- A.B.C.D – a parameter is set in IP-address format. Example: 192.168.0.15
- WORD – a set of characters with no spaces
- <1-N> - a parameter is set as a decimal number in a range from 1 to N
- A.B.C.D/M – a parameter is set in a format IP-address/subnet mask length. Example: 192.168.0.0/24
- IFNAME – name of a physical network interface. Example: **eth0**

If a parameter can be written in different formats, it will be displayed in round brackets, the options are separated by "|" character. Example: **(A.B.C.D|<0-4294967295>).**

If a parameter is optional, it is put into square brackets: "**[]**".

Any command may contain "**no**" prefix. Having this prefix in the command means deleting a corresponding parameter from the configuration.

Start/stop of OSPF

Start of OSPF router is executed by the following command:

ospf start

In order to stop OSPF, execute the following command in *config* mode:

stop (daemon|clear)

Example:

>ospf

OSPF> configure

OSPF(config)# stop daemon

If "stop" command is executed with *clear* parameter, the router will clear its part of the system configuration prior to quitting CS.

Router identifier

Every OSPF router has a unique identifier. Identifier is a 32-bit integer. In order to assign an identifier, execute the following command in *config-router* mode:

router-id A.B.C.D

Example:

OSPF>configure

OSPF(config)# router

OSPF(config-router)# ospf router-id 195.38.45.107

OSPF(config-router)#

If identifier was not set by administrator, the router will automatically assign an identifier which equals to a maximal (by value) IP-address from all IP-addresses participating in OSPF system.

To cancel identifier assigning, use the following command:

no router-id

Filters

In many parameters of the router participating in the configuration filters are used. Filters are represented by two classes of objects:

§ Access lists (access-list)

§ Prefixes lists (prefix-list)

Access lists consist of a set of operators. Each operator consists of a range of IP-addresses and *deny* or *permit* command. The range of addresses is set as <value> <mask for insignificant bits>. The object to be filtrated has its basic parameter in the same format (IP-address, subnet etc). To make a decision whether the object corresponds with a list, each operator from the list is consequently applied to the basic parameter of the object until this parameter satisfies the condition. When a right condition is met, the decision is made according to the record in the command of the operator (*deny* or *permit*).

In OSPF router there are three types of access lists:

§ Standard. Is identified by numbers 1-99 or 1300-1999 and is used to analyze one parameter of filtration object.

§ Extended. Is identified by numbers 100-199 or 2000-2699 and is used to analyze two parameters of filtration object (for example, source address and destination address).

§ Nominate. Identical to Standard but is identified by a name (not number). Moreover, operators are configured in the format of <value>/<mask length>

In order to create or edit an access list in OSPF router the following commands are used (in *config* mode):

1. Standard access lists

access-list	(<1-99>|<1300-1999>)	(deny|permit)	A.B.C.D	A.B.C.D
	List identifier	Command	value	Mask of bits
			Range of values for the parameter	

This command creates an operator in a standard access list. Value and mask define a range (criteria) for the operator. The mask defines those bits of the value which form the range. For example, in order to specify the range of IP-

address from 192.168.12.0 to 192.168.255, one should specify the value of 192.168.12.0 and a mask of 0.0.0.255. For the value and mask of 0.0.0.0 255.255.255.255 there is a key word *any.* For example, the command:

OSPF(config)# access-list 1 permit 0.0.0.0 255.255.255.255

is equal to the command:

OSPF(config)# access-list 1 permit any

Correspondingly, for the range which consists of only one address, the key word *host* is used.

For example, the command:

OSPF(config)# access-list 1 permit 192.168.12.150 0.0.0.0

is equal to the following command:

OSPF(config)# access-list 1 permit host 192.168.12.150

2. Extended access lists

access-list	(<100-199>|<2000-2699>)	(deny|permit)	ip	A.B.C.D A.B.C.D	A.B.C.D A.B.C.D.
	List identifier	command		The range of source addresses	The range of destination addresses

3. Nominate access lists

access-list	WORD	(deny|permit)	A.B.C.D/M	[exact-match]
	List identifier	command	Range	The requirement for the exact match of a parameter to the range

In this case the list identifier is a character expression. The range is specified in a format of <value>/<mask length>. For example, if we need to specify the range of IP-addresses from 192.168.12.0 to 192.168.12.255, 192.168.12.0/24 is specified. For 0.0.0.0/0 range the key word *any* can be used. For example:

OSPF(config)# access-list TestList1 deny 192.168.1.0/24

OSPF(config)# access-list TestList1 permit any

While configuring, the operators are appended to the end of the list.

Lists of prefixes are different from access lists so that each operator has a number aside from a range (condition). Moreover, when a check for the parameter to fit into an operator's range is performed, one can set up additional condition for the parameter's mask length.

prefix-list	WORD	[seq <1-4294967295>]	(deny|permit)	A.B.C.D/M	[ge <0-32>] [le <0-32>]
	List identifier	Operator's position number	Command	Range	The range of the permitted mask length

If a sequential number is not specified the router sets it up automatically by adding 5 to the number of the last operator in a list. Thus, the operator will have the biggest number and will be placed in the end of the list.

Link state advertisement

The router can advertise its link states of two types:

1. *Internal links.* These are links which destinations are addresses of the subnets to which a router is connected directly (using one of its network interfaces) and which are described in OSPF router configuration.

2. *External links.* Links which destinations are route's destinations configured in WANFlex. These can be static routes (**route add (kernel))** or routes which appear in the routing table by assigning IP-address (alias) to one of physical network interfaces (**connected**).

In order to advertise an internal link, a subnet should be specified which destination is an advertised link. This can be done in *config-router* mode:

network A.B.C.D/M area (A.B.C.D|<0-4294967295>)

Network is specified by router's IP-address/mask which belongs to this network. Area ID can be inputted either in IP-addresses format or in decimal number format.

Example:

OSPF>configure

OSPF(config)# router

OSPF(config-router)# network 4.7.8.32/24 area 0.0.0.1

OSPF(config-router)# network 192.168.15.1/24 area 0

OSPF(config-router)#

If none of router's network interfaces has an IP-address from specified subnet, OSPF will not advertise this link although this network will be in configuration (inactive link).

Thus, the router obtains an internal link (for OSPF system) for which a given network is a destination. If this network is a physical interface address (point-to-point) the router gets an internal link with a router ID destination which is connected on the opposite end of point-to-point link.

To cancel internal link advertising use the command:

no network A.B.C.D/M area (A.B.C.D|<0-4294967295>)

In some cases there is a necessity to advertise internal links automatically for the selected network interface. It becomes important when IP-addresses of this interface (aliases) are created and deleted automatically, for example, when CPEs are connecting to the BS via radio. To implement this, use the command in *config-router* mode:

auto-interface IFNAME area (A.B.C.D|<0-4294967295>)

In the command an area ID is specified to which networks (destinations) will be deferred. To cancel an automatic links advertisement for this interface, use the command in *config-router* mode:

no auto-interface IFNAME

To announce external links use the following command in *config-router* mode:

redistribute (kernel|connected|static) [metric <0-16777214>] [metric-type (1|2)] [route-map WORD]

To define criteria according to which a router will advertise the link, use the command in config-router mode:

distribute-list WORD out (kernel|connected|static)

If this filter is not defined the router will advertise all links of the specified type of a system table, if they are not dejected by route-map configured in **redistribute** command parameters.

All links of this type are advertised as external type links with metric type 1 or 2 (External Type1|2). Information about external links is spread all over OSPF

domain (not only in the area). Stub areas are an exception to which the information about external links is advertised as *default gateway* through the area border router (ABR) of the area. Two types of metric differ in a way that metric type 1 is a metric which is "commensurable" with inner OSPF links. When calculating a metric to the external destination, the full path metric is calculated as a sum of a path metric of a router which had advertised this link plus the link metric. Thus, a route with the least summary metric will be selected. If external link is advertised with metric type 2 the path is selected which lies through the router which advertised this link with the least metric despite of the fact that internal path to this router is longer (with more cost). However, if two routers advertised an external link and with metric type 2 the preference is given to the path which lies through the router with a shorter internal path. If two different routers advertised two links to the same external estination but with different metric type, metric type 1 is preferred.

WORD – access list identifier to which destination of system routing table should respond.

Value and type of a metric for external links can be defined in route-map. In this case a type and value of a metric can be defined depending on route parameters (interface, gateway, destination etc).

If type and/or value of a metric left undefined the router will consider these external links to have a default metric and type 2. Default metric is specified using the following command:

default-metric <0-16777214>

If default metric is not defined, it equals 1.

In **redistribute kernel** mode the router will not make an advertisement into OSPF system about having as link to *default gateway* (destination = 0.0.0.0/0 network), even if it is clearly written in the routing table by the administrator. In order for the router to advertise its link to the *default gateway* it is necessary to clearly force him to do that using a command in *config-router* mode:

default-information originate [always] [metric-type (1|2)] [metric <0-16777214>] [route-map WORD]

metric-type (1|2) and **metric <0-16777214>** attributes define the same parameters of the external link for **redistribute** command. They are also not mandatory. This command also has one optional attribute – **always.** This attribute makes a router to advertise its *default gateway* link even if the route is not in the routing table.

To cancel advertising of an external link to *default gateway* us the command:

no default-information originate

Route map (route-map)

For more flexible configuration of metric type and its value for external links, one can use a route-map. Route-map is a set of conditional records. Each record has its number in the map, a condition of correspondence for the importing route of the record, actions to be done with a resulting object in case of its correspondence, resulting action (deny, permit) etc. Routes are listed in the route-map according to their number in ascending order. If a route satisfies a record's condition:

- If a resulting action is **deny**, the route is denied, review of map's records is aborted and a resulting object is cancelled (link is not advertised)
- If a resulting action is **permit**, all actions specified in the record are performed for a resulting object. Further, records viewing is stopped or, if specified in the scenario, it is resumed depending on the option specified in the scenario:

1. **on-match next** – viewing is continued from the record which follows a current record
2. **on-match goto <N>** - viewing is continued from the record which number is more or equal **N** but is not less than current number.

In order to configure a route-map, the following command is used in *config* mode:

route-map WORD (deny|permit) <1-65535>

where **WORD** – route-map identifier. This identifier is followed by a resulting action and the number of the record. If a record with a specified number does not exist it will be automatically created. After executing this command, CS switched to the mode for editing a selected route-map. For example:

OSPF> configure

OSPF(config)# route-map testmap permit 10

OSPF(config-route-map)#

After that, a condition of match between imported route and current record is specified. The following commands are used in *config-route-map* mode:

match address (<1-199>|<1300-2699>|WORD)

match address prefix-list WORD

match interface WORD

match next-hop (<1-199>|<1300-2699>|WORD)

match next-hop prefix-list WORD

These commands set matching conditions for the route according to three different parameters: destination, gateway (next hop) and interface. For every record it is permitted to set a number of different conditions. If several conditions are specified they will be conjugated by logical "and". In **match next-hop** and **match address command** a filtration object is specified (number or name): number or name of **access-list** or **prefix-list** name**.** In this case the condition will be satisfied if a corresponding route's parameter belongs to the specified filtering list, according to the rule corresponding to the list type. In **match interface** command a network interface name is specified to which a route (link) belongs.

If a route matches to all record's rules one can set values for route metric and/or metric type for a link which if formed from this router using commands in *config-route-map* mode:

set metric <0-4294967295>

set metric-type (type-1|type-2)

The next step for the record's behavior, after all conditions are matched by the route, can be configured using one of the following commands:

on-match goto <1-65535>

on-match next

Configuration example:

OSPF> configure

OSPF(config)# access-list AnyNetwork permit any

OSPF(config)# access-list net200 permit 192.168.200.0/24

OSPF(config)# route-map mapForConnected permit 10

OSPF(config-route-map)# match address net200

OSPF(config-route-map)# set metric 7

OSPF(config-route-map)# route-map mapForConnected deny 11

OSPF(config-route-map)# match address AnyNetwork

OSPF(config-route-map)# router

OSPF(config-router)# redistribute connected route-map mapForConnected

OSPF(config-route-map)#

In this configuration the router will advertise external links formed from the connected routes of the system routing table with metric type 2. With this, if a destination for this route is 192.168.200.0/24 network the formed link will have metric 7, any other destination will not lead to external link's advertising it.

Link metric

Link metric is a cost of traffic delivery through its network interface. OSPF router automatically calculates the cost of internal link taking physical interface's capacity to which link belongs into consideration:

M = reference_bandwidth/bandwidth.

reference_bandwidth – by default equals 100 Mbit/sec, **bandwidth** – a capacity (bandwidth) of a physical network interface to which the link belongs. Reference bandwidth can be modified using the following command in *config-router* mode*:*

auto-cost reference-bandwidth <1-4294967>

The parameter is specified in Mbit/sec.

A method for metric configuration described above is used for all links for which interfaces a specific cost is not set. To set an individual cost (metric) for links one can using the following command in *config-if* mode:

cost <1-65535> [A.B.C.D]

In order to get into *config-if* mode for the particular interface, the following command is used:

interface IFNAME

Example:

OSPF> configure

OSPF(config)# interface eth0

OSPF(config-if)# cost 4 192.168.15.1

OSPF(config-if)#

In **cost** command an IP-address is specified which is assigned to the interface in a subnet which is connected to this subnet. If this parameter is not specified every link for this interface will have a specified cost (metric) regardless from the destination subnet.

OSPF system areas

OSPF protocol has an ability to join adjacent networks and hosts into special groups. This group along with a router that has a link to one (any) of the networks included into the group is called an *area.* In each area an independent copy of OSPF is functioning. That means that each area has its own database and a corresponding graph.

A router that is configured to advertise only internal links is called an internal router (IR). A router connected to networks in more that one area is called *area border router (ABR).* A router that advertises its link to external destinations (**redistribute command**) is called AS Boundary Router (ASBR).

Each area is assigned a unique identifier *area-id*. An area with *area-id* equal to zero is called a backbone of OSPF system. OSPF backbone area always includes all ABR. Backbone area is responsible for routing information distribution between other (non-backbone) areas. Backbone area should be contiguous but it does not always imply a physical adjacency – backbone connections can be organized using virtual connections.

ABR models

OSPF router supports four models of ABR:

1. **cisco** – a router will be considered as ABR if it has several configured links to the networks in different areas one of which is a backbone area. Moreover, the link to the backbone area should be active (working).
2. **ibm** – identical to cisco model but in this case a backbone area link may not be active
3. **standard** – a router has several active links to different areas
4. **shortcut** – identical to standard but in this model a router is allowed to use a topology of connected areas without involving a backbone area for inter-area connections

Details on **cisco** and **ibm** models differences can be found in RFC3509. A *shortcut* model allows ABR to create routes between areas based on the topology of the areas connected to this router but not using a backbone area in case if non-backbone route will be "cheaper"

ABR model is selected using the following command in *config-router* mode*:*

abr-type (cisco|ibm|shortcut|standard)

If you want to use "shortcut" routes (non-backbone) for inter-area routes, you can use the following command in *config-router* mode:

area (A.B.C.D|<0-4294967295>) shortcut (default|enable|disable)

Three models define a usage of a specified area for routes shortcutting in shortcut mode:

- **Default** – this area will be used for shortcutting only if ABR does not have a link to the backbone area or this link was lost
- **Enable** – the area will be used for shortcutting every time the route that goes through it is cheaper
- **Disable** – this area is never used by ABR for routes shortcutting

Stub areas

Some of the areas may be defined as stub areas. It is used for the area which has either a single ABR or several ABR but route selection does not depend on external destination address. The information about external link (to OSPF system) is not sent to stub areas by ABR. Instead, ABR advertises a default gateway to the stub area with a route coming through this ABR.

The area can be configured as a stub area using the command in config-router command:

area (A.B.C.D|<0-4294967295>) stub [no-summary]

no-summary option is specified if it is not necessary to advertise a summary ads of other area's links to this area.

Backbone coherence. Virtual links

In general, OSPF protocol requires a backbone area (area 0) to be coherent and fully connected. I.e. any backbone area router must have a route to any other backbone area router. Moreover, every ABR must have a link to backbone area. However, it is not always possible to have a physical link to the backbone area. In this case between two ABR (one of them has a link to the backbone area) in the area (not stub area) a virtual link is organized. This can be done using the following command in *config-router* mode:

area (A.B.C.D|<0-4294967295>) virtual-link A.B.C.D

where

- **(A.B.C.D|<0-4294967295>)** – area identifier through which a virtual link goes
- **A.B.C.D** – ABR router-id with which a virtual link is established. Virtual link must be configured on both routers. For example:

Router 192.168.152.45:

OSPF> configure

OSPF(config)# router

OSPF(config-router)# area 0.0.0.1 virtual-link 192.168.78.12

Router 192.168.78.12:

OSPF> configure

OSPF(config)# router

OSPF(config-router)# area 0.0.0.1 virtual-link 192.168.152.45

Formally, the virtual link looks like a point-to-point network connecting two ABR from one area one of which there is a link to backbone area. This pseudo-network is considered to belong to the backbone area.

Link-to-area information filtering

Summary information about area's links which is advertised by ABR through backbone to other area (export) can be filtered. Moreover, the information from ABR (that came from other areas) can also be filtered (import).

Filters are configured in *config-router* mode:

area (A.B.C.D|<0-4294967295>) export-list NAME

area (A.B.C.D|<0-4294967295>) filter-list prefix WORD (in|out)

area (A.B.C.D|<0-4294967295>) import-list NAME

where

- **NAME** – name of a filtering list (*access-list*),
- **WORD (in|out)** – name of a filtering prefix-list with direction specification (in – import, out – export). Filters can be configured for all areas to which ABR is connected except for the backbone area.

Links aggregation. Advertising suppression

For every area to which OSPF router is connected there is a list of address ranges for link aggregation before sending a summary LSA to the backbone area. Aggregated links are checked to belong to one of the address ranges. If several links belong to one address range, ABR makes an advertisement to the backbone (or to other areas) of only one single link with destination equal to the address range and a metric equal to the maximal metric of all the links or equal to the specified for this range value. It is possible to announce that some range is a

blocking one, and then advertising of the links which belong to this range will be blocked. When advertising an aggregated backbone link to other (non-backbone) areas, the aggregation will not be performed if the area to which backbone links are advertised is a transit area (it has virtual links).

The list of addresses ranges for the area consists of the records that consist of the following fields:

- Range of addresses (R)
- Flag of advertisement suppression (not-advertise)
- The metric of an aggregated link (C)
- Advertised link (Rs)

If non-advertise flag is not specified, C and Rs parameters can be configured. If a destination for one or more links belongs to R, the router will advertise one link with R destination (or Rs, if specified) and with metric that is a maximal metric of the links (or C, if specified).

For addresses ranges there are several commands in *config-router* mode.

The command creates a range R and one can specify a "non-advertise" flag:

area (A.B.C.D|<0-4294967295>) range A.B.C.D/M [not-advertise]

The command creates a range R and configures a metric for an aggregated link C:

area (A.B.C.D|<0-4294967295>) range A.B.C.D/M [cost <0-16777215>]

The command creates a range and possibly creates a Rs destination instead of R:

area (A.B.C.D|<0-4294967295>) range A.B.C.D/M substitute A.B.C.D/M

Adjacency. Neighbors

When two or more routers have links to the same network these routers become neighbors in order to synchronize their Link-State Database. Moreover, a network with more than one router connected to it is a transit network; and, if this network is not point-to-point network, it is an active OSPF object (it can advertise its links to the routers). A special designated router makes a LSA. A designated router is selected from a number of active OSPF routers connected to the network based on their priorities, identifiers and IP-addresses of network interfaces by means of which they are connected to the network. The router uses special protocols which parameters should be identical for the neighbors. These parameters are:

- hello-interval
- dead-interval

By default, hello-interval equals 10 seconds; dead-interval equals 40 seconds. To modify these parameters for any network interface, use the following commands in *config-if* mode:

dead-interval <1-65535> [A.B.C.D]

hello-interval <1-65535> [A.B.C.D]

The value of the parameter is specified in seconds. "**IP-address**" defines IP-address of a specific link, if you need to configure this particular link (optional parameter). If this IP-address is not specified, the parameter will be applied to the network interface. Note that in order to creating adjacency relationship between two routers these parameters should be equal.

One of the routers connected to the network is automatically selected to be a designated router (DR) judging by three parameters. If a link priority is specified for the router it acts as a major criterion for DR selection. If priority is not set, only router-id and IP-address affect the selection.

To set up router's priority for the interface one can using the following command in config-if mode:

priority <0-255> [A.B.C.D]

Alike previously mentioned parameters, the priority can be set either to every link on the interface individually or to the interface as a whole. The bigger the priority the more chances this router has to become a designated router for this network. If this parameter is set to zero, this router will never be selected as a designated router.

OSPF protocol requires that Link-State databases of one area routers should be identical. To do that routers exchange LSA information. In particular, transit networks are used. In order to minimize network traffic, routers exchange their LSA not directly with each other but using DR and Backup DR (BDR). BDR is used for backing up DR in case of DR failure. BDR selection rules are identical to DR selection rules. While Link-state database synchronization the routers exchange database descriptions using master-slave relationship and broadcast IP-packets. Each packet reception should be acknowledged. If acknowledge is not received, initiating party makes a series of retransmits. OSPF administrator can control periodicity of these retransmits for each interface and/or interface's links in *config-if* mode:

retransmit-interval <3-65535> [A.B.C.D]

This retransmit interval is specified in seconds.

LSA exchange is performed in the following cases:

- start of the router or its connection to the network (link creation) after selecting a network designated router
- after receiving LSA from any other area's router
- periodically after old database information expiration

After receiving updated information about links changes, the router initiates its link-state database synchronization with its neighbors, if it's a DR. This process does not start right after new information receipt but after a period of time assuming that some more data may come. This is made in order to avoid network "storms". The time for the delay can be configured for every interface/link in config-if mode:

transmit-delay <1-65535> [A.B.C.D]

Moreover, the router automatically updates link-state information with its neighbors. Only obsolete information is updated which age has exceeded a specific threshold. By default, this threshold equals 1800 seconds (half an hour) and it can be changed using the following command in *config-router* mode*:*

refresh timer <10-1800>

The parameter is specified for the OSPF router in general.

Virtual link is a point-to-point transit network. In this network a neighboring relationship is also established between two routers. For virtual links there are similar parameters for neighboring relationship establishment. These parameters are configured in *config-router* mode*:*

area (A.B.C.D|<0-4294967295>) virtual-link A.B.C.D (hello-interval| <1-65535>

retransmit-interval|

transmit-delay|

dead-interval)

Authentication. Identity check

In order to prevent an unauthorized connection of the routers to OSPF system, the system has an identity check for protocol's packets. Currently the router has two different options for identity check (authentication):

- **Password authentication.** All packets sent to the network should have a corresponding value in a 64-bit OSPF authentication header data field. The value is a 64-bit password (not encrypted). Simple password authentication is vulnerable for passive attacks (sniffing) because broadcasting is used and the packet has a password in an explicit form.
- **Cryptographic authentication.** For each OSPF packet a key is used while generation and check of message-digest signatures which are added to the end of OSPF packet. Digital signature is built based on MD5 algorithm. Digital signature is based on one-way function using OSPF packet and a secret key. As a secret key is never send over the network in a clear form, this gives a protection from passive attacks.

By default, the router does not have any authentication (null-authentication).

Authentication can be configured individually for each interface's link (or for the interface including virtual link) and/or individually for every area to which the router is connected.

For interfaces authentication parameters are configured using the following commands in *config-if* mode:

1. Password authentication:

   authentication-key AUTH_KEY [A.B.C.D]

   where

   AUTH_KEY – password, **IP-address** is an optional parameter when individual link configuration is required.

2. Cryptographic authentication:

   message-digest-key <1-255> md5 KEY [A.B.C.D]

   where **KEY** – secret MD5 key, **IP-address** of the link in case of individual link configuration. <1-255> - a serial number of a secret key. Thus for the current link or interface one can configure up to 255 secret keys. For packets sending the router will use the latter keys among configured. For packets receiving the router will use the key with the same serial number as was used by the sender.

By setting up authentication parameters, one can turn it on by the *config-if* mode commands:

[(null|message-digest)]	**[A.B.C.D]**
Authentication type. null – no authentication (obligatory authentication suppression). With no parameter at all, a simple password authentication is turned on.	IP-address of interface's link

Virtual links authentication is configured in the same way in *config-router* mode:

Parameters:

area (A.B.C.D|<0-4294967295>) virtual-link A.B.C.D authentication-key AUTH_KEY

area (A.B.C.D|<0-4294967295>) virtual-link A.B.C.D message-digest-key <1-255> md5 KEY

Type of authentication settings:

area (A.B.C.D|<0-4294967295>) virtual-link A.B.C.D (authentication|) (message-digest|null)

Authentication type can be specified for the whole area to which a network belongs and a link by means of which OSPF packets are received. If authentication is turned on for both interface and the area, the interface authentication type will be used. In order to configure authentication type if it was disabled for interface (link) one can configure authentication type for the area using a command in *config-router* mode*:*

area (A.B.C.D|<0-4294967295>) authentication [message-digest]

If **message-digest** option is not specified, simple password authentication will be enabled for the area.

As was mentioned before, area authentication type is applied only if interface's authentication was totally disabled. However, interface's authentication parameters will be used.

To turn on area authentication, use the following command in *config-router* mode*:*

no area (A.B.C.D|<0-4294967295>) authentication

Router running configuration view

To review current running configuration of the router there are several commands in the basic mode of CS. In any mode of CS there is a command:

show running-config

This command shows a current router's configuration.

The configuration is shown as list of commands which brought the router to its current state.

Example:

OSPF> show running-config

Current configuration:

interface eth0

interface eth1

interface lo0

interface null0

interface tun0

network point-to-point

router

```
router-id 195.38.45.107
network 1.1.1.1/32 area 0.0.0.0
network 4.7.8.0/24 area 0.0.0.1
network 192.168.15.1/24 area 0.0.0.1
network 195.38.45.107/26 area 0.0.0.0
area 0.0.0.1 virtual-link 192.168.151.10
end
OSPF>
```

Neighbors

show neighbor [A.B.C.D] [detail]

As a parameter one can specify IP-address of a network interface (link), which state and neighbors list is to be shown. If this parameter is not specified the command shows a summary information for all interfaces.

Example:

```
OSPF> show neighbor

Neighbor ID     Pri   State           Dead Time   Address         Interface
9.1.1.8           1   Full/DROther    00:00:32    1.1.1.2         tun0:1.1.1.1
192.168.151.1     1   Full/DR         00:00:32    192.168.15.10   eth1:192.168.15.1
192.168.45.116    1   Full/DR         00:00:32    192.168.45.116  eth0:192.168.45.107
192.168.151.10    1   Full/DROther    00:00:39    192.168.151.10  VLINK0
OSPF>
```

Table columns:

- *Neighbor ID* – neighbor router-id
- *Pri* – priority
- *State* – current state/status. This parameter may be of the following value:
  - **Init.** This state means that a Hello packet was recently received from a neighbor with whom a 2-way connection is not yet established.
  - **2-Way.** A two-way connection is established between two routers. Starting from here an adjacency relationship is initiated.
  - **ExStart.** The first step in adjacency relationship establishing which sets up master/slave relations.
  - **Exchange.** In this state a router fully describes its link-state database by sending packets to its neighbor.
  - **Loading.** A state in which link-state database synchronization happens, i.e. a request for new information is sent to the neighbor.
  - **Full.** This state means that neighboring relationship is established and list-state database is synchronized.
  - Current status may be of the following values:
    - **DR** – the router is selected to be a designated router.
    - **Backup** – the router is selected as a backup designated router.
    - **DROther** – the router is neither DR nor BDR
- *Dead Time* – the time left for neighbor acknowledgement packet.

- *Address* – neighbor's IP-address
- *Interface* – interface (link) through which information with neighbor is exchanged.

If option **detail** is specified in the command**,** the information on neighbors is shown in the detailed way.

Database

show database

The command shows a summary table with a database contents (LSA).

show database (asbr-summary|external|network|router|summary)	**[A.B.C.D]**	**[adv-router A.B.C.D]**
Type of link advertisement for review	Link destination which advertisements are to be reviewed	Router-id which link advertisements are to be reviewed

For example, a database has to be viewed for the link which were announced by transit network, and the advertising routers was 192.168.45.107:

```
OSPF> show database network adv-router 192.168.45.107
       OSPF Router with ID (192.168.151.10)
                Net Link States (Area 0.0.0.0)
                Net Link States (Area 0.0.0.1)
  LS age: 473
  Options: 0x2  : *|-|-|-|-|-|E|*
  LS Flags: 0x6
  LS Type: network-LSA
  Link State ID: 192.168.15.1 (address of Designated Router)
  Advertising Router: 192.168.45.107
  LS Seq Number: 80000001
  Checksum: 0x9148
  Length: 32
  Network Mask: /24
        Attached Router: 192.168.45.107
        Attached Router: 192.168.151.1
                Net Link States (Area 0.0.0.2)
OSPF>
```

Filtration objects

show access-list [(<1-99>|<100-199>|<1300-1999>|<2000-2699>|WORD)]

This command is used to print access lists contents. If list identifier is not specified, all lists are printed. For example:

OSPF> show access-list

IP access list any_network

permit any

IP access list net200

permit 192.168.200.0/24

Similar commands are used for prefix-lists output:

show prefix-list

show prefix-list WORD

Routing table

show route

This command prints a routing table. For example:

```
OSPF> show route
============ OSPF network routing table ============
N IA 1.1.1.1/32            [3] area: 0.0.0.1
                           via 192.168.15.1, eth0
N IA 1.1.1.2/32            [2] area: 0.0.0.1
                           via 192.168.15.1, eth0
N    4.7.8.0/24            [2] area: 0.0.0.1
                           via 192.168.15.1, eth0
N IA 9.1.1.0/24            [12] area: 0.0.0.1
                           via 192.168.15.1, eth0
N IA 192.168.0.0/24        [3] area: 0.0.0.1
                           via 192.168.15.1, eth0
N    192.168.15.0/24       [1] area: 0.0.0.1
                           directly attached to eth0
N IA 192.168.80.0/24       [12] area: 0.0.0.1
                           via 192.168.15.1, eth0
N    192.168.151.0/24      [1] area: 0.0.0.1
                           directly attached to eth0
N IA 192.168.152.0/24      [2] area: 0.0.0.1
                           via 192.168.151.10, eth0
N IA 195.38.45.64/26       [2] area: 0.0.0.1
                           via 192.168.15.1, eth0
============ OSPF router routing table =============
R    192.168.151.10        [1] area: 0.0.0.1, ABR, ASBR
                           via 192.168.151.10, eth0
R    195.38.45.107         [1] area: 0.0.0.1, ABR
                           via 192.168.15.1, eth0

============ OSPF external routing table ===========
N E2 192.168.200.0/24      [1/7] tag: 0
                           via 192.168.151.10, eth0

OSPF>
```

This table consists of three parts:

1. **OSPF network routing table**. This section includes a list of acquired routers for all accessible networks (or aggregated area ranges) of OSPF system. **IA** flag means that route destination is in the area to which the router is not connected, i.e. it's an inter-area path. In square brackets a summary metric for all links through which a path lies to this network is specified. ***via*** prefix defines a router-gateway, i.e. the first router on the way to the destination (next hop).
2. **OSPF router routing table**.

3. **OSPF external routing table**. E2 flag points to the external link metric type (E1 – metric type 1, E2 – metric type 2). External link metric is printed in the format of <metric of the router which advertised the link>/<link metric>.

Interfaces information

show interface [INTERFACE]

This command prints the information on network interfaces including virtual links states. If interface name is not specified, all interfaces information will be printed. For example:

```
OSPF> show interface
VLINK0 is up
  Internet Address 192.168.151.10/24, Area 0.0.0.0
  Router ID 192.168.151.10, Network Type VIRTUALLINK, Cost: 2
  Transmit Delay is 1 sec, State Point-To-Point, Priority 1
  No designated router on this network
  No backup designated router on this network
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    Hello due in 00:00:08
  Neighbor Count is 1, Adjacent neighbor count is 1
eth0 is up
  Internet Address 192.168.151.10/24, Area 0.0.0.1
  Router ID 192.168.151.10, Network Type BROADCAST, Cost: 1
  Transmit Delay is 1 sec, State DR, Priority 1
  Designated Router (ID) 192.168.151.10, Interface Address 192.168.151.10
  Backup Designated Router (ID) 192.168.151.1, Interface Address
         192.168.151.1
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    Hello due in 00:00:05
  Neighbor Count is 1, Adjacent neighbor count is 1
  Internet Address 192.168.152.1/24, Area 0.0.0.2
  Router ID 192.168.151.10, Network Type BROADCAST, Cost: 1
  Transmit Delay is 1 sec, State DR, Priority 1
  Designated Router (ID) 192.168.151.10, Interface Address 192.168.152.1
  No backup designated router on this network
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    Hello due in 00:00:03
  Neighbor Count is 0, Adjacent neighbor count is 0
lo0 is up
  OSPF not enabled on this interface
null0 is down
  OSPF not enabled on this interface
rf4.0 is up
  OSPF not enabled on this interface
rf4.1 is up
  OSPF not enabled on this interface
OSPF>
```

7. Netstat command (Network statistics)

Display the network statistics

Syntax:

netstat -r

netstat -i

Description:

Displays the contents of different system data pertained to network parameters.

"**-r**" parameter displays system routing tables:

```
Routing tables

Internet:
Destination        Gateway            Flags     Refs     Use    Mtu  Interface
default            10.1.0.33          UG           2   744837      -  eth0
10.0.0.12/30       10.1.0.61          UG           0        0      -  eth0
10.0.1/30          10.3.1.26          UG           0      607      -  rf0
10.3.1.22          0:40:96:11:6a:43   UHL          1      182      -  rf0
10.3.1.24/30       link#3             UC           0        0      -  rf0
10.3.1.26          0:40:96:10:ef:be   UHL          2      271      -  rf0
127.0.0.1          127.0.0.1          UH           1        0      -  lo0
195.38.43          10.1.0.61          UG           0     1410      -  eth0
195.38.44.192/28   10.3.1.18          UG           0   258947      -  rf0
195.38.46.64/27    10.1.0.61          UG           0        0      -  eth0
195.38.46.128/26   10.1.0.61          UG           0        0      -  eth0
195.38.52          10.3.1.2           UG           0    19367      -  rf0
```

Flags for specific routes have the following meaning:

- **U** - this routing table element is currently active;
- **H** - this route leads to a host. If this flag is not set, the route goes to a network;
- **D** - the route has been created using the **icmp redirect** protocol; **M** - the route has been modified using the **icmp redirect** protocol;
- **G** - the route is connected to a host. If this flag is not set, it is considered that the route destination is directly connected;
- **S** - static route, set by the operator using a **route add** command;
- **1** - pseudostatic route, set as a result of a **rip static** command;
- **L** - the route points to a directly connected host (for such a route an APR request may be performed);
- **C** - when using this route, more specific routes may be created (e.g. using the **L** flag).

"**-i**" parameter displays the information on each network interface in the system:

```
Name  Mtu   Network          Address              Ipkts Ierrs    Opkts Oerrs
lo0   32768 Link:                                   0     0        0     0
lo0   32768 127              127.0.0.1              0     0        0     0
eth0  1500  Link:            0.5a.35.88.c3.54   192477     0   180180     0
eth0  1500  10.1.0.32/27     10.1.0.60          192477     0   180180     0
rf0   1500  Link:            0.5a.35.88.c3.55   324247  9085   348138     0
rf0   1500  10.3.1.8/30      10.3.1.9           324249  9085   348140     0
rf0   1500  10.3.1/30        10.3.1.1           324249  9085   348140     0
rf0   1500  10.3.1.24/30     10.3.1.25          324249  9085   348140     0
```

8. Ipfw command (IP Firewall)

General description

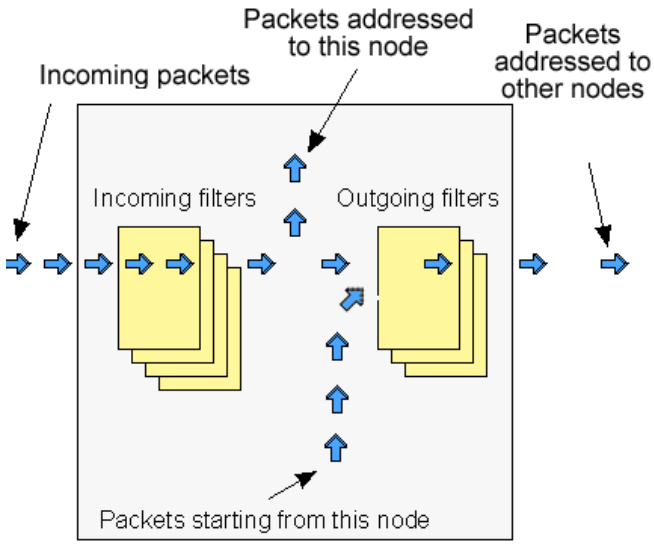
IP Firewall is a mechanism of filtering packets crossing an IP network node, according to different criteria. System administrator may define a set of incoming filters (**addincoming**) and a set of outgoing filters (**addoutgoing**). The incoming filters determine which packets may be accepted by the node. The outgoing filters determine which packets may be forwarded by the node as a result of routing.

Each filter describes a class of packets and defines how these packets should be processed (reject and log, accept, accept and log).

Packets can be filtered based on the following criteria:

- Protocol (IP, TCP, UDP, ICMP, ARP);
- Source address and/or destination address (and port numbers for TCP and UDP);
- The network interface it arrived on;
- Whether the packet is a TCP/IP connection request (a packet attempting to initiate a TCP/IP session) or not;
- Whether the packet is a head, tail or intermediate IP fragment;
- Whether the packet has certain IP options defined or not;
- The MAC address of the destination station or of the source station.

Below figure illustrates how packets are processed by the filtering mechanism of the router.



There are two classes (sets) of filters - prohibiting (**reject**) and permitting (**accept**).
Furthermore, a filter may be applied to all inbound packets or only to packets arriving via a specific interface.

Each received packet is checked against all filters in the order they are put in the set.
The first filter that matches the received packet determines how the packet will be treated. If the filter is an accept filter, the packet is accepted, otherwise it is rejected. If the packet matches no filter in the set, or if the set is empty, the packet is accepted.

The rejected packet will be discarded without notification to the sender.

Filters are defined using the **ipfw** command. For example, a command

ipfw add reject all from 192.168.5.3 to 192.168.11.7

adds to the set of incoming filters a reject filter which will discard all packets with source address 192.168.5.3 and destination address 192.168.11.7.

For better understanding of how filtering mechanism works, it is necessary to read how filters are defined and how filters are used.

Syntax:

ipfw list

ipfw flush

ipfw addincoming [num] . . .

ipfw addoutgoing [num] . . .

ipfw del num

ipfw mov num1 num2

ipfw [-]quiet

Description:

ipfw list

The set of currently defined filters is displayed on the operator terminal.

ipfw flush

All currently defined filters in both the incoming and outgoing filter sets are removed. Filtering is disabled.

ipfw addincoming [num] . . .

ipfw addoutgoing [num] . . .

These two commands are used to add a filter to the incoming and outgoing filter sets, respectively. The **add*** keyword is followed by a filter definition.

The optional **num** parameter may be used to explicitly specify the number of the new filter in the list. Execution of a command with this parameter implies automatic renumbering of the previously specified filters occupying higher positions in the list.

ipfw del num

Removes a filter from the appropriate list. The filter to be removed is specified by its number num which can be seen using the **ipfw list** command. The rules remaining in the list will be renumbered automatically.

ipfw mov num1 num2

Moves the filter **num1** into the **num2** position in the appropriate list. The rules between these two positions in the list will be renumbered automatically, namely: if **num1<num2**, then all numbers i such that **num1<i<num2** will be decremented by 1, and if **num1>num2**, then all i such that **num2<i<num1** are incremented by 1.

ipfw [-]quiet

The **ipfw** quiet command disables registration of rejected packets. Registration is enabled by default, and re-enabled by **ipfw -quiet** command.

Packet filtering rules

Hereafter we give detailed description of how packets are treated by packet filters. Every packet entering a router passes through a set of input filters (or blocking filters). Packets accepted by the input filter set are further processed by the IP layer of the router kernel. If the IP layer determines that the packet should go further and not landing here, it hands the packet to the set of outgoing filters (or forwarding filters).

Information on packets rejected by any filter is displayed on the operator's terminal, and the packets themselves are discarded without any notice to their sender.

A packet, "advancing through" a set of filters is checked by every filter in the set, from the first one till the end of the set, or until the first matching filter. The algorithm is as follows:

1. If the filter set is empty, the packet is accepted.
2. Otherwise, the first matching filter decides the packet's fate. If it is an accept filter, the packet is accepted. If it's a reject filter, the packet is rejected (discarded).
3. If no filter has been found that matches the packet, it is accepted.

The algorithm of applying any specific filter to a packet is as follows:

1. If the value in the **proto** field of the filter is not all, and the packet's protocol is different from that specified in the filter, then the filter is skipped (not applied) for this packet.
2. If the source address in the packet differs from that specified in the filter, then the filter is skipped (if the source address is specified in the filter with a mask, then the mask is applied to both addresses before comparing them).
3. If the destination address in the packet differs from that specified in the filter, then the filter is skipped (a mask, if any, is applied similarly to the previous step).
4. If the **ip_fragment** modifier is specified in the filter, but the packet is not an IP fragment, then the filter is skipped.
5. If the **ip_tail_fragment** modifier is specified, but the packet is either the first or the only fragment, then the filter is skipped.
6. If the **ip_head_fragment** modifier is specified, but the packet is not the first fragment of a fragmented IP packet, then the filter is skipped.
7. If the **tcp_connection** modifier is specified, but the packet is not the first or the only fragment of a TCP connection establishment TCP/IP packet, then the filter is skipped.
8. If the **ip_option** modifier is specified, but the packet has no options (with possible exception for NO-OP or EOL options), then the filter is skipped.
9. If the **ip_recroute_option** modifier is specified, but the packet has no related options, then the filter is skipped.
10. If the **ip_misc_option** modifier is specified, but the packet has no IP options (with possible exception for record-route, timestamp, NO-OP or EOL options), then the filter is skipped.
11. If the value in the **proto** field of the filter is **udp** or **tcp**, and the source address in the filter contains a port list, then, if the packet is neither the first nor the only fragment, or if the source port in the packet does not match any port specified in the filter, then the filter is skipped.

12. If the value in the **proto** field of the filter is **udp** or **tcp**, and the destination address in the filter contains a port list, then, if the packet is neither the first nor the only fragment, or if the destination port in the packet does not match any port specified in the filter, then the filter is skipped.
13. Otherwise, i.e. if none of the above conditions has caused skipping the filter, then the packet is treated in a way specified by the disp field of the filter.

Special filtering rules for ARP packets:

- ARP packets will always be permitted for those IP addresses and ranges of IP addresses that are mentioned in permitting (**accept**) filters, even if those filters are created for other types of packets.

Packet filtering rules syntax

Syntax:

[interface] disp

[vlan=N] [dot1p=N] [ether=X] [dscp=N|tos=N]

proto [modifiers] from [not] endpoint to [not] endpoint

Description:

Below is a description of the syntax rules for creating packet filters. Most attention is given to the syntax itself, but still filter usage questions are described either.

A generic form of the filter description is given above in the Syntax paragraph. Optional field **interface** defines the name of the network interface to which the filter is going to be applied. Interface name depends upon the router model and can be eth0 or rf4.0 for specifying Ethernet interface or radio interface correspondingly. If thy interface parameter is set the filter will be applied only to those packets which are received or transmitted through this interface.

Disp field (abbreviated from disposition) sets an action which is going to be held in case of this filter operation. Possible values are **accept** or **reject**. If **accept** value is set the packet will go through the filter. Using **reject** value means that the packet will be filtered. After the action value one can set an optional parameter log (**accept log** or **reject log**) – this will lead to the system log update in case of the filter operation.

Parameters **[vlan=N] [dot1p=N] [ether=X] [dscp=N|tos=N]** are classifiers that allows analyzing VLAN ID, 802.1p priority, packet type (EtherType) and also **ip_tos** field for having DSCP label value or IP precedence.

Proto field sets some particular IP-protocol, which is used for the filter. Possible values: **tcp**, **udp**, **icmp**, **arp**, **all** or a numeric value of the protocol.

Optional field **modifiers** can be used to set up some additional packet parameters which are going to be described below in this document.

Mandatory key word **from** separates **proto** and **modifiers** fields from the destination address (endpoint). Key word **to** separates source address from destination address.

Endpoint defines either source address or destination address. The exact syntax of endpoint fields depends upon **proto** field value. If **proto** has a value of either **all** or **icmp** than endpoint contains the address information. If **proto** is set as **udp** or **tcp** than endpoint contains the address information and an optional ports list.

Address information is an IP-address with a mask (optional). IP-address should be set in a traditional numeric format (nn.nn.nn.nn). An optional mask can be

set either as mask length in bits or as a numeric value in nnn.nnn.nnn.nnn format. Possible formats for address information are the following:

nn.nn.nn.nn

nn.nn.nn.nn:xxx.xxx.xxx.xxx

nn.nn.nn.nn/NN

Using semicolon means that the mask is set in a numeric address format. Slash symbol means that mask is set as a length in bit (number of first bits which are set as "1", others are set as "0").

Example:

192.168.9.0/24 sets the network address 192.168.9.0 with 24 bits mask length.

Second option: 192.168.9.0:255.255.255.0.

"0/0" means all possible IP-addresses.

If you need to create a filter which is applied to several network addresses or groups, it is more convenient to group all those addresses in one corresponding access list and specify the list name as an IP-address (**$ACLRULE**)

There are several predefined dynamic **ACL** lists which cannot be built in any other way.

$LOCAL list includes all local addresses owned by the router. This list can be used for a convenient filter description which allow (or restrict) the access to the device.

ipfw add accept all from 0/0 to $LOCAL

$ROUTE list contains system routes table excluding default route. When an address matches this list it means that this address has some specific route and default route will not be used in this case.

ipfw add reject all from 0/0 to not $ROUTE

For the interfaces which have physical MAC-addresses in Ethernet standard, it is possible to use a value of MAC-address with a key word **mac**. At that for the incoming filters one can set only the MAC-address of the source, and for outgoing – only the MAC-address of the destination. Moreover, instead of a numeric value a key word **$BS** can be used. In this case the MAC-address of the corresponding BS (on which the CPE is configured) will be used. One should keep in mind that the rules which use MAC-addresses for the incoming packets will be applied in the first turn, and the rules for the outgoing packets will be applied in the last turn. Be careful.

After **from** and **to** key words one can use a negative prefix **not**. Its action will spread only on the corresponding address (addresses) but will not influence the ports if they are used in the command.

Example:

ipfw add reject all from mac 0012345678 to 0/0

ipfw addout reject all from 0/0 to mac 0012345678

ipfw add rf1 reject all from mac $BS to 0/0

ipfw add reject all from 0/0 to not 1.1.1.0/24

Ports list is set as a simple enumeration of ports separated by space bars.

The first element in the list can be a port couple separated by a semicolon. These ports will specify a port values range (from the smallest to the biggest inclusively).

One can specify up to 10 ports in the list.

The packets which are not a first fragment of the fragmented IP-packets are not checked to fulfill the port number restrictions (as a port number is specified only in the first fragment). If the first fragment is filtered the rest of the fragments will be rejected by the target machine IP-protocol.

Modifiers field is used for the additional packet characteristics which can be considered by the filter.

Possible values:

- **tcp_connection**

  The filter is refered only to the packets of an establishing a TCP-connection. **connection** is synonym of tcp_connection. Technically, a packet for requesting a connection has a TCP header with SYN flag set and ACK flag cleared.

- **ip_fragment**

  The filter refers only to fragmented packets. Technically, either offset field in the packet has non-zero value or a more fragments bit is set.

- **ip_head_fragment**

  The filter is applied only to the first fragment of the fragmented packet. Technically an offset field in the packets has non-zero value or a more fragments bit is set.

- **ip_tail_fragment**

  The filter is applied to all packet's fragments excluding the first one. Offset field has non-zero value. More fragments field value is of no importance.

- **ip_option**

  The filter is applied to the IP-packets which have any IP-options set (excluding NO-OP option)

- **ip_recroute_option**

  The filter is applied only to those IP-packets which have either record-route or timestampIP options set without any other options. These options can be set by violators to build your network map. No other threat is possible here.

- **ip_misc_option**

  This filter is applied only to the packets which have one or more IP-options but record-route, timestempIP or NO-OP. Many of IP-options of MISC group are used by the violators to avoid filters in order to enter the network.

There are several additional rules for the modifiers field:

1. **tcp_connection** value can be used only when the **proto** field has **tcp** value

2. If more than one option among **ip_fragment**, **ip_head_fragment** or **ip_tail_fragment** is used, than the latter ones will cancel the action the former ones.

3. If more than one option among **ip_option**, **ip_recroute_option** or **ip_misc_option** is used, than the latter ones will cancel the action the former ones. The packet must fulfill all options set, otherwise it will go through the filter.

Examples of packets filtering

Hereafter some examples are given of how to use the **ipfw** command in different cases.

Simple examples:

Our first example will be a filter prohibiting passage of any packet from some "unreliable" address 1.1.1.1 to the address 2.2.2.2:

ipfw add reject all from 1.1.1.1 to 2.2.2.2

As enemies often attack in unite front, let us now bar the way to all packets from the whole hostile network:

ipfw add reject all from 1.1.1.0/24 to 2.2.2.2

Here **24** after the slash means the mask length in number of bits. The mask length of 24 corresponds to a C class network with 256 different node addresses. Using a colon sign ("**:**"), the same command may be equally expressed as follows:

ipfw add reject all from 1.1.1.1:255.255.255.0 to 2.2.2.2

We can go even further, stopping all packets sent from the enemy network to any address (provided of course that they pass through our router):

ipfw add reject all from 1.1.1.0/24 to 0/0

Filtering by port numbers

Now suppose that we want to authorize everybody to address an smtp service (mail agent) at the host with IP address 192.5.42.1. It may be done with the following command:

ipfw add accept tcp from 0/0 to 192.5.42.1 25

The **tcp** keyword means that the filter will be applied to TCP packets only. The IP-address of the mail host machine is followed by the port number 25, corresponding to the SMTP service.

You can use a port list to specify several ports in the same command. The first element in a list may be an interval of port numbers, specified by its lowest and highest values separated by a colon. For example, the following command

ipfw add accept tcp from 0/0 to 1.1.1.1 900:5000 25 113

will authorize passage of tcp packets sent to the IP address 1.1.1.1, if the destination port number is within the 900 to 5000 interval (including both extreme values), or is equal to 25 (smtp) or 113 (ident).

All the subnetworks of the inner network, including the innerhost address, belong to the same network (or group of network). Suppose that we know for certain that there may not be any host in the outer network having an address within the inner network's address range. Therefore, any packet received from the rf4.0 interface of a router running ipfirewall, hence from the outer network, but having the source address within the inner network's address range, must be discarded. It is done by the following command:

ipfw add rf4.0 reject all from innerhost/16 to 0/0

Unlike the filters in the previous examples, this filter will be applied to packets arriving through the rf4.0 interface only. Packets arriving through any other interface will not be discarded (in this example the inner network is supposed to be of the B class).

As an additional measure it may be useful to reject packets having a source address from within the loopback network (**127.0.0.0**):

ipfw add rf4.0 reject all from 127.0.0.0/8 to 0/0

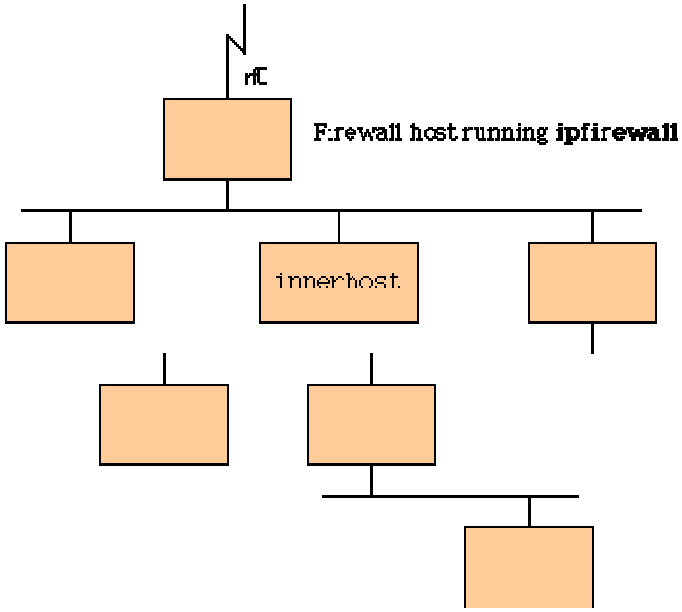
IP spoofing has been widely used in the Internet as an aggression method. For additional information, see CERT summary CS-95:01, and also summaries on the CERT WWW site.

It is important to consider that a malefactor may use IP spoofing for breaking in your network despite an obvious fact that he will never receive any reply. See e.g. CERT advisory CA-95:01.

IP-spoofing

In the previous examples, the source address was used a main and the only criteria for the address reliability checking. Unfortunately, there is a possibility to send the packets from an unreliable address, substituting the return address with that you rely on (this attack method is called IP spoofing). It is clear that the checking only of the source address is not enough. It is necessary to check the path of the packet or, which is more practical, to check the interface through which the packet was accepted.

A network example is shown below:



All subnets of an inner network, including a host address innerhost, are owned by the one network (or a network group). Let's imagine that outer network has no hosts which are within the range set up for the inner network. Therefore, all the packets that are accepted via rf4.0 interface of the router with firewall run on it and have the source address which is in the range of addresses of the inner network must be blocked. The following command can perform this action:

ipfw add rf4.0 reject all from innerhost/16 to 0/0

Compared to all previous examples this filter will be applied only to those packets which come through rf4.0 interface. Packets which come through any other interface ill not be blocked (in the example the inner network has addresses of the B class.

As an additional security measure it makes sense to block all packets with source address from the loopback network (127.0.0.0):

ipfw add rf4.0 reject all from 127.0.0.0/8 to 0/0

Filtering TCP connections

TCP/IP clients normally use port numbers between 900 and 5000 inclusive, leaving port numbers below 900 and above 5000 for servers. The following pair of filters will bar access to your servers for any outside clients (assuming that all

communications between your network and the external world pass through the **rf4.0** interface):

```
ipfw add rf4.0 accept tcp from 0/0 to 0/0 900:5000
ipfw add rf4.0 reject tcp from 0/0 to 0/0
```

The first of these filters accepts packets from external sources to ports from 900 to 5000 on the inner network hosts (normally assigned to internal clients). The second filter rejects all the rest.

Unfortunately, this is not enough. Some internal servers may be assigned port numbers within the 900 to 5000 range, and the above filter set would allow access to those servers for external clients. The problem consists in restricting external access to your servers having such port numbers while leaving them open for internal access. One of the possible solutions is to reject any attempt from an external client to establish a TCP connection with an internal server.

The **tcp_connection** modifier makes it possible to do:

```
ipfw add rf4.0 reject tcp_connection from 0/0 to 0/0 900:5000
ipfw add rf4.0 accept tcp from 0/0 to 0/0 900:5000
ipfw add rf4.0 reject tcp from 0/0 to 0/0
```

The first filter in the above filter set wards off any attempt of TCP connection establishment from outside clients to your internal servers with port numbers 900 to 5000. The second filter authorizes any other incoming TCP packets aimed at port numbers within the same range; and the third filter rejects all other TCP packets.

This unreliable UDP protocol

Unlike the connection-oriented TCP protocol, the UDP protocol sends separate packets (datagrams). In this protocol every packet is transmitted independently from all others, and if there is a logical connection or session between a client and a server communicating through UDP, such connection or session exists between higher layer application entities only, and is invisible to UDP.

As all UDP packets are independent of each other, a UDP packet header bears no information on whether it is a client to server or a server to client packet (in fact, UDP users are all equal in rights; the terms client and server cannot be defined explicitly).

Therefore, the only recipe we can propose is to define as precisely as possible the range or set of those UDP port numbers which are allowed to communicate with the outer world.

A domain name server (**DNS**) is an example of a server using the UDP protocol (at port number 53). Assuming that your communications with the outer world all pass through the rf4.0 interface, the following filter set will provide for proper interaction between your internal DNS server and external DNS servers while rejecting any other UDP traffic:

```
ipfw add accept udp from 0/0 53 to 0/0 53
ipfw add rf4.0 reject udp from 0/0 to 0/0
```

Though it may appear an easy task, in reality it is very difficult to establish more open UDP access policy without creating large security holes. If, in particular, you decide to authorize your internal clients accessing external UDP servers, then you should take into account the following considerations (the list is far from exhaustive):

If you have NFS servers, these are traditionally using the UDP port 2049 (TCP versions of NFS servers also use the port number 2049, which may possibly be protected by the **tcp_connection** modifier - see examples above).

Some RPC portmapper implementations have grave security problems. Be very careful when authorizing external access to your internal portmapper resource (at TCP or UDP port 111).

Be also very careful in your choice of source and destination ports to authorize. You might be tempted to authorize external packets arriving from some port numbers you know. If you do, always remember that a malefactor can easily send any TCP/IP or UDP/IP packets with any combination of source ports and addresses replacing his own ones.

Some Microsoft LAN Manager services use UDP. As Microsoft has a visceral enmity against open secure protocols, and its own implementations have unprecedented number of bugs and errors, you should better exclude any possibility for potential malefactors to profit by this security hole:

ipfirewall add rf4.0 reject tcp from 0/0 to 0/0 135:139

ipfirewall add rf4.0 reject udp from 0/0 to 0/0 135:139

This subset of filters protects you quite securely from almost any possible attempt to break in your internal network having Windows NT/95/98 servers and/or workstations installed.

IP fragments

The **ip_fragment**, **ip_head_fragment** and **ip_tail_fragment** modifiers are intended for managing a flow of fragmented IP packets. For better understanding how you can use them, the following considerations should be taken into account:

- A filter verifying TCP or UDP port numbers never checks IP fragments except the first one in a sequence.
- If your filter accepts incoming IP fragments, a malefactor may use a "denial of service" attack, by flooding you with fragments having different source addresses, thus causing memory overflow on your router.

Therefore, to be protected from a possible "denial of service" attack, the only solution would be to prohibit reception of any fragmented packets:

ipfw add reject all ip_fragment from 0/0 to 0/0

This measure certainly strengthens your security; don't forget, however, that a malefactor still may use other methods of aggression, e.g. by simply pelting you with any packets or with useless e-mail messages.

Moreover, rejecting all incoming fragmented packets may hamper your normal work. Consider the following example. The maximum possible IP packet length is usually circa 1500 bytes; but it may be less or more on different network segments. Even those packets which have not been sent fragmented by their source, may have become fragmented somewhere on their way to destination, because they have encountered a network segment with more severe packet length limitation. Even the newest protocols for defining the maximum possible IP packet length along any given route are not always bringing guaranteed result, because IP packets from the same source are progressing independently through the network, and may take different routes. Therefore, fully prohibiting reception of fragmented packets may hinder (temporarily or permanently) normal operation of some applications communicating with some hosts.

If you decide to authorize incoming fragmented packets, then one of the first filters to apply could be

ipfw add accept all ip_tail_fragment from 0/0 to 0/0

The above filter accepts all incoming fragments except the first fragments (of their respective packets). Such an authorization is not harmful for your security (with the exception of a "denial of service" attack), because the first fragment of a packet, bearing the main information about the whole packet, will be already verified by some of the preceding filters. If the first fragment has been rejected by a filter, then all the remaining fragments, when received by the destination host in the absence of the first one , will be rejected there after some delay (normally fixed at 60 sec.).

Logging of packets

IP Firewall registers all rejected packets, writing appropriate message in the system log. Registering all accepted packets may be additionally requested by putting a **log** keyword:

ipfw add accept log icmp from 0/0 to 0/0

The above command will register **all** incoming ICMP packets.

A big number of logged packets may cause system log overflow (if you have redirected log messages to a remote workstation).

9. Loadm command (load meter)

This is a tool to perform the channel load monitoring

Syntax:

loadm [-B] [-l] [-w XX] interface

Description:

This command allows estimating the load of a system interfaces specified by **interface** parameter. By default, the information is displayed in one line and is updated every second; the load is measured in Kbit/s.

The following additional keys change default settings:

- **-B:** display values in thousand of bytes per second;
- **-l:** display information line by line;
- **-w XX:** specifies time interval XX between updates.

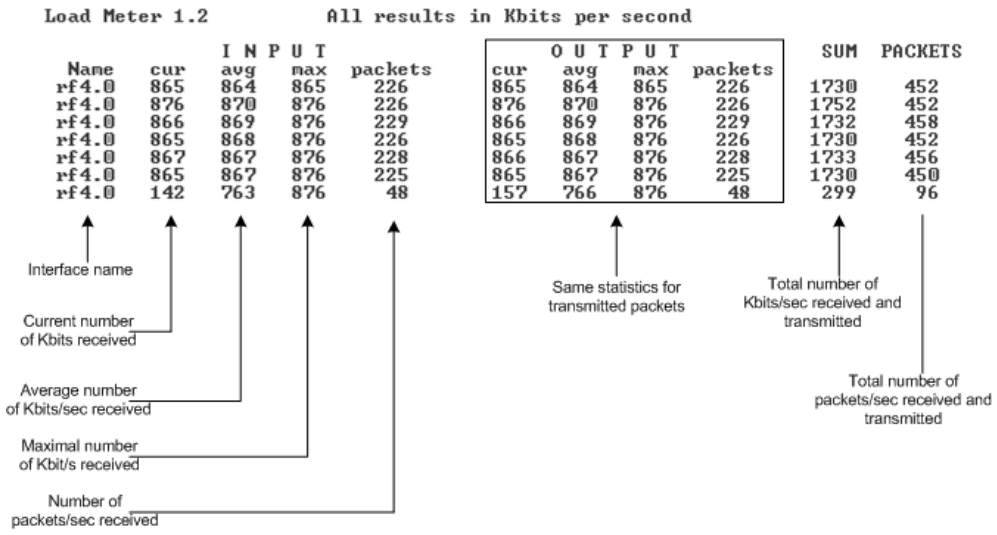
For the rf4.0 radio interface the loadm command shows the total channel load, and not only that part of it which corresponds to the given device. On a client unit, for example, not only that unit's own traffic appears on its rf4.0 interface and is taken into account by the loadm command, but also the traffic between the base station and other client units. In other words, the channel being observed comprises in this case the totality of a cell traffic (generated by several mutually affecting units on the same frequency). This is a particular feature of R200 radio modules equipping RWR 110/120/210/510/520 wireless routers.

Therefore, the real load of such unit is better seen on its **eth0** interface.

Example:

loadm -l rf4.0

The output example is shown below.



10.Bpf command (Berkeley Packet Filter)

The command enables packet capturing regime (Berkeley Packet Filter)

Syntax

bpf ifname IP_addr port

bpf ifname -

Description:

The packet capturing regime, which is enabled by the first of the above commands and disabled by the second, allows for replicating entire information flow through any of the system interface and forwarding the replica to a remote workstation for subsequent analysis and check. The filter does not interfere with normal operation of the router.

Because of limited memory capacity and CPU speed, the router software is not capable itself of sorting and analyzing data flows. The bpf command helps to perform thorough analysis on any network workstation, even in real time and with the help of more powerful analysis tools (e.g., tcpdump utility).

Each packet of the data flow through the specified interface (together with its MAC header) is sent using the UDP protocol to a remote workstation at the specified address and port.

Parameters are as follows :

- **ifname**: the name of the interface delivering the stream to be analyzed
- **IP_address**: the IP address of the destination of the replicated data stream
- **port**: the number of the port to which the replicated data stream should be sent

It does not cost much labour to write a simple program for receiving packets at the workstation, storing them in memory and/or delivering for examination by a packet analyzer. You may use as a prototype the source code of the bpfshow command for the FreeBSD/OpenBSD system. In this case, the program is launched with a port number as the only parameter, and stores all packets received in a file named **pcap.raw**; then the flow contents could be analyzed with the tcpdump utility:

bpfshow 8000

^c

tcpdump -r pcap.raw

Example:

bpf rf4.0 10.11.12.13 8000

Enables packet capturing regime, sending all packets from the rf4.0 interface to a workstation at the address **10.11.12.13.**

bpf rf4.0 -

Disables packet capturing regime at the **rf4.0** interface.

11.Snmpd command (SNMP daemon)

SNMP protocol version 1 and 3 daemon

Syntax:

snmpd comm[unity] access_key

snmpd user user_name add|set [pass password] [sec[urity] noAuthNoPriv|authNoPriv]

snmpd user user_name del[ete]

snmpd v1disable|v1enable

snmpd start | stop

Description:

This command enables/disables the SNMP (Simple Network Management Protocol) Version 1 and 3 daemon.

SNMP protocol support is an important feature of all communication devices because it allows the system administrator to use a uniform mechanism to manage the operation of a network as a whole and of every its component individually.

Although the first version of the SNMP protocol lacks security in the operation of the protocol itself, which hinders its use for network management, it is widely used to monitor and analyze network operation. MIB variables changing is turned off for the first version; it works only in read-only mode. **v1disable** option disables 1st version support completely and slightly fastens incoming SNMP-requests processing.

Now we have a support of SNMP-V3 with USM (User-based Security Model) and MD5 authentication excluding encryption. For access granting, a user with username, password and access rights (with or without authentication) is being created.

The present implementation supports MIB II (Management Information Base, Version II) and MIB Enterprise and is very easy to configure.

Example:

snmpd comm secret

snmpd user john add pass mypassword security authNoPriv

snmpd on

12.Td command (Telnet daemon)

Telnet daemon management.

Syntax:

td enable | disable RemoteHOST

td start | stop | flush

Description:

Telnet daemon makes it possible to remotely configure and manage a router, and more generally to execute any operation system commands in the same way as it is done on a local operator workstation.

Telnet daemon starts automatically when the router is switched on.

To stop the daemon operation, a **td stop** command shall be executed; a **td start** command restarts the daemon.

By default, the daemon accepts **SNMP** connection establishment requests from any host in the network. After executing one or several **td enable RemoteHOST** commands, remote SNMP access becomes only possible from the explicitly specified IP-addresses (one host specified per each td enable command, up to 10 hosts enabled simultaneously).

To retire from a remote host a previously granted access authorization, a td disable command with its IP-address shall be executed.

Finally, a **td flush** command fully clears the current telnet daemon configuration.

Examples:

td enable 195.38.44.1

td enable 195.38.44.11

td start

13.Nat command (Network Address Translation)

Network address translation according to RFC1631.

Syntax:

nat command [arguments]

General description

NAT allows solving to the certain extent the problem IPv4 address space exhausting. It means that several computers in the given LAN may connect to Internet via the same public IP address. NAT-module receives outgoing IP-packets, modifies sender's IP address to the public IP address and forwards it to Internet. Sender's IP address is modified in such a way that it it possible to identify the sender when IP packet received on the LAN incoming interface and to forward the IP packet to the initial sender. NAT-module is similar to **natd** and **libalias** from FreeBSD. Original manuals can help understand the subject better.

As its known (rfc1918), some part of IPv4 address space is reserved for using in so called private IP networks (private internets).

10.0.0.0 - 10.255.255.255 (10/8 prefix)

172.16.0.0 - 172.31.255.255 (172.16/12 prefix)

192.168.0.0 - 192.168.255.255 (192.168/16 prefix)

Internet backbone routing protocols do not advertise these addresses, which allows to use the same addresses in different Internet segments. These addresses are used by ISP's and enterprises to build internal transport environment and/or to connect small subscriber communities.

Perhaps, when connecting your LAN to Internet, ISP will suggest you to minimize the number of really existing IP-addresses in order to save its own address space. Common user needs very limited set of well-known services: WWW, FTP, ICQ, Telnet, SMTP, Games. This is quite accessible using private internets and NAT. Besides, there are dedicated proxy-servers for concrete services which fit better for this task. E.g. for HTTP and FTP it is better to use caching proxy server Squid.

If you decided to use IP-telephone based on H.323 standard, then it is better to use private internets. We have H.323 support module in our NAT version.

So, we have the following scenario: using private internets in your LAN and you have a limited number of public IP-address.

Commands description

nat local_acl $NAME [public_addr|dhcp IFNAME] [enable|disable|delete]

This command sets the real (public) IP-address which will be used for address translation. In order for the routing protocols to work normally, this address must be assigned to any physical interface of the router. Infinet router has at least two physical interfaces: Ethernet (eth) and radio (rf). Usually, the system is linked to ISP's backbone networks via radio interface and ISP's backbone is built using private internets. So what is the physical interface to assign the public IP?

It may be assigned using alias name to any physical interfaces or to virtual interface null0.

ifconfig null0 123.1.1.1/32 up

More than that, sometimes one can avoid public IP assignment to physical interfaces at all. The procedure goal is to provide public IP accessibility from Internet. But this may be done using static routing. All packets routed to this public address will get into your LAN. Link with the physical interfaces is not necessary. NAT-module will perform conversion before packet forwarding - enough packets entering into the router.

If the provider gave you a small block of address (e.g. 123.1.1.0/30), you can assign the whole block on **null0** (e.g. "ifconfig null0 123.1.1.0/30") and use these addresses. For example, in this case you can use the first address 123.1.1.0 as an alias_address, and the rest – for the packets redirection on the local machines using **nat redirect_xxx** (see below) or for other public addresses for other private networks.

NAT module is designed in such a way so the original source and destination addresses are used (this is important when creating firewall rules, qm rules, ipstat analyzing). For example, when creating a Firewall rule, one should use local addresses for the private network. They will be shown in **ipstat** module also.

This command also sets the name of an access list (ACL) of your private networks, which require network address translation.

All packets with source addresses that are included into the **local_acl** list are considered as outgoing and are subject to translation. Exceptions are the packets going from local_acl to local_acl, and packets going from local_acl to the system own addresses. All these packets and the rest of the packets ae considered are incoming and, if they are not reserse to the translated connections, pass through without being changed.

acl add $NAT net 192.168.1.0/24

nat local_acl $NAT 123.1.1.1

In this example we created a list with the only network 192.168.1.0/24 (your private network), referring to it in **local_acl** command and assigning 123.1.1.1 address as a public address for this network.

You can create several private networks having assigned different public addresses to each of them. Translation will be carried out independently.

Also, an address obtained by DHCP protocol can be used as a public address. To do this, an option **dhcp** should be used instead of specifying a fixed public IP address. Also, after key word **dhcp** an interface through which DHCP parameters were obtained should be specified.

Example:

nat local_acl $NAT dhcp eth0

enable, **disable** and **delete** commands are used for local_acl command manipulations – enabling this record, disabling or deleting correspondingly.

nat alias_address 123.1.1.1

This command is obsolete. Use **local_acl** command.

nat maxlinks NUM

This command set the maximum number of supported connections. 1000 by default.

The system automatically observes all the connections and dynamically destroys all unnecessary connections according to their type and time of activity. However, when using different network scanners there is a possibility that current number of connections will increase enormously or until there is a free space in the RAM. Using this command one can avoid this situation to happen. In the case when the number of current connection exceeds the threshold set the system will put the warning into the system log and restrict new connection establishment until the situation becomes stable. When connections number will decrease the corresponding message will be put into the system log and a normal work will be resumed.

Generally, it is enough to run NAT.

nat enable

This command enables NAT-module to start NAT according to specified rules.

Example:

ifconfig null0 123.1.1.1/32 up

rip start # to start dynamic routing for public IP

acl add $NAT net 192.168.1.0/24

nat local_acl $NAT 123.1.1.1

nat enable

Done. One can start to check access from the LAN.

nat disable

Disables NAT.

nat same_ports yes|no

This command forces NAT-module to leave ports numbers in the modified packets as they are. If it is impossible then arbitrary port numbers will be used.

nat verbose yes|no

Enables diagnostic mode and prints modified packets inot system log.

nat Proxy only yes|no

If enabled then NAT-module only forwards packet according to proxy_rule commands. Usual NAT not performed.

nat stat

Shows NAT statistics.

Packet redirection

NAT disadvantage is that local hosts are not accessible from Internet. Local hosts can establish outgoing connections but cannot serve incoming. This hinders starting Internet applications on local hosts. Simple solution is to redirect traffic from some ports to local hosts.

The below commands dedicated for creating redirection rules (**redirect_xxx** and **proxy_rule**). Multiple command execution with different arguments allowed. Commands are numbered when browsed using **config show**. This allows to delete not needed rules using **nat del XX** where XX is sequential number in the **config show** list.

nat redirect_port

The command comes with two flavours.

First type:

redirect_port proto localIP:localPORT[-localPORT]
[aliasIP:]aliasPORT[-aliasPORT]
[remoteIP[:remotePORT[-remotePORT]]]

Redirect incoming packets for specified port to other address and other port.

Argument **proto** may be **tcp**, **udp**, **ras** or **cs**.

In case of **ras** and **cs** address modification is performed according to H.323

TargetPORT - given port or port range on the system.

AliasPORT - destination port (or port numbers).

The port ranges **aliasPORT** and **targetPORT** may not coincide in numbers but should be of the same range.

If you are using several pairs of public address-private network, it is recommended to specify the exact public address.

Parameters **remoteIP** and **remotePORT** may be specified for more exact definition of incoming packets (packets only from specified source and port will be allowed). If **remotePORT** is not specified then its range should coincide with range of **targetPORT**.

nat redirect_port tcp 192.168.1.5:23 7777

In this example all incoming tcp connections to port 7777 will be redirected to host 192.168.1.5 port 23 (telnet).

nat redirect_port tcp 192.168.1.4:2300-2399 123.1.1.2:3300-3399

All incoming tcp packets with targetPORT range 3300-3399 and destination address 123.1.1.2 will be redirected to 192.168.1.4. Port mapping is "1 to 1", i.e. 3300->2300, 3301->2301.

For example, IRC-server is running on client A and WEB-server is running on client B. Then in order to get it work, connections accepting on ports 6667(irc) and 80(web), should be redirected to the appropriate hosts:

nat redirect_port tcp 192.168.0.2:6667 6667

nat redirect_port tcp 192.168.0.3:80 80

Second type:

redirect_port proto localIP:localPORT[,localIP:localPORT[,...]]
[aliasIP:]aliasPORT
[remoteIP[:remotePORT]]

Cyclic redirection of incoming packets to several destination addresses (lke redirect_address) for unform load distribution between them (**LSNAT**):

nat redirect_port tcp 192.168.1.2:80, 192.168.1.3:80 123.1.1.2:80

In this case all requests to WEB-server 123.1.1.2 will be redirected to the LAN servers.

nat redirect_address local IP [,local IP,…] public IP

Redirects all incoming traffic directed to **publicIP** to **localIP**. If several **localIP** addresses specified then redirection will be done in round-robbin fashion.

nat redirect_address 192.168.1.2 192.1.1.1

nat redirect_address 192.168.1.3 192.1.1.2

In this case all traffic incoming to 192.1.1.1 will be redirected to the LAN address 192.168.1.2, and traffic incoming to 192.1.1.2 will be redirected to 192.168.1.3.

Address redirection makes sense when there are several IP-addresses on the same host. In this case NAT can assign to every LAN client its own external IP-address.

Then NAT transforms outgoing packets, changing IP-addresses to public external IP-addresses. For example, IP-addresses 128.1.1.1, 128.1.1.2, 128.1.1.3 belong to the gateway. 128.1.1.1 can be used as public gateway IP-address, and 128.1.1.2 and 128.1.1.3 will be redirected to LAN clients A and B:

nat redirect_address 192.168.1.2 128.1.1.2

nat redirect_address 192.168.1.3 128.1.1.3

redirect_proto proto localP [publicIP [remoteIP]]

Redirects all the incoming packets with specified protocol **proto** to the host with address **localIP**.

nat redirect_proto 47 192.168.1.2

If **publicIP** then used value of command **alias_address**.

nat default_h323 [yes|no]

Includes address modification according to H.323 stack for outgoing connections. Affects all incoming UDP packetsd destined for port 1719 and incoming TCP connections for port 1720. By default disabled.

Do not enable this option unless needed, because this will hinder NAT performance if not used in IP telephony applications.

nat h323_destination ras|cs remote_addr[:remote_port] [local_addr[:local_port]]

Enables to describe more specifically using of H.323 elements in the external network.

- **ras|cs** - H.323 stack layer specified for processing.
- **remote_addr** - address of external network, its connections will be processed.
- **remote_port** - port, its outgoing connections will be processed. If port not specified then used value 1719 for ras and value 1720 for cs.
- **local_addr** - LAN host address, its outgoing connections will be processed. If address not specified then any port connections will be processed.
- **local_port** – a port outgoing messages from which will be processed. If the port is not specified, the all connections from all ports are processed.

nat proxy_rule parameter value [parameter value]…

Redirection of outgoing packets. TCP packets outgoing from LAN to any address with specified port, redirected тщ specified server and port. Optionally initial destination address may be included into the packet using several ways. Command line consists of word pairs: key parametert and its value.

Allowed parameters:

type encode_ip_hdr | **encode_tcp_stream** | **no_encode**

If transparent gateway requires information of initial address and an access port of a new server, then it may be done in two following ways:

- If option **encode_ip_hdr** specified then original address and port are transmitted in extended IP header fields (IP option).
- If option **encode_tcp_stream** specified, then original port and address are transmitted in a packet before data start in format "DEST IP port".

port portnum

Only packets sent to specified port are processed.

server host[:portnum]

Mandatory parameter. Specifies server address and port for packet redirection. If port not specified then original destination port will be used.

proto tcp | udp

If specified then only packets with specified protocol will be processed.

src IP[/bits]

dst IP[/bits]

Non-mandatory parameter. Specifies source/destination net (subnet) for packet redirection.

Example:

nat proxy_rule proto tcp port 80 server 123.1.1.1:3128

In given example all outgoing LAN TCP packets destined for port 80 will be redirected to provider proxy server.

nat del rule_number

Deletes the rule numbered by **rule_number**.

NAT and H.323 telephony

Subscribers and gatekeepers use several H.323 protocols. We are interested in two. RAS (registration, admission, status) used for subscriber registration on the gatekeeper and to monitor subscriber status. CS (call signaling) used by subscribers for signaling established for a spefic call. Both these protocols described *H.225.0* standard. Well known system configurations includes the following examples:

1. A subscriber resides in LAN, and a gateway has a public IP-address. Subscriber calls are outgoing only. For setting subscriber access from LAN to the gateway one may use **h323_destination** rule using CS protocol. If the gateway accepts call incoming to 1720 well-known port, there will be enough to include **default_h323** mode.

Example 1. Subscriber resides in LAN, with address 10.0.0.99; gateway is in Internet with address 123.45.67.89. Requirement: enable to subscriber outgoing calls to gateway. Solution:

nat h323_destination cs 123.45.67.89 10.0.0.99

Example 2. Subscriber resides in LAN, with address 10.0.0.99; gateway or several gateways - in Internet with unknown addresses. Requirement: enable to subscriber outgoing calls to gateway. Solution:

nat default_h323

2. Several subscribers reside in LAN, a gateway has a public IP address, calls are both incoming and outgoing.

For access from the gateway to the subscribers one should use *redirect_port* command with specified protocol *cs*, different alias addresses or ports and also directly specify gateway port and address (subscriber ports may be specified as well).

Example: subscribers reside in LAN having addresses 10.0.0.98 and 10.0.0.99; gateway resides in Internet having address 123.45.67.89. NAT alias_address is 123.45.67.65. It is necessary for the subscribers to make outgoing calls to the gateway and receive incoming calls from the gateway. Solution:

nat redirect_port cs 10.0.0.98:1720 1720 123.45.67.89

nat redirect_port cs 10.0.0.99:1720 1721 123.45.67.89

Gateway configuration should have the following subscribers' addresses: 123.45.67.65:1720 and 123.45.67.65:1721 respectively.

3. Subscriber resides in LAN, gets registered on the gatekeer with public IP address and works via gatekeeper.

In this case there will be enough to specify a command **h323_destination** ras gatekeeper_address. If the subscribers make registration on standard port 1719 then one can simply enable mode **default_h323**.

Example 1: subscriber resides in LAN having address 10.0.0.99 and gatekeeper resides in Internet having address 123.45.67.89. It is necessary for the subscriber to get registered on this gatekeeper, for making and receiving calls. Solution:

nat h323_destination ras 123.45.67.89 10.0.0.99

Example 2: several subscribers reside in LAN and gatekeeper in Internet having address 123.45.67.89 and non RAS standard port 1024. It is necessary for any subscriber to get registered on this gatekeeper for making and receiving calls. Solution:

nat h323_destination ras 123.45.67.89:1024

Example 3: subscriber resides in LAN having address 10.0.0.99 and gatekeeper or several gatekeepers reside in Internet with unknown addresses. It is necessary to get registered on unknown addresses. Solution:

nat default_h323

4. Subscriber with private IP address gets registered on the gatekeeper from LAN.

Here one should specify **redirect_port** rule with ras protocol specified and to specify here its private IP and gatekeeper RAS port. This should be done to enable for subscribers from Internet to be registered on this gateekeeper. Since static subscribers also should work with the gatekeep, one should specify **redirect_port** rule with protocol CS and with private gatekeeper IP-address and its port.

Example 1: Subscriber resides in Internet having address 123.45.67.89, and gatekeeper resides in LAN having address 10.0.0.99. It is necessary for subscriber registered on this gatekeeper for making and receiving calls. NAT alias_address is 123.45.67.65. Solution:

nat redirect_port ras 10.0.0.99:1719 1719 123.45.67.89

RAS gatekeeper address: 123.45.67.65:1719.

Example 2: Static subscriber resides in Internet having address 123.45.67.89 and gatekeeper resides in LAN having address 10.0.0.99. NAT alias_address 123.45.67.65. Solution:

nat redirect_port cs 10.0.0.99:1720 1720 123.45.67.89

In subscriber configuration gatekeeper address should be: 123.45.67.65:1720.

14.Trapd command (SNMP trapd support)

SNMP trapd support module

Syntax:

trapd agent X.X.X.X dstaddr X.X.X.X

trapd start | stop

Description:

SNMP protocol allows a network agent to send asynchronous traps when some specific event occurs on the controlled device (object).

Trapd module performs a centralized information delivery from internal router subsystems to the configured SNMP server.

SNMP server address is set by "**trapd dstadd X.X.X.X**" command (UDP port 162). Agent's own address, which is set in SNMP-trap packet, is defined by "**trapd agent X.X.X.X**" command. 127.0.0.1 address by default.

Example:

trapd dstaddr 192.168.1.1

trapd start

15.DHCP Server

DHCP Server Command Language

Commands used for configuration/review of current DHCP server state are entered using console or Telnet. Prefix command for WANFleX command interpreter is **dhcpd.**

Full command list (without prefix command):

Syntax:

add scope <SCOPE_NAME> <INTERFACE|> <START_IP> <END_IP>*
add virtual interface <GATEWAY>
clear
delete option <OPTION_NAME>
delete scope <SCOPE_NAME>
delete virtual interface <GATEWAY>
interface <INTERFACE> delete option <OPTION_NAME>
interface <INTERFACE> option <OPTION_NAME> <OPTION_VALUE>
interface <INTERFACE> reservation
<CLIENT_ID> delete option <OPTION_NAME>
interface <INTERFACE> reservation
<CLIENT_ID> option <OPTION_NAME> <OPTION_VALUE>
interface <INTERFACE|> show boundhistory*
interface <INTERFACE|> show client <CLIENT_ID|*>*
lock interface <INTERFACE>
option <OPTION_NAME> <OPTION_VALUE>
scope <SCOPE_NAME> add classid <CLIENT_CLASS_ID>
scope <SCOPE_NAME> add exclude <START_IP> <END_IP>
scope <SCOPE_NAME> add reservation <CLIENT_ID> <CLIENT_IP>
scope <SCOPE_NAME> delete classid <CLIENT_CLASS_ID>
scope <SCOPE_NAME> delete exclude <START_IP>
scope <SCOPE_NAME> delete option <OPTION_NAME>
scope <SCOPE_NAME> delete reservation <CLIENT_ID>
scope <SCOPE_NAME> interface <INTERFACE|>*
scope <SCOPE_NAME> option <OPTION_NAME> <OPTION_VALUE>
scope <SCOPE_NAME> reservation
<CLIENT_ID> delete option <OPTION_NAME>

scope <SCOPE_NAME> reservation
<CLIENT_ID> option <OPTION_NAME> <OPTION_VALUE>
scope <SCOPE_NAME> set range <START_IP> <END_IP>
scope <SCOPE_NAME|> show declinehistory*
show config
show interface <INTERFACE|>*
show options
show scope <NAME|>*
show unleases <SUBSTR|>*
show version
start
stop
unlock interface <INTERFACE>
virtual interface <GATEWAY> add
subnet <IP_ADDRESS> <SUBNET_MASK>
virtual interface <GATEWAY> delete
subnet <IP_ADDRESS> <SUBNET_MASK>

Commands are not case-sensitive and can be shortened unless ambiguity appears.

For example, **dhcpd show scope *** command can be shorted to **dhcpd s s** *, in its turn **dhcpd show config** - to **dhcpd sh c**. The commands which change DHCP configuration (including "**stop**" and "**start**" commands) can be executed only by administrator with *super-user* rights. Other commands can be executed by any user.

In above command list parameters are put into <>. If parameter value contains spaces, this parameter must be put into quotes.

Example:

*#2>**dhcpd scope** MSOFT **add classid** "MSFT 5.0"*

or

*#2>**dhcpd add scope** "Micro Soft" eth0 9.1.1.201 9.1.1.250*

Attention! DHCP executes commands ONLY after its start:

dhcpd start

DHCP Client

DHCP protocol is used for (workstations and servers) TCP/IP network hosts connection parameters dynamic configuration. UDP/IP protocol is used as a transport protocol. Host which requests data for its network connection configuration (IP-address, subnet mask, default gateway etc) is called DHCP-client. IP-address is a basic configuration parameter. After client's start it sends a DHCP request over the network so it could get a lease of IP-address and other network parameters. For its identification in its request a client may use *client identifier.* In general case, client identifier is a binary set of bytes which is unique within a physical network segment to which a client is connected. If client does not provide an identifier, the server will accept client's MAC-address for network interface. Thus, in DHCP server a client is identified by its identifier and network interface from which server accepts client's requests (*client's interface).* Client's identifier (<CLIENT_ID> parameter in commands) is represented as ID:<identifier> or 01:<MAC-address of network adapter>.

Example:

```
ID:01:00:04:35:22:88:1D.
```

In its requests to the server, a client may indicate its class (class identifier). Class identifier is a string which defines one of client's properties which is common for a set of clients. For example, it can be client operating system's name. E.g. DHCP clients which work under OS Windows XP send "MSFT 5.0" as a class identifier, InfiNet Wireless IP-phones – "IW_IP_PHONE". Client's class can be used by server administrator for automatic clients' grouping in address scopes in order to conveniently assign them specific configuration parameters (options).

Address Scope

Scope is a range of IP-addresses within which a server can assign addresses to its clients. Scopes are located in a configuration database of a server and are identified by names configured by server administrator when this scope was created. Scope is created by the following command:

Syntax:

dhcpd add scope <SCOPE_NAME> <INTERFACE|>*
<START_IP> <END_IP>

here

- *SCOPE_NAME* – scope name. It is not case-sensitive and must be unique. If scope name contains spaces, server will automatically substitute them with "underscore" sign (_).

- *INTERFACE* – name of network interface with which this scope will be attached (allowed interface). If * is specified as interference, that means that this scope can be attached to all suitable network interfaces. Suitable network interface is an interface which contains a subnet of IP-addresses (aliases) that includes starting and ending IP-addresses of the scope.
- *START_IP* and *END_IP* – starting and ending IP-addresses of the scope correspondingly. When attaching to network interface, it is checked if a range of this scope does not intersect (and is not included) within another scope that might be attached to this interface. When IP-addresses are assigned to clients, only those scopes can be used which are connected to the same network interface as a client.

In any case, if a scope cannot be attached, it is not deleted.

Example:

#2> ***dhcpd add scope*** *MSOFT eth0 192.168.177.20 192.168.177.22*

[eth0] <192.168.177.12> (MSOFT):
192.168.177.20-192.168.177.22 Scope attached

OK

In the example, we created a scope with MSOFT as a name and for suitable interface **eth0**.

#2> ***dhcpd add scope*** *new * 10.12.12.30 10.12.12.50*

WRN: Scope created, but not attached.

Here a scope with **new** name was created to be attached to any suitable interface. A scope was successfully created but could not find a suitable interface to be attached to.

In order to change a range of addresses of existing scope one can use the following command.

Syntax:

dhcpd scope *<SCOPE_NAME>* ***set range*** *<START_IP> <END_IP>*

where

- *SCOPE_NAME* – scope name which range we change
- *START_IP* and *END_IP* – new starting and ending IP-addresses of a scope correspondingly

In order to change an interface for the scope one can use the following command.

Syntax:

scope *<SCOPE_NAME>* ***interface*** *<INTERFACE|*>*

where

- *SCOPE_NAME* – scope name which interface we change
- *INTERFACE* – name of the network interface to which a scope is attached to. If a system does not have an interface with specified name or a system cannot attach this scope to specified interface, the scope will be immediately detached. This feature can be used for temporary shutdown of one of the scopes.

Example:

#2> dhcpd scope OTHER interface -eth0

```
[eth0] <192.168.177.12> (OTHER):
        192.168.177.10-192.168.177.19   Scope detached
        OK
```

Thus, we detached **OTHER** scope. In order to attach it again we need the following command:

```
#2> dhcpd scope OTHER interface eth0 (или *)
        [eth0] <192.168.177.12> (OTHER):
        192.168.177.10-192.168.177.19   Scope attached
        OK
```

One can set up **excludes** into scope range of addresses. Excludes are range of addresses which belong to the scope but are not given to DHCP server clients. The following command should be used:

Syntax:

dhcpd scope *<SCOPE_NAME>* ***add exclude*** *<START_IP> <END_IP>*

where

- *SCOPE_NAME* – scope name to which we add excludes
- *START_IP* and *END_IP* – starting and ending addresses of an exclude. Exclude's range should not intersect (or belong) with any of previous excludes assigned to this scope. Exclude's range should belong to the scope. To delete an exclude, one should do the following:

Syntax:

dhcpd scope *<SCOPE_NAME>* ***delete exclude*** *<START_IP>*

This command's parameters are identical to the command for exclude configuration besides the fact that here one can specify only starting address of an exclude to be deleted.

Attention! When executing command **dhcpd scope <SCOPE_NAME> set range <START_IP> <END_IP>**, excludes which were created before range changing and which stop satisfying conditions described above, will be deleted automatically.

Clients class filter (CLASSID)

Scope of addresses has clients class filter. If a client in its request submits its class, a server is able to give an IP-address only from those scopes which are connected to client's interface and which have client's class specified in their class filter. Class filter is a set of **client vendor class id** from which it is allowed to give a lease for IP-addresses from the scope. In order to create a class filter for a scope, one should add one or more **client vendor class id**. To add a client vendor class id to the scope, the following command is used:

Syntax:

scope *<SCOPE_NAME>* ***add classid*** *<CLIENT_CLASS_ID>*

where

- *SCOPE_NAME* – name of the scope to which client vendor class id is added (CLIENT_CLASS_ID)
- *CLIENT_CLASS_ID* – a set of characters of variable length (up to 255 characters). If this parameter contains spaces it should be specified in quotes. This <CLIENT_CLASS_ID> is compared to what client submits when requests for IP-address lease. If client submitted a class which does not present in any of scope's filters or a client did not submit any class name, only scopes with no class filters can be used for IP-address lease.

In order to delete a class from the filter, the following command is used:

Syntax:

scope *<SCOPE_NAME>* ***delete classid*** *<CLIENT_CLASS_ID>*

Network interfaces (INTERFACE)

Network interface – physical or VLAN network adaptor registered in OS WANFlex core. After its start, the server automatically detects all network interfaces which are suitable for serving DHCP clients. Suitable interface is an interface connected to a multiple-access network with broadcast support (including VLAN support). In server database each interface is identified by its name which was assigned to it while registration in WANFlex OS core. In order to review all interfaces, use the following command:

Syntax:

show interface *<INTERFACE|*>*

where

- *INTERFACE* – network interface name which information is required. If * is specified instead of interface name, all interfaces' information is printed. Command output is a structured list::

Example:

```
#2> dhcpd show interface *
        >INTERFACES
        [eth0] UP
         <SUBNET> 9.1.1.100/255.255.255.0
              <SCOPE> (PHONES) 9.1.1.151 - 9.1.1.200
         <SUBNET> 192.168.177.12/255.255.255.0
              <SCOPE> (OTHER) 192.168.177.10 - 192.168.177.19
              <SCOPE> (MSOFT) 192.168.177.20 - 192.168.177.22
        [vlan0] DOWN
         <SUBNET> 192.168.178.1/255.255.255.0
        OK
```

From this example it is seen that two network interfaces (**eth0** and **vlan0**) are served. ***eth0*** is turned on (UP) and it has two IP-subnets. To one of the subnets we can see a scope PHONES connected. To another subnet: OTHER and MSOFT. None of the scopes can be connected to **vlan0** interface as it was turned off by the administrator (DOWN).

If required it is possible to lock one or several interfaces – in this case they cannot be used. Command is the following:

Syntax:

lock interface <INTERFACE>

where

- *<INTERFACE>* - interface name. When locking interface, all attached scopes will be detached. Other scopes cannot be attached to the interface while it is locked.

Example:

```
#2> dhcpd show interface *
        >INTERFACES
        [eth0] UP
         <SUBNET> 9.1.1.100/255.255.255.0
              <SCOPE> (PHONES) 9.1.1.151 - 9.1.1.200
         <SUBNET> 192.168.177.12/255.255.255.0
```

```
        <SCOPE> (OTHER) 192.168.177.10 - 192.168.177.19
        <SCOPE> (MSOFT) 192.168.177.20 - 192.168.177.22
    [vlan0] DOWN
     <SUBNET> 192.168.178.1/255.255.255.0
    OK
```

In this example DHCP server has two interfaces: eth0 and vlan0. vlan0 interfaces was turned down by WANFlex command: **ifconfig vlan0 down**. Eth0 is turned on and we see three scopes attached to it: phones, other and msoft. PHONES is attached to 9.1.1.100/255.255.255.0 subnet, two others - to 192.168.177.12/255.255.255.0 subnet. Imagine that we want lock eth0 interface:

Example:

```
#2> dhcpd lock interface eth0
    [eth0] <9.1.1.100> (PHONES):
      9.1.1.151-9.1.1.200   Scope detached
    [eth0] <192.168.177.12> (OTHER):
      192.168.177.10-192.168.177.19   Scope detached
    [eth0] <192.168.177.12> (MSOFT):
      192.168.177.20-192.168.177.22   Scope detached
    OK
```

After locking, let us see interfaces information again:

```
#2> dhcpd show interface *
    >INTERFACES
    [eth0] UP LOCKED
     <SUBNET> 9.1.1.100/255.255.255.0
     <SUBNET> 192.168.177.12/255.255.255.0
    [vlan0] DOWN
     <SUBNET> 192.168.178.1/255.255.255.0
    OK
```

Now eth0 interface is locked and it had all his scopes detached.

Interface can be unlocked:

Syntax:

```
dhcpd unlock interface <INTERFACE>
```

Example:

```
#2> dhcpd unlock interface eth0
    [eth0] <192.168.177.12> (MSOFT):
      192.168.177.20-192.168.177.22   Scope attached
    [eth0] <192.168.177.12> (OTHER):
      192.168.177.10-192.168.177.19   Scope attached
    [eth0] <9.1.1.100> (PHONES):
      9.1.1.151-9.1.1.200   Scope attached
    OK

#2> dhcpd show interface *
    >INTERFACES
    [eth0] UP
     <SUBNET> 9.1.1.100/255.255.255.0
        <SCOPE> (PHONES) 9.1.1.151 - 9.1.1.200
     <SUBNET> 192.168.177.12/255.255.255.0
        <SCOPE> (OTHER) 192.168.177.10 - 192.168.177.19
        <SCOPE> (MSOFT) 192.168.177.20 - 192.168.177.22
    [vlan0] DOWN
     <SUBNET> 192.168.178.1/255.255.255.0
    OK
```

Scope reservation

The target of scope reservation is to reserve an IP-address for a specific client. The command is the following:

Syntax:

dhcpd scope *<SCOPE_NAME>* ***add reservation*** *<CLIENT_ID> <CLIENT_IP>*

where

- *SCOPE_NAME* – name of the scope to which reservation is added,
- ***CLIENT_ID*** – client identifier,
- ***CLIENT_IP*** – IP-address which will be given to this client. Scope reservations are saved in configuration database of the server and are identified by scope name and client's identifier.

Example:

*#2>**dhcpd scope** PHONES **add reservation** ID:01:00:04:35:00:22:23 9.1.1.170*
OK

Thus if a client ID:01:00:04:35:00:22:23 sends a request to the interface with attached scope PHONES, the server will definitely give this client 9.1.1.170 address. IP-address of the reservation must be within a scope range. Excludes does not affect the reservation. If you add a reservation and another registration for the same client exists in another pool, new reservation will not be created and the user will see an error message.

#1> dhcpd scope other add reservation ID:01:00:04:35:00:22:23 192.168.177.10

[eth0] <192.168.177.12> (OTHER): 192.168.177.10-192.168.177.19 Reservation for "ID:01:00:04:35:00:22:23" already exists in scope PHONES with IP=9.1.1.170

ERR: Reservation's IP is out of scope's range

Moreover, reservation does not obey class filtering rules.

Example:

```
#2> dhcpd show scope *
          >SCOPES:
          (MSOFT)         192.168.177.20  - 192.168.177.22  [eth0]
          ATTACHED [eth0] <192.168.177.12>/255.255.255.0
           <CLIENT CLASS IDs>: "IW_BRI_GATEWAY" "MSFT 5.0"
           <CLIENT> ID:01:00:C0:DF:10:AF:69 "MSFT 5.0"      'wad     '
          192.168.177.20  <BOUND>    since 01/01/2003 05:01:08
           <FREE RANGE>   192.168.177.21  - 192.168.177.22   =2


          (NEW)           10.12.12.30     - 10.12.12.50     [*]

          (OTHER)         192.168.177.10  - 192.168.177.19  [eth0]
          ATTACHED [eth0] <192.168.177.12>/255.255.255.0
           <CLIENT> ID:01:00:05:90:02:1F:C8 ""              '        '
          192.168.177.10  <BOUND>    since 01/01/2003 05:34:24
           <FREE RANGE>   192.168.177.11  - 192.168.177.11   =1
           <FREE RANGE>   192.168.177.13  - 192.168.177.19   =7

          (PHONES)        9.1.1.151       - 9.1.1.200       [*] ATTACHED
          [eth0] <9.1.1.100>/255.255.255.0
           <CLIENT CLASS IDs>: "IW_IP_PHONE"
           <CLIENT> ID:01:00:04:35:00:22:24 "IW_IP_PHONE"   'Stas    '
          9.1.1.151       <BOUND>    since 01/01/2003 05:00:34
           <FREE RANGE>   9.1.1.152       - 9.1.1.169        =18
```

```
 <RESERV> ID:01:00:04:35:00:22:23 "IW_IP_PHONE"   'Andrew  '
9.1.1.170       <BOUND>    since 01/01/2003 05:49:35
 <FREE RANGE>   9.1.1.171       - 9.1.1.200        =30
 <OPTION>       Router  9.1.1.3
 <OPTION>       H323_GK_ADDRESS 195.38.45.84

OK
```

Here, a client **ID:01:00:05:90:02:1F:C8** in his DHCP request did not specify his class (“”), so OTHERS scope (192.168.177.12/255.255.255.0 subnet, eth0 interface) as this scope does not have class filters. However, administrator wants this client to get his additional configuration parameters from PHONES scope. In order to do that, a reservation is created:

```
#2> dhcpd scope PHONES add reservation
                ID:01:00:05:90:02:1F:C8 9.1.1.200
          OK

#2> dhcpd show scope *
          >SCOPES:
          (MSOFT)         192.168.177.20  - 192.168.177.22  [eth0]
          ATTACHED [eth0] <192.168.177.12>/255.255.255.0
           <CLIENT CLASS IDs>: "IW_BRI_GATEWAY" "MSFT 5.0"
           <CLIENT> ID:01:00:C0:DF:10:AF:69 "MSFT 5.0"      'wad     '
          192.168.177.20  <BOUND>    since 01/01/2003 05:01:08
           <FREE RANGE>   192.168.177.21  - 192.168.177.22   =2

          (NEW)           10.12.12.30     - 10.12.12.50     [*]

          (OTHER)         192.168.177.10  - 192.168.177.19  [eth0]
          ATTACHED [eth0] <192.168.177.12>/255.255.255.0
           <FREE RANGE>   192.168.177.10  - 192.168.177.11   =2
           <FREE RANGE>   192.168.177.13  - 192.168.177.19   =7

          (PHONES)        9.1.1.151       - 9.1.1.200       [*] ATTACHED
          [eth0] <9.1.1.100>/255.255.255.0
           <CLIENT CLASS IDs>: "IW_IP_PHONE"
           <CLIENT> ID:01:00:04:35:00:22:24 "IW_IP_PHONE"   'Stas    '
          9.1.1.151       <BOUND>    since 01/01/2003 05:00:34
           <FREE RANGE>   9.1.1.152       - 9.1.1.169        =18
           <RESERV> ID:01:00:04:35:00:22:23 "IW_IP_PHONE"   'Andrew  '
          9.1.1.170       <BOUND>    since 01/01/2003 05:49:35
           <FREE RANGE>   9.1.1.171       - 9.1.1.199        =29
           <RESERV> ID:01:00:05:90:02:1F:C8 ""              '        '
          9.1.1.200       <BOUND>    since 01/01/2003 06:22:30
           <OPTION>       Router  9.1.1.3
           <OPTION>       H323_GK_ADDRESS 195.38.45.84

          OK
```

If reservation is no more required, you can delete it:

Syntax:

dhcpd scope <SCOPE_NAME> delete reservation <CLIENT_ID>

If a client acquired its IP-address, after reservation deletion a server will hold a lease of this address to this client if a client does violate scope’s rules (excludes and class filters).

Example:

```
#1> dhcpd scope phones delete
               reservation ID:01:00:05:90:02:1F:C8
          OK
          #1> dhcpd show scope *
          >SCOPES:
          (MSOFT)         192.168.177.20  - 192.168.177.22  [eth0]
          ATTACHED [eth0] <192.168.177.12>/255.255.255.0
           <CLIENT CLASS IDs>: "IW_BRI_GATEWAY" "MSFT 5.0"
           <CLIENT> ID:01:00:C0:DF:10:AF:69 "MSFT 5.0"      'wad     '
          192.168.177.20  <BOUND>    since 01/01/2003 01:01:08
           <FREE RANGE>   192.168.177.21  - 192.168.177.22   =2

          (NEW)           10.12.12.30     - 10.12.12.50     [*]

          (OTHER)         192.168.177.10  - 192.168.177.19  [eth0]
          ATTACHED [eth0] <192.168.177.12>/255.255.255.0
```

```
          <CLIENT> ID:01:00:05:90:02:1F:C8 ""              '        '
         192.168.177.10  <BOUND>    since 01/01/2003 01:16:36
          <FREE RANGE>   192.168.177.11  - 192.168.177.11   =1
          <FREE RANGE>   192.168.177.13  - 192.168.177.19   =7

         (PHONES)        9.1.1.151       - 9.1.1.200       [*] ATTACHED
         [eth0] <9.1.1.100>/255.255.255.0
          <CLIENT CLASS IDs>: "IW_IP_PHONE"
          <CLIENT> ID:01:00:04:35:00:22:24 "IW_IP_PHONE"   'Stas    '
         9.1.1.151       <BOUND>    since 01/01/2003 01:01:47
          <FREE RANGE>   9.1.1.152       - 9.1.1.169        =18

 <RESERV> ID:01:00:04:35:00:22:23 "IW_IP_PHONE"   'Andrew  ' 9.1.1.170
         <BOUND>    since 01/01/2003 01:01:37
          <FREE RANGE>   9.1.1.171       - 9.1.1.200        =30
          <OPTION>       Router  9.1.1.3
          <OPTION>       H323_GK_ADDRESS 195.38.45.84

         OK
```

In this example after the reservation was deletes, the server cancelled a lease for **ID:01:00:05:90:02:1F:C8** client for IP-address 9.1.1.2000 in PHONES scope because client's class does not fulfill class filter requirements in the scope. After some time, the same client obtained another IP-address from OTHER scope.

```
#1> dhcpd scope phones delete
                  reservation ID:01:00:04:35:00:22:23
         OK
         #1> dhcpd show scope phones
         >SCOPES:
         (PHONES)        9.1.1.151       - 9.1.1.200       [*] ATTACHED
         [eth0] <9.1.1.100>/255.255.255.0
          <CLIENT CLASS IDs>: "IW_IP_PHONE"
          <CLIENT> ID:01:00:04:35:00:22:24 "IW_IP_PHONE"   'Stas    '
         9.1.1.151       <BOUND>    since 01/01/2003 01:01:47
          <FREE RANGE>   9.1.1.152       - 9.1.1.169        =18
          <CLIENT> ID:01:00:04:35:00:22:23 "IW_IP_PHONE"   'Andrew  '
         9.1.1.170       <BOUND>    since 01/01/2003 01:01:37
          <FREE RANGE>   9.1.1.171       - 9.1.1.200        =30
          <OPTION>       Router  9.1.1.3
          <OPTION>       H323_GK_ADDRESS 195.38.45.84

         OK
```

ID:01:00:04:35:00:22:23 client did not have his lease cancelled (9.1.1.170 address) because this client fulfills all scope's rules.

Attention! When executing **dhcpd scope <SCOPE_NAME> set range <START_IP> <END_IP>** command, all reservations that stop fulfilling scope's range of addresses will be deleted automatically.

Configuration options

Configuration options are parameters which clients might request from the server for more precise host configuration. These parameters are *Address Time*, *Router*, *NTP Servers* etc. Clients may request a different set of these parameters. The parameters are only sent when a client included them in its request and only when server knows the value of the parameter. Divisions and values of the parameters are defined while DHCP server configuration. Divisions can be defined for the following purposes:

1. Scope reservation. Options values from this division will be given to the client of this reservation.
2. Interface reservation. Options are sent if requested option's value is not in scope's reservation divisions.
3. Scope. Option values from this division can be sent to the client who received an address lease from this scope only if the option requested by the client is not in scope's or interface's reservation division.
4. Interface. Sent to the client which received a lease from one of the scopes which is attached to the interface (and the value of the requested option was not in scope's reservation, in the scope itself and in interface's reservation).

5. Server. Sent to clients which received a lease from one of the scopes (if the value of the option was not in all divisions listed above)? Meaning of the division – default value.

If option's value does not exist in all divisions, client does not receive anything from the server. Two exceptions are possible:

- *Address Time* – the value of this parameter is ALWAYS sent to the client. If this value is not specified in all divisions, the client receives a default value of 120 (lease time – 2 minutes).
- *Subnet Mask* – the value of this parameter is ALWAYS sent to the client. The value of this option is automatically determined by the server and it cannot be defined in options divisions while server configuration. The value of the subnet mask for the client always equals subnet mask of the interface to which the scope is attached (this scope gave a lease to the client)

DHCP configuration options (overall table) is available using the following link: http://www.iana.org/assignments/bootp-dhcp-parameters

To define a set of options, DHCP server has special commands for each division. These commands have parameters, which are inputted in a common way (for all divisions):

OPTION_NAME – name of the option (see the link for the table above). If option name has spaces, they must be substituted with "_" sign. Option name is not case-sensitive.

OPTION_VALUE – value of the option. Input format depends on the purpose of the option and is divided into three categories by DHCP server:

1. Symbolic. A string (e.g. for *Bootfile-Name option).* If this option's value has spaces, the option value should be put in quotes.
2. Binary. One or several decimal numbers. If several numbers should be specified, they are separated by commas. Options examples: *Address Time, Time Offset.*
3. IP-address. One or several values – IP-addresses. Several IP-addresses are separated by commas.

Commands for defining/adding options for different divisions:

1. Scope reservation division

Syntax:

dhcpd scope <SCOPE_NAME> reservation
<CLIENT_ID> option <OPTION_NAME> <OPTION_VALUE>

where

- *SCOPE_NAME* – scope name for which reservation one need to define an option value.
- ***CLIENT_ID –*** reservation client identifier. If this option with the same name was defined, the value will be changed to the one specified in this command.

2. Interfaces reservations division

Syntax:

dhcpd interface <INTERFACE> reservation
<CLIENT_ID> option <OPTION_NAME> <OPTION_VALUE>

where

- ***INTERFACE –*** name of the interface where client's (CLIENT_ID) reservation is created. If this interface did not have a reservation for this client, this command will automatically create this reservation and will add it to the options set.

Interfaces reservations are required for specific settings for the client no matter from what scope the client is getting his address lease. Interface reservation is different from scope reservation in two parameters:

- *Does not define a fixed IP-address for the client.* Thus it takes for the server to dynamically define from which scope and which IP-address is to be given to the client.
- *Allows changing client's class.* If *Class ID* option is defined for the interfaces reservation, the class will be changed for the option's value when a client from this reservation sends a request. It becomes necessary when DHCP client does not send its class.

Creating interface reservation does not contradict with scope reservation for the same client.

3. Scope divisions

Syntax:

```
dhcpd scope <SCOPE_NAME>
            option <OPTION_NAME> <OPTION_VALUE>
```

4. Interface divisions

Syntax:

```
dhcpd interface <INTERFACE>
          option <OPTION_NAME> <OPTION_VALUE>
```

5. Server divisions

Syntax:

```
dhcpd option <OPTION_NAME> <OPTION_VALUE>
```

Of course, there is a set of commands which delete all of these options from the divisions:

Syntax:

```
dhcpd scope <SCOPE_NAME>
            reservation <CLIENT_ID> delete option <OPTION_NAME>

dhcpd scope <SCOPE_NAME> delete option <OPTION_NAME>

dhcpd interface <INTERFACE>
            reservation <CLIENT_ID> delete option <OPTION_NAME>

dhcpd interface <INTERFACE> delete option <OPTION_NAME>

dhcpd delete option <OPTION_NAME>
```

One should pay a great deal of attention to the deletion of interfaces reservation division options. If, after deletion, it turns out that options set for this reservation is empty, the interface reservation will be deleted automatically.

Not all of the options can be defined in any division. Apart from *Subnet Mask* (was described above), there are options which can be defined for *some particular* divisions.

Example:

```
#1> dhcpd scope phones option class_id "TestClass"
```

ERR: This option cannot contain in the given division.

Moreover, there is a set of service options which although are included into a summary table, they do not act as configuration parameters but act as service parameters. The list of service options of DHCP server looks as follows:

- *Subnet Mask*
- *Address Request*
- *Overload*
- *DHCP Msg Type*
- *DHCP Server Id*
- *Parameter List*
- *DHCP Message*
- *DHCP Max Msg Size*
- *Client Id*

If you attempt to add one of these options to any division, the server will report an error: **ERR: This option cannot contain in the given division**.

To control options which were requested by the client and given to him, one can use the following command:

Syntax:

dhcpd interface <INTERFACE|> show client <CLIENT_ID|*>*

where

- ***INTERFACE –*** name of a network interface which information is requested
- ***CLIENT_ID*** – client's identifier, which information is requested. Instead of interface name one can specify "*****": this will print information for all clients and interfaces. Instead of client's identifier it is permitted to specify "*****": this will print information about all clients for the specified interface. The information is shown only for clients with given address lease from one of the scopes which is attached to the specified interface.

Example:

```
#2> dhcpd interface * show client *
          >INTERFACES CLIENTS
          --------- [eth0] ---------
          (IPHONES) <CLIENT> ID:01:00:04:35:00:22:24 "IW_IP_PHONE" 'Unknown node'
          192.168.0.101   <BOUND>    since 25/04/2005 11:32:57
          SUPPLIED OPTIONS:
          #1       .  .  .  .  . DF Subnet Mask              255.255.255.0
          #2       .  .  .  .  .  . Time Offset              <not supplied>
          #3       .  .  S  .  .  . Router                   192.168.0.1
          #7       .  .  .  .  .  . Log Server               <not supplied>
          #42      .  .  S  .  .  . NTP Servers              192.168.0.1
          #230     .  .  S  .  .  . H323 GK ADDRESS          192.168.0.1
          #231     . IR  .  .  .  . H323 LOGIN ALIAS         IWPhone/V. Pupkin/101
          #232     .  .  .  .  .  . H323 GK ID               <not supplied>
```

Here, the list of client's supplied options consists of records (strings) which contain a number (#<N>) of a supplied option, a map of server's divisions from which this option was supplied to a client (if was supplied), name of the option and its value. If a requested option was not defined in any of server's divisions, it is diplayed as <not supplied> in the list. On the map the divisions are displayed using the following indication:

1. SR – scope reservation division
2. IR – interface reservation division
3. S – scope reservation
4. I – client's interface division
5. SV – server's division

Moreover, the options which were requested by clients and supplied to them but which were not defined in any division (e.g. *Subnet Mask)* are marked as DF.

Address Time

Any IP-address lease is limited by the time specified in *Address Time* option. If a client which was given a lease does not extend it within *Address Time* period, the server will cancel the lease. The value of this time may be defined by the client but it should not exceed its maximal value. The maximal time of a lease is set up in *Address Time* of one of the divisions to which this client is applied. If a server does not have this option defined, the maximal time will be set to 120 seconds. In case if a client does not request *Address Time* parameter, the server will give a lease for a maximal time according to the scheme described above.

A client which received a lease, confirms it periodically. The periodicity is usually equal to the half of *Address Time.* As an acknowledgement to the lease prolongation the server resends configuration parameters (options). Thus, if during the lease some of the options were changed in the server (or division to which this client was applied) the client will learn it in the moment of lease prolongation.

If after lease expiration the client does not confirm it, the scope cancels the lease. If the client is not a scope reservation client, the scope will mark the IP-address of this lease as "conditionally free". On scope state output (dhcpd show scope *) this state will be marked as <OBIND>. Thus, with other addresses available for lease, the scope will not give <OBIND> addresses for new clients. If during 24 hours from the moment of lease expiration the client will request for a lease again, the server will give him the same IP-address.

```
#1> dhcpd show scope MSOFT
        >SCOPES:
        (MSOFT)       192.168.177.20  - 192.168.177.22  [eth0] ATTACHED
        [eth0] <192.168.177.12>/255.255.255.0
         <CLIENT CLASS IDs>: "IW_BRI_GATEWAY" "MSFT 5.0"
         <CLIENT> ID:01:00:C0:DF:10:AF:69 "MSFT 5.0"     'wad     '
        192.168.177.20  <BOUND>    since 01/01/2003 01:01:14
         <O_BIND> ID:01:00:0F:EA:05:29:C6 "MSFT 5.0"
        'win2k3sbs' 192.168.177.21  <OBIND>
         <FREE RANGE>   192.168.177.22  - 192.168.177.22   =1
        OK
```

At the same time, the scope writes down the parameters of expired lease into a special database (boundhistory).

```
#1> dhcpd interface eth0 show boundhistory

[eth0]

>BOUND_HISTORY 1

(MSOFT) ID:01:00:0F:EA:05:29:C6 BOUND=192.168.177.21     until
        02/01/2003 13:25:37

OK
```

The information about expired leases is saved in the database during 24 hours. After 24 hours the record is automatically deleted from the database, and the IP-address becomes a free address (after being <OBIND>).

The server will use <OBIND> addresses for other clients if all the scopes (which suit new clients) ran out of free addresses. The server will use the oldest records in "boundhistory" in the first turn.

The server will also cancel an address lease after a client's corresponding request.

Admissibility check for IP-addresses lease

The check is made in order to avoid IP-addresses conflicts. After the server detected the IP-address as being free, it will perform an admissibility check prior to IP-address lease to the client. In other words, the server makes sure that this IP-address is not occupied by any host (except, may be, for the target client itself) on the client's interface. The server makes ARP-requests on the client's interface. If no one answered the request (may be except for the target client), the IP-address will be given for a lease.

This check is performed in any case except for case of virtual interfaces when the check is a client's responsibility.

If IP-addresses conflict is detected, this IP-address will not be given for a lease. The server will attempt to give a next free IP-address. If, eventually, there is no free IP-address left, the server looks into *boundhistory* for the client's interface. If this step failed, the server puts this client into a database of unleases.

Unleases

Clients to which DHCP server failed to give an IP-address for a lease are put to a special list – unleases. The records in this list are saved for 15 minutes if a client does not repeat an attempt to get a lease. Each record in the list consists of the following fields:

1. Name of a network interface from which a client's request for a lease was received (client's interface).
2. Client's identifier
3. Client's class identifier
4. Host name

To view the list, use the following command:

Syntax:

dhcpd show unleases <SUBSTR|>*

where

- ***SUBSTR*** – a substring for a partial list view. When executing a command the server will print only those records which fields contain the substring (one of the fields). Substring is case-sensitive. If * is specified as a substring the full list is printed.

Example:

```
#1> dhcpd show unleases *
        >UNLEASES 1
        eth0    ID:01:00:C0:DF:10:AF:69 "MSFT 5.0"      wad
        OK
```

Virtual interfaces

After their start, DHCP clients send broadcast request in order to get an IP-address lease. As a client at this time does not yet have an IP-address the server also uses broadcast packets to communicate with a client. It is known that broadcast packets are not routed and, thus, the dialog between DHCP server and DHCP client can occur only within one network (physical network). If DHCP server is connected to another network, the direct dialog cannot take place. However, the router which logically connects two networks with DHCP client and DHCP server can have a special software running – DHCP Relay Agent (DRA). DRA retranslates DHCP packets (including broadcast packets) from DHCP clients to DHCP server and back. Data exchange between DRA and DHCP server is performed using unicast packets only. Thus, DRA and DHCP must know each other's IP-addresses starting from their configuration stage. For this purpose DHCP server has *virtual interfaces.* In fact DHCP-server virtual interface is a physical network interface placed in DRA. As DHCP does not know this interfaces

subnets sets, one should specify these subnets while virtual interfaces configuration.

To create virtual interface, use the command:

Syntax:

dhcpd add virtual interface <GATEWAY>

where

- **GATEWAY** – IP-address of DRA which has a corresponding physical interface. After executing this command, one more interface is created in server's configuration with a name formed from DRA's IP-address: v.GATEWAY. Example: v.192.168.177.81

Example:

```
#1> dhcpd add virtual interface 192.168.177.81
        [v.192.168.177.81]
          Virtual interface v.192.168.177.81 added
        OK

#1> dhcpd show interface *
        >INTERFACES
        [eth0] UP
         <SUBNET> 9.1.1.100/255.255.255.0
              <SCOPE> (PHONES) 9.1.1.151 - 9.1.1.200
         <SUBNET> 192.168.177.12/255.255.255.0
              <SCOPE> (MSOFT) 192.168.177.20 - 192.168.177.50
         <SUBNET> 192.168.15.55/255.255.255.0
         <RESERVATION>  for ID:01:00:05:90:02:1F:C8
              <OPTION>       Class_Id       "Swissvoice"
        [vlan0] DOWN
         <SUBNET> 192.168.178.1/255.255.255.0
        >VIRTUAL INTERFACES
        [v.192.168.177.81] UP
        OK
```

In server's configuration we can observe one more interface with v.192.168.177.81 name. Working with this interface is no different from other interfaces. However, before a scope is attached to it, one should configure a set of subnets. The following command can be used:

Syntax:

dhcpd virtual interface <GATEWAY> add subnet <IP_ADDRESS>
<SUBNET_MASK>

where

- ***GATEWAY*** – IP-address of DRA which corresponds to the virtual interface
- ***IP_ADDRESS*** – IP-address which DRA has for this subnet
- ***SUBNET_MASK*** – subnet mask

Example:

```
#1> dhcpd virtual interface 192.168.177.81
              add subnet 192.168.188.1 255.255.255.0
        [v.192.168.177.81]  Subnet added
                    192.168.188.1/255.255.255.0
        OK

#1> dhcpd show interface *
        >INTERFACES
        [eth0] UP
```

```
 <SUBNET> 9.1.1.100/255.255.255.0
      <SCOPE> (PHONES) 9.1.1.151 - 9.1.1.200
 <SUBNET> 192.168.177.12/255.255.255.0
      <SCOPE> (MSOFT) 192.168.177.20 - 192.168.177.50
 <SUBNET> 192.168.15.55/255.255.255.0
 <RESERVATION>  for ID:01:00:05:90:02:1F:C8
      <OPTION>      Class_Id      "Swissvoice"
[vlan0] DOWN
 <SUBNET> 192.168.178.1/255.255.255.0
>VIRTUAL INTERFACES
[v.192.168.177.81] UP
 <SUBNET> 192.168.188.1/255.255.255.0
OK
```

After that, we can create a scope of addresses from which a DHCP server can give a lease to the clients to which a DRA has an access.

Example:

```
#1> dhcpd add scope VIRTUAL_TEST
        v.192.168.177.81 192.168.188.20 192.168.188.50
    [v.192.168.177.81] <192.168.188.1> (VIRTUAL_TEST):
       192.168.188.20-192.168.188.50   Scope attached
    OK

#1> dhcpd show interface v.192.168.177.81
    >VIRTUAL INTERFACES
    [v.192.168.177.81] UP
     <SUBNET> 192.168.188.1/255.255.255.0
          <SCOPE> (VIRTUAL_TEST)
            192.168.188.20 - 192.168.188.50
    OK

#1> dhcpd show scope virtual_test
    >SCOPES:
    (VIRTUAL_TEST)  192.168.188.20  - 192.168.188.50
    [v.192.168.177.81] ATTACHED [v.192.168.177.81]
    <192.168.188.1>/255.255.255.0
     <FREE RANGE>   192.168.188.20  - 192.168.188.50   =31

    OK
```

You can delete a subnet from virtual interface's list using the following command:

Syntax:

```
dhcpd virtual interface <GATEWAY>
        delete subnet <IP_ADDRESS> <SUBNET_MASK>
```

Example:

```
#1> dhcpd virtual interface 192.168.177.81
        delete subnet 192.168.188.1 255.255.255.0
    [v.192.168.177.81] <192.168.188.1> (VIRTUAL_TEST):
        192.168.188.20-192.168.188.50   Scope detached
    [v.192.168.177.81]  Subnet dropped
        192.168.188.1/255.255.255.0
    OK

#1> dhcpd show scope virtual_test
    >SCOPES:
    (VIRTUAL_TEST)  192.168.188.20  - 192.168.188.50
    [v.192.168.177.81]
```

OK

As we deleted a subnet to which a scope was connected, the scope will be detached automatically. This scope will stay detached until an appropriate subnet is configured for v.192.168.177.81 virtual interface.

DHCP server configuration cleanup

In order to clean DHCP server configuration, it first should be stopped by *dhcp stop* command. After that, the configuration can be cleaned:

dhcpd clear

16.DHCP relay. dhcpr command

General Description

For DHCP protocol regular work, the server and the hosts that get the service should be allocated within one network segment – no routers should be placed in between. If the network consists of several segments, each segment should have its own DHCP server as routers block broadcast packets. One of the alternatives to this solution is installing in each segment that does not have the server DHCP Relay Agent which forwards the requests from network hosts to DHCP server. Some routers may also have a function of DHCP Relay.

Syntax:

dhcpr [add]|delete SERVERIP

dhcpr (flush|trace|notrace)

dhcpr (lock|unlock) INTERFACE

dhcpr (info|noinfo)

dhcpr (start|stop)

Commands Description

Start / Stop of DHCP Relay

Syntax:

dhcpr {start | stop}

This command starts / stops DHCP relay.

Example:

dhcpr start

DHCP servers listing

Syntax:

dhcpr [add]|delete SERVERIP

This command adds / deletes DHCP servers to the list for which client's requests forwarding will be made.

Example:

dhcpr add 125.12.100.12

dhcpr 125.12.100.13

dhcpr delete 125.12.100.12

Interface blocking

By default, DHCP Relay accepts client's requests from all network interfaces of Ethernet type. If one of the interfaces needs to be blocked not to forward requests from it, a special command should be used.

Syntax:

dhcpr (lock|unlock) INTERFACE

INTERFACE – a name of one or several (separated by spaces) interfaces.

Example:

dhcr lock eth0

Using "DHCP Relay agent information" option

In order to identify client's interface when receiving server's replies, the relay can use a special DHCP option which he appends to the client's request packet while relaying. Not all of DHCP server support this capability. DHCP Relay has this option turned off by default. A special command can be used to turn this feature on.

Syntax:

dhcpr (info|noinfo)

Example:

dhcpr info

17.DHCP Client. dhcpc command

General Description

DHCP client is used for automatic retrieving of different parameters from DHCP server for one or several unit's network interfaces. Among the parameters are IP-address, network mask, default gateway etc.

DHCP client management is implemented via **dhcpc** command.

Syntax:

dhcpc [options] [IFNAME] [commands]

IFNAME – name of the network interface to which **options** and **commands** are referred.

Options

Options define working parameters of DHCP client on a corresponding interface, or these options defaults if no interface name is specified. For each option special values can be specified: *none* and *default*. Option value *none* means this parameter absence for this interface even though default value of this parameter exists. Option value *default* means the absence of a specific parameter value (meaning that only default option exists). With this, default parameter value is applied if specified. *Default* option value is not displayed in DHCP client configuration.

- *-l (none|default|$ACLNAME|acl:ACLNAME)* – sets the list of IP-addresses of DHCP servers from which the client is permitted to receive parameters. Here, ACLNAME – the name of access control list (see ***acl*** command). If specified list is not configured in the system (this *acl* does not exist), DHCP client will be able to receive parameters from any DHCP server.
- *-k (none|default|key:KEYVALUE)* – sets authorization key. DHCP authorization is in accordance with *"RFC 3118 - Authentication for DHCP Messages"*.
- *-a (none|default|NUMBER)* – sets the number of repeated *arp* requests which sends DHCP client after getting a lease of IP-address from DHCP server. In accordance with DHCP, the client is obliged to check received IP-address if there are any other network devices with the same IP-address. For higher reliability, DHCP client sends a series of such request with ¼ second interval. If *arp* requests number is not specified for all of the interfaces (including absence of default value for this parameter), DHCP client sends 16 requests.
- *-t (on|off)* - This option turns on/off sending debug information to the system log. The option is not attached to any specific interface.

Commands

- *start* - starts DHCP client on a specified interface
- *stop* - stops DHCP client on a specified interface
- *delete* – stops DHCP client on a specified interface and clears all the options.
- *dump* - shows current status of DHCP client.

Examples

```
dhcpc –a 5
dhcpc –l $DHCP_SERVERS eth0 start
dhcpc –a none –k key:qwerty  rf4.0 start
```

This configuration sets the number for ARP requests of 5. For eth0 interface the list of allowed DHCP servers is specified in DHCP_SERVERS ACL. The client is started for eth0 interface. For rf4.0 interface *none* option is set for the number of ARP requests. Thus, rf4.0 will send 16 ARP requests. Also, DHCP client on rf4.0 interface will use "qwerty" as authorization key.

```
dhcpc dump
```

The command prints current status of DHCP client.

```
ID  I-face   IP address/mask   Gateway address    Server ID    Lease exp.
== ======== ================== ================ ===
 0  eth0   192.168.61.29/26    192.168.61.1    192.168.61.1 000:35:16
 1  rf4.0  ------------------ ----------
```

Here, clients are started on eth0 and rf4.0 interfaces.

For eth0 interface DHCP client obtained a lease for 192.168.61.26 IP-address with 26 bits network mask length from 192.168.61.1 DHCP server. The lease expires in 35 minutes and 16 seconds.

DHCP client on rf4.0 interface has not yet received any parameters.

18.VRRP server. VRRP command

General Description

At the present time most of the large LANs are built with a router in the center. In such LANs virtual networks with routing are usually organized, redundant connections between devices are provided and additional central router is installed. While this kind of structure might seem to be reliable, it is the central router that seems to be the vulnerability of the system. In the case of central router's malfunctioning, there might be a few scenarios, each of which would require a system administrator to interfere: workstations configuration in order they could work with other router as a gateway, changing the configuration of a redundant unit, additional router installation etc.

VRRP server is able to keep the network alive in case any of the described situations occurs In fact, the server provides with giving the "responsibilities" from one device to another if the first one fails. When VRRP server is used additional router automatically comes onto operation.

Each redundant router should be a part of virtual router (VR). For VR there is a list of its VR IP-addresses. At the time one of routers becomes a primary one it starts to serve each of this list IP-addresses (i.e. to reply on ARP-requests and takes the host functions with these IP-addresses).

VR is referred to by its identifier – the number in 1…255 range (VRID).

Hence, the logic of VRRP-server operations is the following:

1) Several VRRP routers form VR. Each of them has identical VRID and identical list of IP-addresses;

2) The main router should be selected from the list of VRRP-routers (MASTER mode). Other ones get the status of slave routers (BACKUP mode). The main router periodically sends special packets (sweeping). By receiving these packets, BACKUP routers make a decision about MASTER's availability.

3) In the case of the main router failure (there are no keep-alive-messages from MASTER for a long time) one of the slave routers becomes the main router and starts to process packets addressed to VR.

The main virtual router selecting is implemented automatically: this status gets the router with the highest priority or (in the case their equality) – with the biggest network interface IP-address.

Full syntax :

vrrp {start|stop|dump [IFNAME:VRID]}

vrrp IFNAME:VRID [start|stop|clean|flush]

vrrp IFNAME:VRID [add]|delete IPADDRESS[/(MASK|MASKLEN)] …

vrrp IFNAME:VRID [-(password|key)=[PASSWORD]]

vrrp IFNAME:VRID [-priority=[PRIO|own]] [-interval=AINT]

vrrp IFNAME:VRID [-preempt=(on|off)] [-owner=[on|off]] [-learn=(on|off)]

Command description

Server start/stop

Syntax:

vrrp {start | stop}

The command starts / stops VRRP-server.

Example:

vrrp start

Creating Virtual Router (VR)

Syntax:

vrrp IFNAME:VRID add IPADDRESS[/{MASK|MASKLEN}] …

The command creates virtual router on **IFNAME** interface with **VRID** identifier. VRID is a number in 1....255 range. Also, the command adds IP-address specified as a parameter to the list of VR IP-addresses. None of VR IP-address should coincide with the primary IP-address of interface it has been created on. VRRP-server allows creating several VRs for one network interface, but lists of their IP-addresses should not crossover.

Example:

vrrp eth0:10 add 9.8.7.6/24

VR start/stop

Syntax:

vrrp IFNAME:VRID {start|stop}

Command starts/stops this router in a specified VR.

Example:

vrrp eth0:10 start

Setting router priority

Syntax:

vrrp IFNAME:VRID -priority=[PRIO|own]

The ccommand sets the priority of the specified router in VR. Priority value varies in 2…255 range. Router priority is considered in the procedure of selecting a main router selecting. At that the router with the greatest priority becomes the main one.

Priority of 255 has a special meaning. It shows this router will be the main within specified VR. The main router with such a priority owns all of VR's IP-addresses.

Example:

vrrp eth0:10 –priority=200

Owner mode

Syntax:

vrrp IFNAME:VRID -owner=on|off

In **owner** mode the router owns all of VR's IP-addresses regardless its priority. I.e. even if this route is a slave at the moment, VR's IP-addresses are in the lists of network interface IP-addresses on which VR is created. At the same time these addresses stay in a "passive" mode. I.e. the router will not reply on these addresses until it takes the functions of the main router.

"Owner" mode is enabled by default.

Example:

vrrp eth0:10 –owner=off

Inheritance mode

Syntax:

vrrp IFNAME:VRID –preempt=on|off

If inheritance mode is disabled the router (regardless its priority) would never take the functions of the main router while there are other operating routers in VR.

Inheritance mode is enabled by default.

Example:

vrrp eth0:10 –preempt=off

Keep-alive messaging interval setting

Syntax:

vrrp IFNAME:VRID -interval=AINT

This command allows set the required keep-alive messaging interval for the main route. Parameter's value is set in seconds.

The router acting as MASTER periodically sends service packets to other VR routers. On receiving these messages BACKUP routers get the information about MASTER's availability. Default value of the parameter is 1 second.

If you set another value of this parameter you should keep in mind it has to be equal for all routers of specified VR.

Example:

vrrp eth0:10 –interval=2

Self-learning mode

Syntax:

vrrp IFNAME:VRID –learn=on|off

The mode allows a router to collect the list of VR's IP-addresses while it acts as a BACKUP router. This mode is used to simplify VRRP server configuration. You can simply make a list of VR's IP-addresses only for one router – the owner of IP-addresses (with the priority of 255). For the rest routers it is enough to create VR with an empty IP-addresses list and set up a self-learning mode.

Example:

vrrp eth0:10 –learn=on

VR deleting

Syntax:

vrrp IFNAME:VRID clean

Command deletes specified VR.

Example:

vrrp eth0:10 clean

IP-address removing from VR list

Syntax:

vrrp IFNAME:VRID delete IPADDRESS

Command deletes specified IP-address from VR-list.

Deleting VR IP-addresses list

Syntax:

vrrp IFNAME:VRID flush

Command deletes all IP-addresses from VR-list.

Example:

vrrp eth0:10 flush

VRRP authorization

According to RFC 2338 VRRP server supports two authorization modes:

1. Simple text password
2. IP Authentication Header

Enabling these authorization schemes can be done using the following commands:

vrrp IFNAME:VRID –password=PASSWORD

vrrp IFNAME:VRID –key=PASSWORD

Experience shows that neither of two VRRP authorization methods can provide absolute VR security. This fact was described in the later RFC 3768 version:

10. Security Considerations

VRRP does not currently include any type of authentication. Earlier versions of the VRRP specification included several types of authentication ranging from none to strong. Operational experience and further analysis determined that these did not provide any real measure of security. Due to the nature of the VRRP protocol, even if VRRP messages are cryptographically protected, it does not prevent hostile routers from behaving as if they are a VRRP master, creating multiple masters. Authentication of VRRP messages could have

prevented a hostile router from causing all properly functioning routers from going into backup state. However, having multiple masters can cause as much disruption as no routers, which authentication cannot prevent. Also, even if a hostile router could not disrupt VRRP, it can disrupt ARP and create the same effect as having all routers go into backup.

VRRP server state output

Syntax:

vrrp dump

Command displays VRRP-server current state.

Example:

vrrp dump

```
VRRP  interface:ID   Prio  AInterval      Master  IP        STATE           Time
Stop reason
================== ==== ========= ============== ======= ==============  ===
          eth0:010 200o    001    192.168.15.50   BACKUP      0/0:0:3:000
```

VRRP-server state is printed in a table consisting of following columns:

- *VRRP interface: ID* –displays VR in IFNAME:VRID form
- *Prio* – displays the priority of the router in a specified VR. If "owner" mode is enabled, a letter "**o**" is also printed.
- *AInterval* – displays set keep-alive messaging interval.
- *Master IP* – displays primary IP-address of MASTER router
- *STATE* – displays router's current state:
    - *MASTER*
    - *BACKUP*
    - *STOP*

    If specified router has self-learning mode enabled small **l** is printer before the state name, for example, *lBACKUP*
- *Time* – displays time period during which the route is in STATE mode. The period is represented in DAYS/HOURS:MINUTES:SECONDS:000 form
- *Stop reason* – the router stops operating in specified VR if current situation forces it to do so - for a specific VR router it changes its state to STOP. This column displays the reason of the problem. Possible reasons are:
    - *Configuration conflict* – This situation occurs if different VR's with the same interface have the crossovering IP-addresses lists.
    - *IP Address list is empty* – no IP-addresses are specified.
    - *Interface has no primary IP address* – interface does not have primary IP-address or it has been deleted.
    - *Interface is down* –interface that VR is built on is in the down state.

V.Other commands

1. Ctl command (external devices management)

The command to manage the external devise.

Команды управления внешними устройствами

Syntax:

ctl

ctl heater

ctl switch

ctl temperature

ctl signaling

ctl trap

Description:

The router has a sensor for the inside temperature and two ports for external devices connection: executive devices connection and signaling sensors connection.

Signalling port supports sensors (relays) with shorted and not shorted contacts.

Executive devices port can work in two modes: with external and internal power source. The voltage of the internal power supply source is 5V, maximal load current is 200 mA. When connecting with an external power supply source, its voltage should not exceed 30V and maximal load current should not exceed 1 A.

ctl command executed with no additional parameters shows current temperature inside the router in Celsius.

ctl heater command sets a threshold for the temperature under which the heater is turned on. The range of threshold temperatures is from -15 C to 0 C. This command is available only for "**OT**" devices. For example:

ctl heater -10

ctl switch command manages the executive devices port and lets switching on/off the executive device manually or automatically.

ctl switch on | off

- **on** – turn the executive device on
- **off** – turn the executive device off

ctl switch auto signalling | temperature

- **temperature** – manages the executive device automatically according to the temperature sensor data
- **signalling** – manages the executive device automatically according to the external sensor data

Ctl temperature command sets the range of temperatures and the mode for the automatical external device management.

***ctl temp**erature **hi**gh значение **lo**w значение **co**oler | **he**ater*

- **high** – sets the upper range for the temperature reaching which it is required to turn on/off the executive device

- **low** – sets the lower range for the temperature reaching which it is required to turn on/off the executive device
- **cooler** – the executive device is a cooler, and the work is carried out according to the following scheme:
    - if the temperature is higher or equal with **high**, the executive device turns on
    - if the temperature is less than **low**, the executive device turns off
- **heater** parameter means that the executive device is a heater, and the work is carried out according to the following scheme:
    - if the temperature is less than **low**, the executive device is turned on
    - if the temperature is higher or equal with **high**, the executive device is turned off

Ctl signaling command sets the work mode and type of an external sensor.

ctl signalling *[**en**able | **di**sable | **ma**niac] [**op**ened | **cl**osed] [**lo**g | **nol**og] [**bt** value] [**fp** | **nof**p] [**t**raps quantity]*

- **enable** – turn on the sensor data reading
- **disable –** turn off the sensor data reading
- **maniac** – turn the signalization mode on. In this mode, if sensor sends a signal, the events are sent until the **disable** command
- **opended –** not shorted contacts
- **closed –** shorted contacts
- **log, nolog –** turn on/off logging of the events for the sensor
- **bt –** sensor contact bouncing time in milliseconds (according to the passport)
- **fp, nofp –** turn on/off the first event from the sensor. In **fp** mode when the first event occurs it is being sent and then the algorithm against bouncing works; if the sensor event was false, the event is sent that the sensor came back to the original state. **fp** mode cannot be used together with **maniac** mode. If **nofp** mode is on, the anti-bouncing algorithm works first, and then the event is sent.
- **traps** – sets the number of events sent via SNMP protocol when the sensor event occurs. If **traps** value equals 0, no packets are sent. The packets are sent every second. The sensor event in SNMP format has oid 1.3.6.1.4.1.3942.0.100. The event of sensor's return to the original state has oid 1.3.6.1.4.1.3942.0.101. In order to send event in SNMP format, one needs to run the **trapd** service using "**trapd on**" command and configure the required addresses.