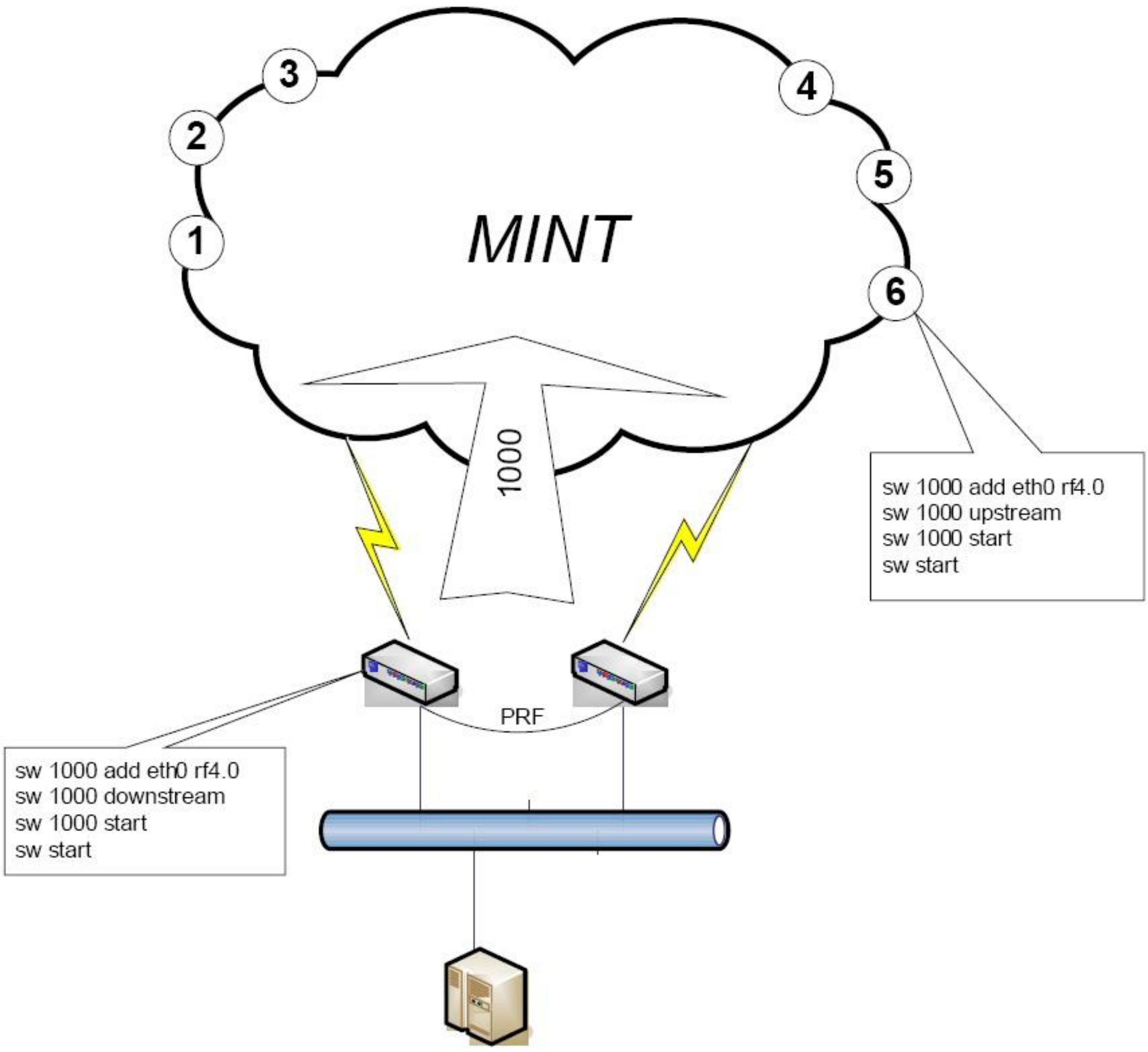
Hypercable
Telecommunications & Broadcast

OS WANFLEX Wi 300 MIMO
Gamme InfiNet Wireless R5000
Manuel technique des configurations utilisateur



```
sw 1000 add eth0 rf4.0
sw 1000 upstream
sw 1000 start
sw start
```

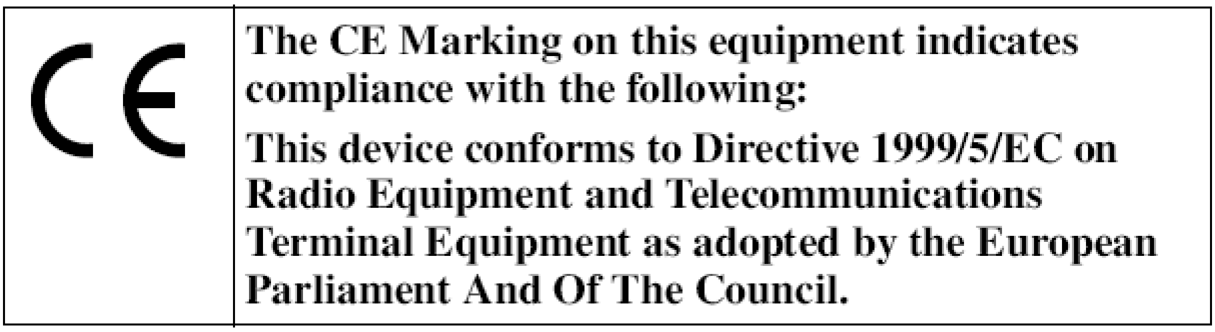
```
sw 1000 add eth0 rf4.0
sw 1000 downstream
sw 1000 start
sw start
```

Liste des commandes du WANFLEX "Operating System" Wi300 & InfiNet Wireless R5000.

Declaration de Conformité

Hereby, InfiNet Wireless declares that this equipment is in compliance with the essential requirements and other relevant provisions of Directive 1999/5/EC.

The declaration of conformity may be consulted at:
http://www.hypercable.fr/images/stories/wirake/DoC_RTTE.pdf



The CE Marking on this equipment indicates compliance with the following:

This device conforms to Directive 1999/5/EC on Radio Equipment and Telecommunications Terminal Equipment as adopted by the European Parliament And Of The Council.

- Ces équipements sont conformes aux prescriptions ARCEP et ANFR selon les arrêtés et publications au Journal officiel :

- Version pour usage en France - *Wi 12/300 R 5000 sont conformes aux recommandations ERC/REC/70-03 du Comité des communications électroniques (ECC) et aux décisions ARCEP qui en tiennent compte. Toute utilisation est soumise au respect des conditions techniques issues des décisions ARCEP n° 2008-568, n° 2005-1080 et n° 2005-1081. Décisions et publications au JO de la République.*

- L'interface Homme Machine (IHM) des Wi 300 et des R 5000 concernant la configuration et l'utilisation est rédigée en Français et en conformité avec la loi Toubon.

- Déclaration de conformité:

  http://www.hypercable.fr/images/stories/wirake/DoC_RTTE.pdf

TABLE DES MATIERES

WANFleX OS User Manual

1 Début du Document en page 4 bis du PDF associé

1.1 Recommandations spécifiques à la transmission Video Surveillance

1.1.1 Début des configurations Video en page 83

1.1.2 Fin des recommandations Vidéo en page 119

COMPREHENSION DE LA GAMME ET DES MANUELS ASSOCIÉS

This document summarizes some technical data and contains detailed descriptions of several key features to give our customers deeper understanding of our products.

Product Line description

InfiNet Wireless produces 2 product lines of devices – **InfiMAN** and **InfiLINK**. **InfiMAN** is a family of Point-to-Multipoint solutions, whereas **InfiLINK** devices are used in Point-to-Point scenarios. Both **InfiMAN** and **InfiLINK** product lines feature MIMO and non-MIMO(SISO) devices. MIMO devices have additional 2x2 in their name, i.e. **InfiMAN 2x2** or **InfiLINK 2x2** (see additional information on MIMO and SISO devices below).

InfiNet Wireless uses several hardware platforms in both product families. The main difference between different hardware platforms is the CPU used in the devices, so the devices mostly differ in performance and price. Different hardware platforms bear simple indexes – H01, H02.. up to H08. Currently InfiNet Wireless uses platforms with indexes H05, H07 and H08(see comparison table below).

№	Platform Index	CPU frequency	Peak performance in kpps	Wired interfaces
1	H08	1GHz	200 kpps	1xGE
2	H07	266 MHz	20 kpps	1xFE
3	H05	533 MHz	50 kpps	1xFE

Different InfiNet Wireless devices can be used in different roles in the network. The role determines the functionality of the device(including the hardware platform and performance, as a consequence). The role of the device can be distinguished from the name of the device:

InfiMAN R5000-Mmx/57.300.2x63.2x22|40

InfiMAN or **InfiLINK**, as we already know, determines the product family. The **R5000-Mmx** part determines the role of the device.

№	Role	Platform	Technology	Family	Product codes
1	High capacity point-to-multipoint base station	H08	MIMO	InfiMAN 2x2	R5000-Omx, R5000-Mmx
2	High capacity point-to-multipoint subscriber terminal	H07	MIMO	InfiMAN 2x2	R5000-Smc, R5000-Lmc
3	PRO| High capacity point-to-point	H08	MIMO	InfiLINK 2x2	R5000-Omx, R5000-Mmx
4	LIGHT| High capacity point-to-point	H07	MIMO	InfiLINK 2x2	R5000-Sm, R5000-Lm
5	Medium capacity point-to-multipoint base station	H05	SISO	InfiMAN	R5000-M, R5000-O
6	Medium capacity point-to-multipoint subscriber terminal	H07	SISO	InfiMAN	R5000-Sc, R5000-Lc
7	Medium capacity point-to-point	H07	SISO	InfiMAN	R5000-S, R5000-L

The difference between the M and O devices (i.e. devices with product codes Mmx, Omx, M, O) is in the antenna used for the device. **M-based devices have an integrated antenna** and **O-based devices use an external antenna** that has to be ordered separately. The same applies to S and L devices. **S-based devices have an integrated antenna** and **L-based devices use an external antenna** that has to be ordered separately.

The last part, after the slash describes the capabilities of the device: **57.300.2x63.2x22|40**

57 represents the frequency range of the device:

23 – 2300-2400 MHz   24 – 2400-2500 MHz   25 – 2500-2600 MHz

26 – 2600-2700 MHz   34 – 3400-3500 MHz   35 – 3500-3600 MHz

36 – 3600-3700 MHz   49 – 4850-4950 MHz   50 – 4950-5050 MHz

51 – 5050-5150 MHz   52 - 5150-5250 MHz   53 – 5250-5350 MHz

54 – 5350-5450 MHz   55 – 5450-5550 MHz   56 – 5550-5650 MHz

57 – 5650-5750 MHz   58 – 5750-5850 MHz   59 – 5850-5950 MHz

60 - 5950-6050 MHz   61 – 6050-6150 MHz   62 - 6150-6250 MHz

63 – 6250-6350 MHz   64 – 6350-6425 MHz

300 – the maximum transfer rate in Mbps

2x63.2x22 - quantity of parallel transmitters and antennas and the power of the transmitters in mW. Here we have **2 transmitters** that support transmit powers **up to 63 mW** and a **22 dBi antenna with 2 receivers for 2 polarizations**;

|40 – this part describes license speed restrictions of the device, if any. |40 means that the device is **license restricted to 40 Mbps** Ethernet interface performance.

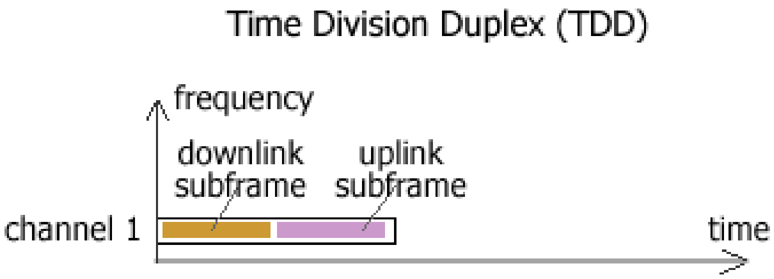
So the description of **InfiMAN R5000-Mmx/57.300.2x63.2x22** product code will be **-** a base station that uses MIMO technology, supports bit-rates up to 300 Mbps, transmit powers up to 63mW and uses 22 dBi dual-pol antenna.

Features

Adaptive Polling (Dynamic TDD)

To prevent over-the-air collisions InfiNet Wireless uses polling(marker access), where one of the devices plays the role of the communications master. This master device consequentially gives the right for transmission and reception to every slave device in the wireless network. Using polling effectively lowers delay and jitter and allows the use of wireless networks for real-time traffic delivery.

InfiNet Wireless devices use a single frequency for transmitting and receiving data over the wireless links. In this case Dynamic Time Division Duplex(Dynamic TDD) multiplexing technology is used to enable full duplex data transmission. Dynamic time slots are allocated for data going in both directions, allowing any ratio for upstream/downstream data flow, depending on the demand in free bandwidth. The size of the allocated time slots depends on the amount of traffic, priority, number of retries and errors on the link, thus allowing InfiNet Wireless effectively serve any kind of data streams – Internet, VoIP, CCTV, etc, as the timeslot structure adapts to the data flow, resulting in the most effective data transmission with the lowest latency and packet loss rate.



Classical TDD systems (Exalt, Orthogon) allow only fixed ratio for incoming/outgoing data flow (50:50, 30:60, etc.). Some systems (e.g. Winlink and most microwaves) allow fixed 50:50 ratio only without the capability to alter ratios. Other TDD systems can be statistically adjusted from the base ratios.

MIMO, MISO and SISO

MIMO (Multiple Input and Multiple Output) - is the use of multiple antennas at both the transmitter and receiver side to improve communication performance. Data is sent on both the horizontal and vertical polarizations (Figure 1). Data is space-time coded (spatial multiplexing) to improve the reliability of data transmission. InfiNet MIMO 2x2 technology effectively doubles spectrum efficiency and allows to achieve real throughput up to 240 Mbps in 40 MHz band.

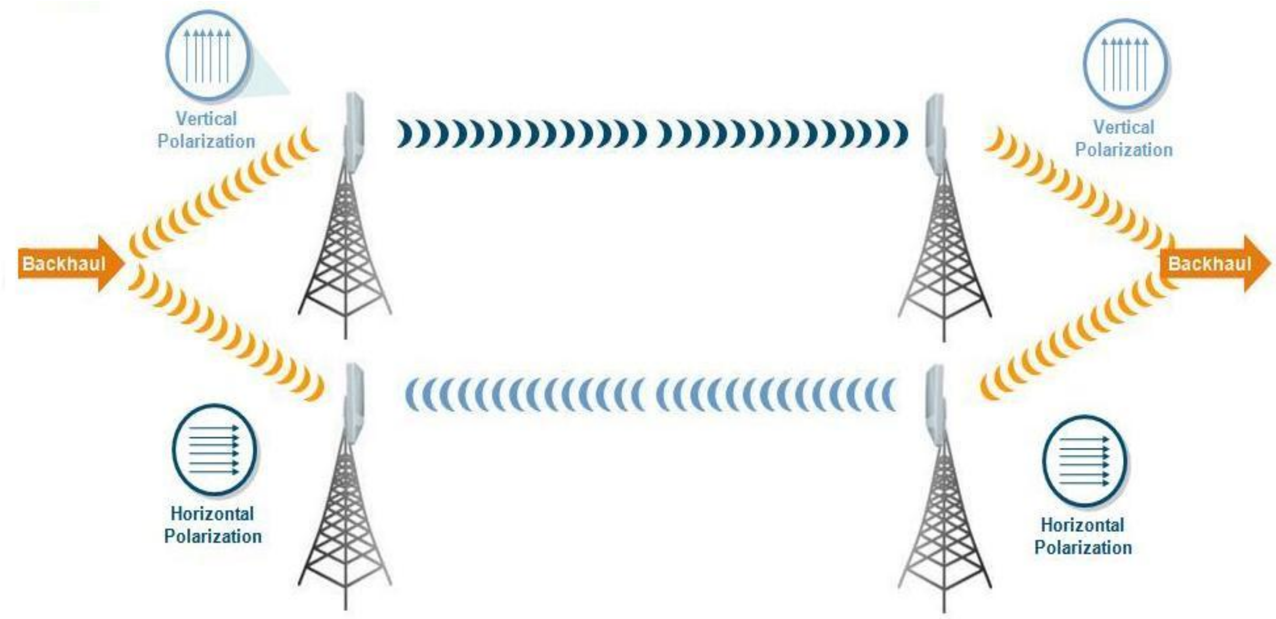


Figure 1. Data transmission path in InfiNet Wireless MIMO products

MISO (Multiple Input Single output) – is a special mode of operation of MIMO devices. It is used in nLOS conditions or in a noisy RF environment. In MISO mode the same data is transmitted over both polarizations, lowering the performance of the link, but enhancing the ability to transmit data in case of interference or obstacles in transmission path (Figure 2).

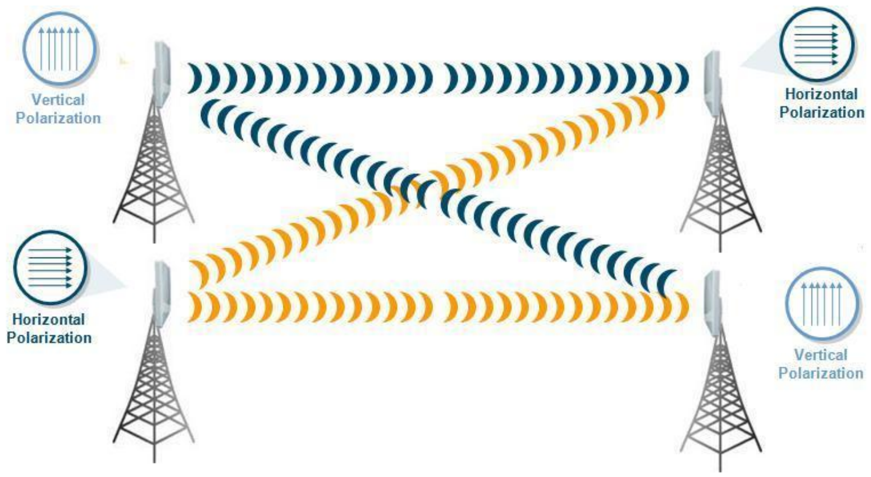


Figure 2. Data transmission path in InfiNet Wireless MIMO products in MISO mode

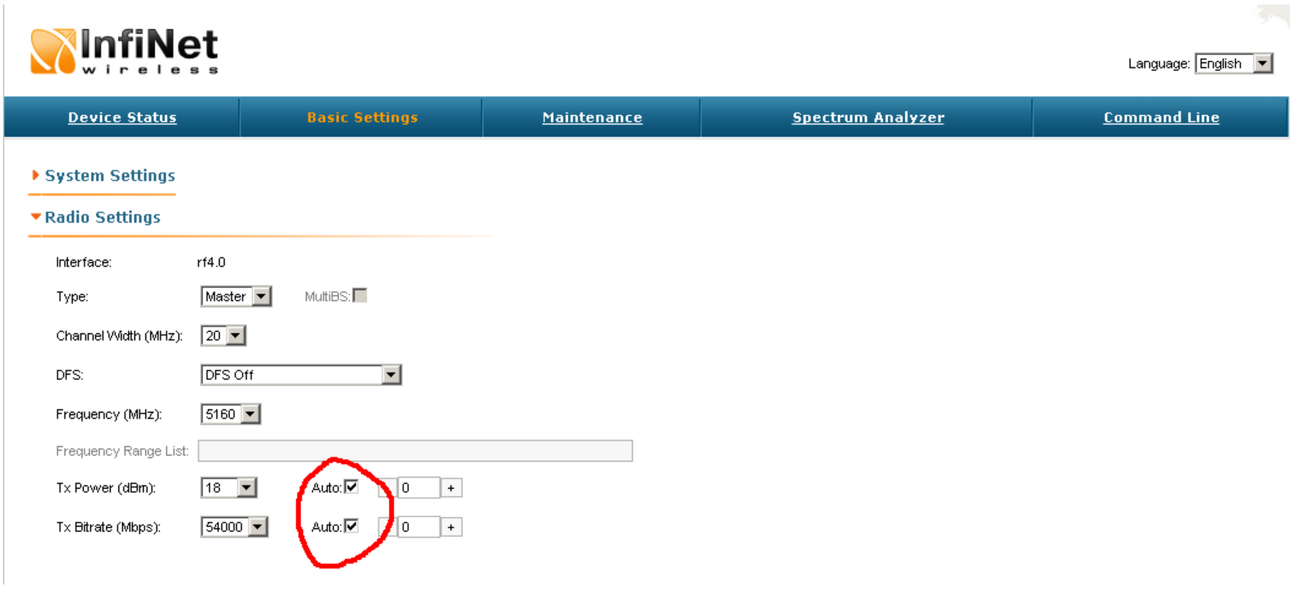
Non-MIMO (or SISO – Single Input Single Output) – is a low performance mode of wireless device operation. Devices use single polarization for data transmission, and as a consequence, allow lower bitrates and are less prone do interference and obstacles in the transmission path. SISO devices are used in LOS conditions, undemanding applications and networks with low bandwidth requirements.

Autobitrate and Automatic power control

Autobitrate Functionality allows the device to maintain optimal rate in given environmental conditions (signal levels, retries, errors on the link). InfiNet rate control algorithms are proactive: they change the rate **before** any data gets lost. InfiNet Wireless algorithms performs active channel testing before switching real traffic to another bitrate, they send series of test packets to ensure a consistent fit for actual data. This is equivalent to AMOD functionality from other vendors

Automatic power control (ATPC) ensures **optimal transmit power** level for the link, maintaining best SNR for the current bitrate and retry level. ATPC works on a per packet basis, allowing individual power setting for every packet transmitted to every CPE or point-to-point wireless node, ensuring best achievable performance and link quality.

Both Autobitrate and ATPC can be toggled using the Web-interface, Basic Settings tab, Radio Settings section(see figure 3):



Language: English

Interface: rf4.0
Type: Master MultiBS: □
Channel Width (MHz): 20
DFS: DFS Off
Frequency (MHz): 5160
Frequency Range List:
Tx Power (dBm): 18 Auto: ☑ 0 +
Tx Bitrate (Mbps): 54000 Auto: ☑ 0 +

Figure 3. Radio settings configuration section

Auto distance learning

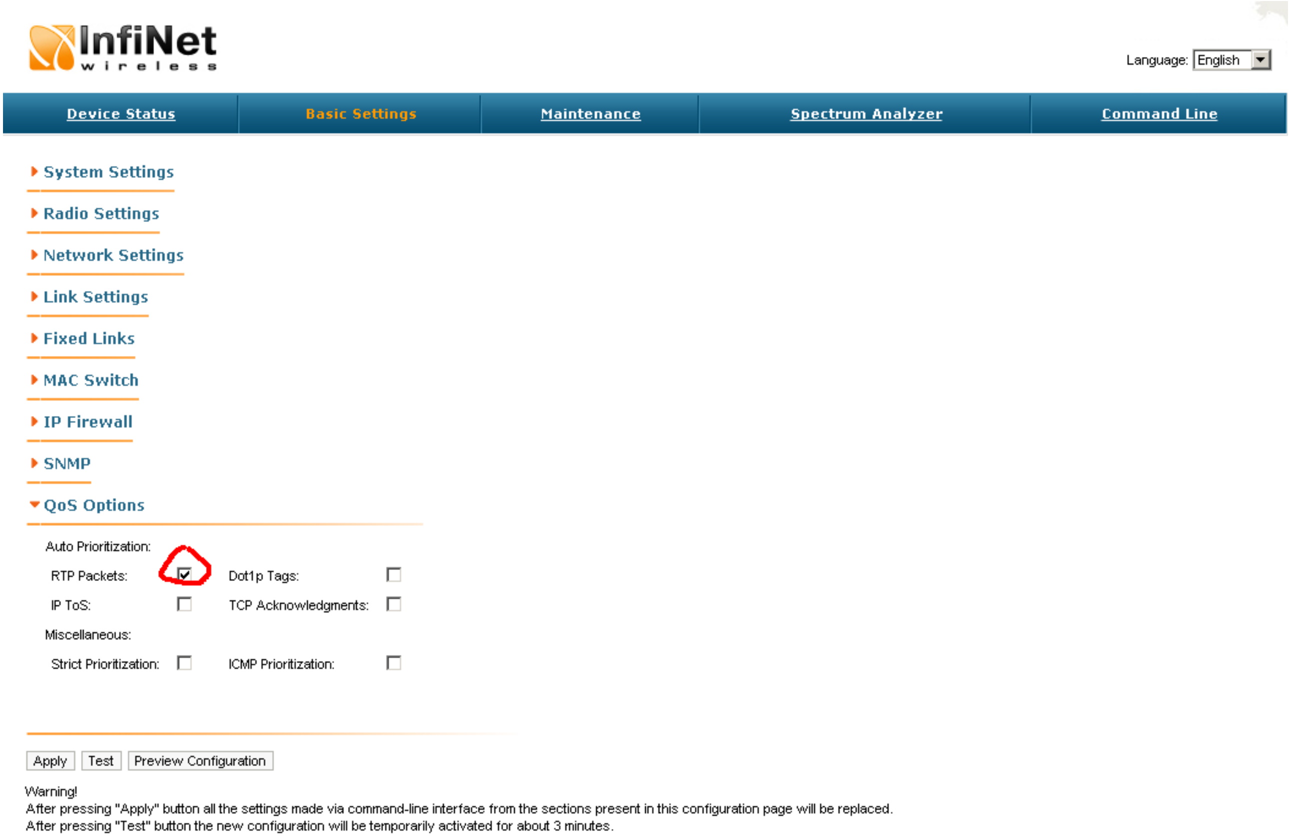
Another feature that ensures maximum achievable link quality is auto distance learning. This feature constantly calculates the distance between the wireless nodes of the link, correcting transmission and reception timers, minimizing delay and optimizing the link. This is enabled by default and can be disabled only using CLI.

Voice/RTP Aware Superpacketing

Every frame transmitted by an InfiNet Device device has a significant amount of overhead, including radio level headers, Media Access Control (MAC) frame fields, interframe spacing, and acknowledgment of transmitted frames. At the highest data rates, this overhead can consume more bandwidth than the payload data frame. To address this issue, InfiNet implements intellectual aggregation algorithm(superpacketing). Superpacketing algorithm groups several data frames into one large frame. Because management information needs to be specified only once per frame, the ratio of payload data to the total volume of data is higher, allowing higher throughput.

Voice aware superpacketing in InfiNet devices allows to automatically disable frame aggregation for RTP traffic, while continuing aggregating all the rest of the traffic. Voice packet aggregation is disabled as aggregation leads to high jitter values.

InfiNet Wireless products allow to control when to enable voice awareness with RTP packets check-box in the QoS configuration section of the Web-interface(see figure 4).



Auto Prioritization:

RTP Packets: ☑ Dot1p Tags: □

IP ToS: □ TCP Acknowledgments: □

Miscellaneous:

Strict Prioritization: □ ICMP Prioritization: □

Warning!
After pressing "Apply" button all the settings made via command-line interface from the sections present in this configuration page will be replaced.
After pressing "Test" button the new configuration will be temporarily activated for about 3 minutes.

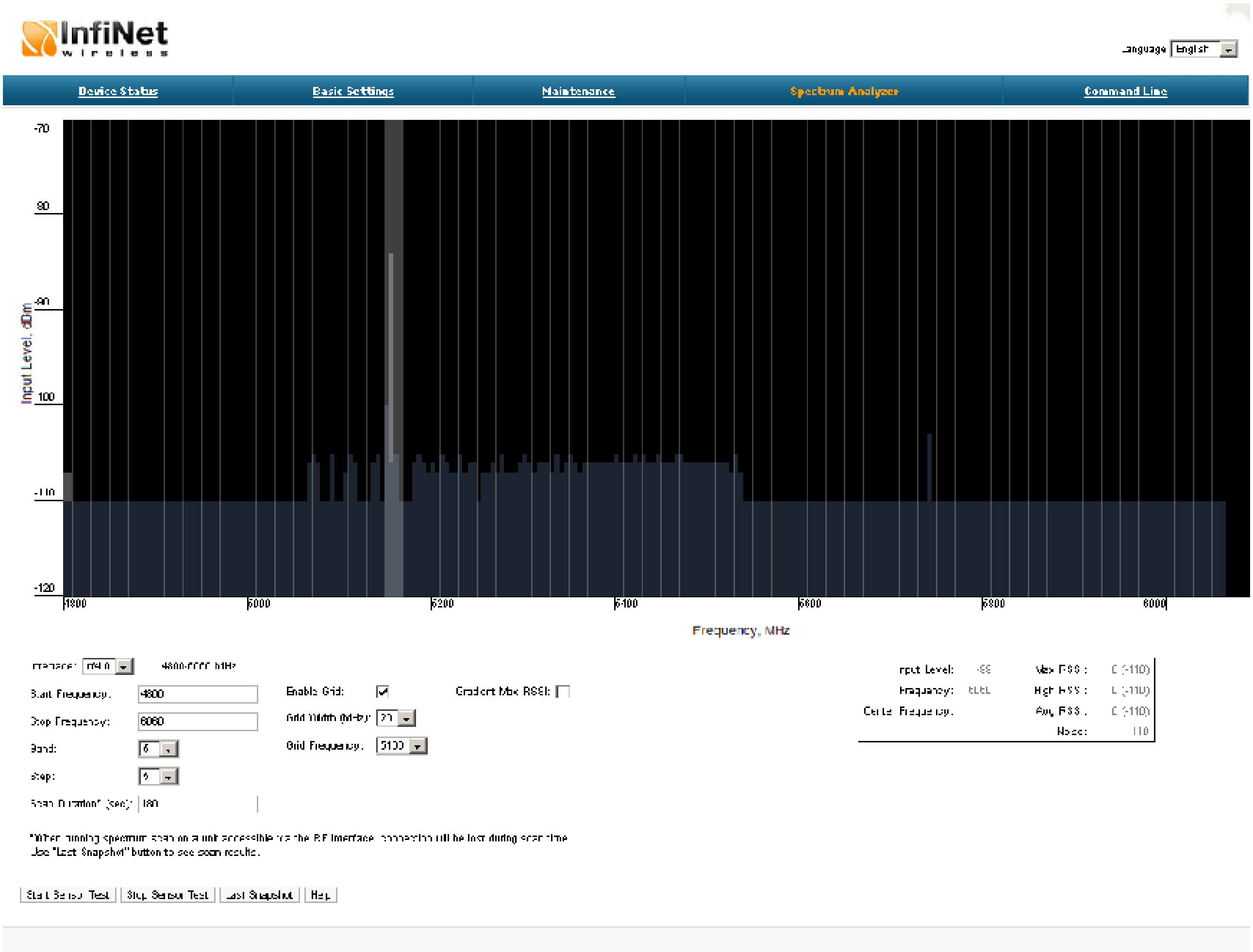
Figure 4. QoS settings configuration section

IGMP snooping

InfiNet Wireless devices fully support IGMP snooping for the purpose of effective multicast traffic transport across the wireless network. In addition, every InfiNet Wireless device can act as IGMP queerer, thus enabling the use of IGMP snooping with multicast traffic sources that don’t support IGMP.

Spectrum analyzer mode

Infinet Wireless devices have the ability to work in spectrum analyzer mode. Spectrum analyzer mode shows noise floor, interference sources and gives the ability to explore the RF medium and choose the best operational frequency for the current link(see figure 5).



Interface: rf4.0 4800-6000 MHz

Start Frequency: 4800

Stop Frequency: 6060

Band: 6

Step: 5

Scan Duration* (sec): 180

Enable Grid: ☑

Grid Width (MHz): 20

Grid Frequency: 5100

Gradient Max RSSI: □

*When running spectrum scan on a unit accessible via the RF interface, connection will be lost during scan time.
Use "Last Snapshot" button to see scan results.

Input Level: -88

Frequency: 6060

Center Frequency:

Max RSSI: 0 (-110)

High RSSI: 0 (-110)

Avg RSSI: 0 (-110)

Noise: 110

Figure 5. Integrated spectrum analyzer

Note! the integrated spectrum analyzer disables radio transmission and reception during operation.

Bridge and switch mode operation

InfiNet Wireless devices allow operation in bridge and switch mode by default. In a point-to-point scenario devices operate in bridge mode, thus transporting all the Ethernet traffic that enters one side of the wireless link to the other side of the link.

Point-to-multipoint scenario allows operation in switch mode, turning you multipoint wireless network into a distributed wireless switch. In this mode traffic entering the wireless network at the base station side or at one of the CPEs will exit the wireless network only through the Ethernet port of the wireless node that belongs to the destination network of this traffic, preventing the flooding of the wireless network and the waste of wireless bandwidth.

Bridge and switch mode are enabled by default on all InfiNet Wireless devices, so after powering them on and some basic configuration you can use you wireless network as a wireless bridge or wireless distributed switch.

Key radio parameters and basic configuration scenarios

All InfiNet Wireless devices are shipped preconfigured for basic operation in the most common scenarios - point-to-point and point-to-multipoint networks.

For a point-to-multipoint scenario, if there is no restriction on the frequency range to use, there is no network configuration needed. Mounting and powering up the base station and the subscribers will lead to link setup on the best frequency from the ones available in the license. The only configuration that you may do – is reconfigure the IP address of the device, as it will have the default address 10.10.10.1.

Note! no configuration is needed if you use devices with the same frequency range ordered in your license.

For a point-to-point scenario, devices are delivered to our customers preconfigured with the default IP address and setup as master devices. So for basic operation the IP address has to be reconfigured and one of the devices should be changed to slave.

Note! no other configuration is needed if you use devices with the same frequency range ordered in your license.

Basic configuration will allow the operation of you wireless network in bridge(point-to-point scenario) or in switch mode(point-to-multipoint scenario).

Quality of Service (QoS)

InfiNet Wireless devices fully support Quality of Service features, allowing the use of wireless networks for real-time and delay sensitive data transport:

- Up to 200 service classes and service channels
- Up to 16 priority levels for granular traffic prioritization
- End-to-end traffic prioritization
- Automatic VoIP (RTP) streams prioritization
- Automatic traffic prioritization based on DSCP, IP ToS or 802.1p QoS tags
- Smart and flexible traffic classification using any attributes including IP ToS/DSCP/802.1p tags, VLAN/IP/MAC address, protocol and port fields, etc.
- Service channels and classes hierarchy for building granular QoS policies
- Policy-based IP ToS/DSCP/802.1p QoS field set/change/strip
- All QoS features are available in both routing and in switching mode
- Traffic shaping (absolute/relative/mixed) with maximum latency limiting (from 20 to 200 ms) on service class/channel basis
- Strict priority queuing and Weighted fair queuing support

Although InfiNet Wireless products allow very sophisticated QoS policies, the easiest way to enable QoS in you wireless network is to enable automatic prioritization. This will give priority to real-time traffic and will transparently prioritize all the rest of the data according to 802.1p or TOS values set. This is easily done in the Web-interface.

For deeper understanding of InfiNet Wireless QoS features please address InfiNet Wireless QoS white paper.

Complete Supported Feature List

RADIO

Voice/RTP Aware Super packeting
- to minimize jitter and latency for multimedia applications

DFS
- for Radar Detection and intelligent search for cleanest channel within a regulatory domain

Automatic Bitrate Control
- to ensure a 100% stable link irrelevant of changes in external conditions

Automatic Transmit Power Control
- to track and keep optimal input signal level to maximize performance for each link and reduce overall interference within a given transmit power and EIRP limitations

Automatic Distance Learning
- to optimize performance for any link distances from tens of meters to 100 km and above

Channel Time Adjustment
- to improve performance on heavily loaded links

Spectrum Analyzer mode
- for interference detection and avoidance

Channel testing tools
- channel performance measurement
- advanced diagnostics

MANAGEMENT FEATURES

Web-interface
- basic settings
- channel diagnostics: spectrum analysis, antenna alignment, channel throughput measurement
- unit and RF links monitoring
- maintenance: firmware upgrade, license and configuration import/export
- tech support diagnostic reports generation
- command-line access

Command-line interface for indepth configuration and diagnostics accessible via:
- secure shell (SSH)
- telnet
- serial port
- remote shell

SNMPv1 / SNMPv3 support
(MIB II, private MIB)

Configurable SNMP Traps

SECURITY FEATURES

Line-speed AES128 over-the-air encryption
Storm / flood protection
Password protection
Protocol messages encryption
Secure command-line access via SSH protocol

NETWORKING

Ethernet-over-IP tunneling
ARP protocol support
MAC/IP filtering
Fully-fledged Layer 2 switch:
- Transparent transport for any type of Ethernet traffic including MPLS, stacked VLANs, etc.
- Multiple switching groups
- Full VLAN support including Q-in-Q (IEEE 802.1q and 802.1ad)
- STP/rSTP support
- IGMP Snooping with Querrier mode
- Trunk groups support

RIPv2 / OSPFv2 /static routing
Tunneling
(Ethernet-over-IP, IP-over-IP)
L2/L3 Firewall
NAT(multipool, H.323-aware)
DHCP client/server/relay

QUALITY-OF-SERVICE

With many QoS permutations, QoS implementation works transparently in the network based on IEEE802.1p standard as well as ToS/DiffServ, guaranteeing perfect performance under any load and lowest jitter/delays for priority traffic.

Quality-of-Service features:
16 priority queues
IEEE 802.1p support
IP TOS / DiffServ support
Full voice support
Traffic limiting (absolute, relative, mixed)
Traffic redirection

MAC

Dynamic adaptive Polling
- Centralized marker grant mode
- Dynamically takes into account channel activity
- Permanent channel testing

Pseudo-radio Interface
- unique InfiNet Wireless feature to join InfiNet Wireless networks via 3rd party equipment (Wired Ethernet segments, IP clouds)

Automatic over-the-air firmware upgrade

STANDARD COMPLIANCE

Radio
- EN 301 893 v.1.5.1
- EN 302 502 v.1.2.1

EMC
- EN 301 489-1
- EN 301 489-17

Safety
- EN 60 950-1:2006

RoHS
- Directive 2002/95/EC

ENVIRONMENTAL

Outdoor Units:
-40..+60C, 100% humidity, condensing

Indoor Unit:
0..+40C, 95% humidity, non-condensing

InfiNet Wireless R5000

WANFleX OS User Manual

Updated**: 15 June 2011**

OS Version**: V1.83**

Legal Rights

The information contained in this document is originated by, proprietary, confidential and owned by InfiNet Wireless. No part of this document should be disclosed, reproduced or distributed without the express written permission of InfiNet Wireless Ltd.

InfiNet Wireless Ltd. reserves the right to change the information contained in this document without prior notice. No part of this document may be considered as a part of any contract or warranty.

Statement of Conditions

The information contained in this manual is subject to change without notice.

InfiNet Wireless Ltd. shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this manual or equipment supplied with it.

Disclaimer

The software is sold on an "AS IS" basis. InfiNet Wireless, its affiliates or its licensors make no warranties, whatsoever, whether express or implied, with respect to the software and the accompanying documentation. Infinet Wireless specifically disclaims all implied warranties of merchantability and fitness for a particular purpose and non-infringement with respect to the software. Units of product (including all the software) delivered to purchaser hereunder are not fault_ tolerant and are not designed, manufactured or intended for use or resale in applications where the failure, malfunction or inaccuracy of products carries a risk of death or bodily injury or severe physical or environmental damage ("high risk activities"). High risk activities may include, but are not limited to, use as part of on-line control systems in hazardous environments requiring fail-safe performance, such as in the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, life support machines, weapons systems or other applications representing a similar degree of potential hazard. Infinet wireless specifically disclaims any express or implied warranty of fitness for high risk activities.

InfiNet Wireless hereby declares that R5000-Om, R5000-Mm, R5000-Sm and R5000-Lm are in compliance with the essential requirements and other relevant provisions of Directive 1995/5/EC. The declaration of conformity may be consulted at ***http://www.infinetwireless.com/products-technologies/type-approval-certificates/DoC_RTTE.pdf****.*

Indication of the countries

InfiNet Wireless equipment has no geographical limitations for selling and can be supplied to any country of the world.

Limitation of Liability

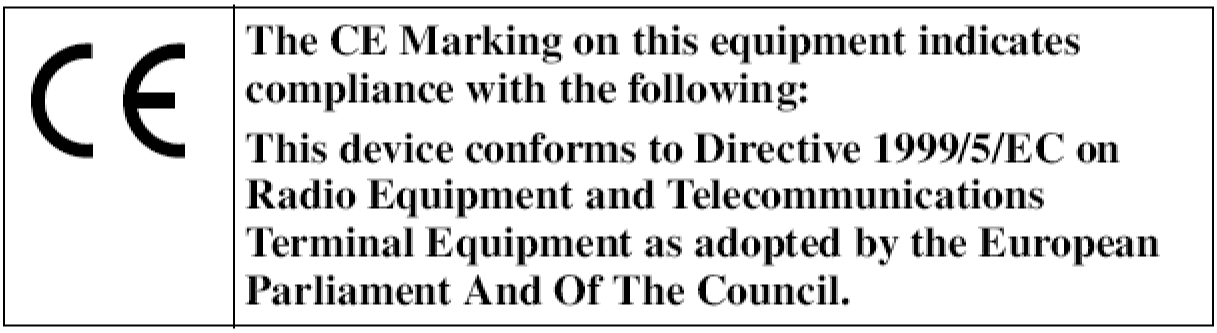
Infinet Wireless shall not be liable to the purchaser or to any third party, for any loss of profits, loss of use, interruption of business or for any indirect, special, incidental, punitive or consequential damages of any kind, whether arising under breach of contract, tort (including negligence), strict liability or otherwise and whether based on this agreement or otherwise, even if advised of the possibility of such damages.

To the extent permitted by applicable law, in no event shall the liability for damages hereunder of Infinet Wireless or its employees or agents exceed the purchase price paid for the product by purchaser, nor shall the aggregate liability for damages to all parties regarding any product exceed the purchase price paid for that product by that party (except in the case of a breach of a party's confidentiality obligations).

International Regulatory Information

This equipment has been tested and found to comply with the limits for a Class B digital device.

Hereby, InfiNet Wireless declares that this equipment is in compliance with the essential requirements and other relevant provisions of Directive 1999/5/EC.



The CE Marking on this equipment indicates compliance with the following:

This device conforms to Directive 1999/5/EC on Radio Equipment and Telecommunications Terminal Equipment as adopted by the European Parliament And Of The Council.

Table of contents

I. Introduction

1. General notes

This manual lists the commands of the WANFleX operating system used in the InfiNet Wireless R5000.

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:

This command displays the list of all the device's commands. It is 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 new user 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 describes the system location; used in SNMP protocol.

  Example:

  ```
  sys location On the Carlson'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 automatic radio module 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 a maximum 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 OfficialAddress IP-address**

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

- **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 out coming 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 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. Do not support automatic summer/winter time switching.

Syntax:

```
set TZ TIMEZONE
```

Example:

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

For more details on time zones please visit: http://en.wikipedia.org/wiki/Time_zone

4. Config command (configuration manipulations)

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

Syntax:

```
config [show | save | clear]
config import | export 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 route rip
```

will display parameter configuration for route command 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).

- **clear**

  Clears (resets to default) configuration in device flash. To take effect device should be rebooted without saving the configuration.

- **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|put login[:password]@host/file [-S src_addr]
```

Description:

With "**get**" option loads a new software version into the device from a remote server using FTP. With "**put**" option downloads current software from the device.

File 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. **Host** is the IP_address of the remote FTP server.

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.

Example:

```
flashnet get ftp:ftp@192.168.1.1/R5000-H05S01-MINTv168.7.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 repeated sign ".".

6. Restart command

The command performs soft router reset.

Syntax:

```
restart [y]
restart SECONDS
restart stop
```

Description:

Full reset and re-initialization 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 re-initialization (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 lose control over the device. The system will periodically inform the user about the time left to re-initialization 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 re-initialization.

7. Ping command

The command sends test packets.

Syntax:

```
ping IP [size|-s size_in_bytes] [count|-c count_packets] [source|-S IP]
```

Description:

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

- **IP** - the IP-address of the tested host;
- **Size/-s** - the test packet length within the range of 10 to 8000 bytes (optional, 64 by default);
- **Count/-c** - the number of the test packets (optional, 5 by default).
- **Source/-S** – sets different source address. 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.

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. Webcfg (Web interface support)

Web-interface support module.

Syntax:

webcfg start|stop

Description:

This command enables/disables Web-interface support on the device. Web-interface allows easy graphical device configuration with the help of a Web-browser.

Example:

webcfg start

11.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
```

12. SSH protocol support

SSH (Secure Shell) protocol allows secure remote management of network devices. Its functionality is similar to Telnet protocol but, as opposed to Telnet, SSH encrypts all the traffic including transmitted passwords. Moreover, SSH is a flexible protocol that allows requiring multiple forms of authentication for access and choosing different encryption algorithms.

For using SSH protocol SSH Server and SSH Client is needed. SSH Server accepts connections from client hosts (SSH Clients), performs their authentification and start serving the authorized clients. SSH Client is used for access and management of the remote device.

InfiNet Wireless devices has built-in SSH Server and SSH Client functionality.

SSH Server ("sshd" command)

Built-in SSH Server (SSH daemon) configuration is performed using "sshd" command.

Syntax:

sshd -window=SIZE

sshd -keepalive=TIME

sshd -banner=on | off

sshd -log-level={emerg | alert | crit | error | warning | notice | info | debug | LEVEL}

sshd -algo-list

sshd -kex-algos=ALGO-LIST

sshd -hostkey-algos=ALGO-LIST

sshd -cipher-algos=ALGO-LIST

sshd -hash-algos=ALGO-LIST

sshd -comp-algos=ALGO-LIST

sshd -auth-methods[=AUTH-METHODS-LIST]

sshd start

sshd stop

sshd newkeys

sshd pub[key] {sh[ow] | cl[ear] | de[lete] N}

sshd pub[key] {in[stall] | im[port] [LOGIN[:PASSWORD]@]HOST/FILE} [COMMENT]

Description:

By default, the SSH Server is disabled. To start/stop the server "sshd start"/ "sshd stop" commands are executed.

When first-time started SSH Server generates DSS and RSA Host Keys to be used for public key based authentication of the SSH Server. Compulsory Host Keys re-generation is possible at any time by typing "sshd newkeys" command.

When started, SSH Server accepts authorization requests from SSH Clients.

Access to the device via SSH protocol may be limited by using "$ACLOCAL" access control list. When "$ACLOCAL" list is configured on the device SSH Server rejects all connection attempts from SSH Clients with IP-address or networks that are not present in the list. More information about "$ACLOCAL" list can be achieved in "Access Control Lists" section of this manual.

The following SSH Server options are available for management:

- **-window=SIZE** - allows changing SSH Server internal receiving window size in bytes. SSH Server window size defines maximum allowed bandwidth for "SSH Client - SSH Server" data channel. By default, SSH Server window size is 24576 bytes.
- **-log-level={emerg | alert | crit | error | warning | notice | info | debug | LEVEL}** – allows choosing logging levels of the SSH

Server service information that will be written into the system log (to manage system log please use “sys log” command). Different levels of logging can be chosen by “emerg”, “alert”, “crit”, “error”, “warning”, “notice”, “info”, “debug” parameters or specified by the number of the needed level (from 0 to 7) using numeric “LEVEL” parameter. By default, “info” (6th level) is chosen.

- **-keepalive=TIME** - sets session activity check duration period in seconds. By default server doesn’t make activity check (“0” value).
- **-banner=on/off** – shows/hide IW WANFleX SSH login banner.

“Options” for SSH algorithms management:

- **-algo-list** – shows a list of all available SSH algorithms for key exchange (kex), authentification (host key), data encryption (cipher), data verification (hash) and data compression (compress).
- **-kex-algos=ALGO-LIST** - allows choosing kex algorithms from the list of algorithms (-algo-list) to be used in SSH key exchange process.
- **-hostkey-algos=ALGO-LIST** - allows choosing host key algorithms from the list of algorithms (-algo-list) to be used in SSH Server-Client authentification process.
- **-cipher-algos=ALGO-LIST** - allows choosing cicpher algorithms from the list of algorithms (-algo-list) to be used in SSH data encryption.
- **-hash-algos= ALGO-LIST** - allows choosing hash algorithms from the list of algorithms (-algo-list) to be used in SSH data verification.
- **-comp-algos=ALGO-LIST** - allows choosing compression algorithms from the list of algorithms (-algo-list) to be used in SSH data compression.

Authentication of SSH Clients

By default, SSH Clients are authorized by SSH Server after typing a valid password. However, password authentication may be not enough to provide the needed security level. InfiNet Wireless devices have several built-in SSH authentication methods that are managed by “sshd pubkey” and “sshd -auth-methods” commands.

“Sshd pubkey” command allows enabling public key based authentification of SSH Clients. In the Public key authentication mode SSH Server authorize SSH Client bypassing password login procedure. This mode is enabled automatically once a public key of the SSH Client is cached in SSH Server’s registry.

To upload SSH Client’s public key into the SSH Server registry from a remote FTP server use the following command:

```
sshd pubkey import [LOGIN[:PASSWORD]@]HOST/FILE [COMMENT]
```

Where “HOST” – is an IP-address of the remote FTP server and “FILE” is a file containing SSH Client’s RSA/DSS public key in OpenSSH or SSH2 format. If login and password are set on the remote FTP server they should be specified as “LOGIN” and “PASSWORD” parameters in the command syntax. “COMMENT” parameter allows adding a comment to the public key entry in the SSH Server list of clients public keys. By default, a comment with clients IP-address or FTP IP-address from where the key was obtained is added.

The other way to cash SSH Client’s public key in the SSH Server registry is to perform a password authentification of the SSH Client and then type the following command on the SSH Server:

```
sshd pubkey install [COMMENT]
```

By doing this SSH Server will save a public key of the connected SSH Client. Usage of the "COMMENT" parameter here is the same as it is in the "sshd import" command described above.

"Sshd show" command shows SSH Client's public keys that are registered in the SSH Server list.

"Sshd clear" command deletes all the SSH Client's public keys from the SSH Server.

"Sshd delete N" command deletes a certain SSH Client's public key from the SSH Server list. Parameter "N" – is an index of the key in the list.

"Sshd -auth-methods" command allows viewing available authentification methods and choosing them for using in authentification process.

To view all available authentification methods please type:

```
sshd -auth-methods
```

To choose which of them to use:

```
sshd -auth-methods=AUTH-METHODS-LIST
```

Where "AUTH-METHODS-LIST" is a list of authentification methods allowed for using. To enable all authentification methods you can use "all" value for "AUTH-METHODS-LIST" parameter (set by default).

Example:

The following command prohibits using password authentification allowing only "Public key" method.

```
sshd -auth-methods=publickey
```

SSH Client ("sshc" command)

Built-in SSH Client configuration is performed using "sshc" command.

Syntax:

```
sshc [options] [LOGIN@]HOST [REMOTE-COMMAND]
options:
  -window=SIZE
  -keepalive=TIME
  -compress, -C
  -bind-addr=ADDR, -b ADDR
  -pubkey-show
  -pubkey-new=BITS
  -pubkey-clear
  -pubkey-export=[LOGIN[:PASSWORD]@]HOST/FILE
  -algo-list
  -kex-algos=ALGO-LIST
  -hostkey-algos=ALGO-LIST
```

```
-cipher-algos=ALGO-LIST, -c ALGO-LIST
-hash-algos=ALGO-LIST, -m ALGO-LIST
-comp-algos=ALGO-LIST
```

Description:

To connect to the remote device (with SSH Server running on it) using SSH Client type the following command:

```
sshc [options] [LOGIN@]HOST [REMOTE-COMMAND]
```

Where “LOGIN” is a login user name for remote device (maybe omitted when default logging name is used on the remote device) and “HOST” is IP-address of a remote device. Optional “REMOTE-COMAND” parameters defines a command that should be executed on the SSH Server after successful login.

The following SSH Client «options» are available:

- **-window=SIZE** - allows changing SSH Client internal receiving window size in bytes. SSH Client window size defines maximum allowed bandwidth for “SSH Server - SSH Client” data channel. By default, SSH Client window size is 24576 bytes.
- **-keepalive=TIME** - sets a frequency of sending compulsory session activity confirmations to the server. This allows not to loose the session to the server when SSH Client leaved unused for a long time period. By default, SSH Client doesn’t send any special activity confirmations (“0” value). Measured in seconds.
- **-compress, -C** – enables data compression.
- -**bin-addr=ADDR** (short form: **-b ADDR**) – sets source IP-address of the SSH packets. This source IP-address substitutes the default sending interface's IP-address field of the SSH packets.

“Options” for Public key management:

- **-pubkey-new=BITS** – generates new DSS and RSA SSH Client’s public keys. “BITS” parameter should be specified as a key size in bits (possible values: 512-4096). For example, the following command generates RSA and DSS 512 bits public keys:

  ```
  sshc –pubkey-new=512
  ```

- **-pubkey-show** – shows generated public keys
- **-pubkey-export=[LOGIN[:PASSWORD]@]HOST/FILE** – exports public keys from SSH Client to a file on the remote FTP server. Where “HOST” – is an IP-address of the remote FTP server and “FILE” is a file name that will contain SSH Client’s RSA/DSS public keys. If login and password are set on the remote FTP server they should be specified as “LOGIN” and “PASSWORD” parameters.
- **-pubkey-clear** – deletes public keys on SSH Client.

“Options” for SSH algorithms management:

- **-algo-list** – shows a list of all available SSH algorithms for key exchange (kex), authentification (host key), data encryption (cipher), data verification (hash) and data compression (compress).
- **-kex-algos=ALGO-LIST** - allows choosing kex algorithms from the list of algorithms (-algo-list) to be used in SSH key exchange process.

- **-hostkey-algos=ALGO-LIST** - allows choosing hostkey algorithms from the list of algorithms (-algo-list) to be used in SSH Server-Client authentification process.
- **-cipher-algos=ALGO-LIST** (short form: **-c ALGO-LIST**) - allows choosing cicpher algorithms from the list of algorithms (-algo-list) to be used in SSH data encryption.
- **-hash-algos=ALGO-LIST** (short form: **-m ALGO-LIST**) - allows choosing hash algorithms from the list of algorithms (-algo-list) to be used in SSH data verification.
- **-comp-algos=ALGO-LIST** - allows choosing compression algorithms from the list of algorithms (-algo-list) to be used in SSH data compression.

For compulsory interruption of the SSH Client's session (for example, if SSH Server is not responding to SSH Client's requests) please use the following key sequence: **<Enter>~.** (on the keyboard, firstly, press «Enter» key, then «~» key, then «.» key)**.**

Example:

```
sshc -C root@1.2.3.4
```

This command will try to connect SSH Client to the SSH Server located on the host with IP-address 1.2.3.4 under "root" login user name and enable compression of the data flow.

13. Ipstat command (IP-statistics)

IP statistics gathering module.

Syntax:

```
ipstat enable [incoming|outgoing|full] [detail] [SLOTS] | disable
ipstat clear
ipstat traf [detail] [bytes | total_bytes]
ipstat fixit | fixget | fixclear
ipstat strict | -strict
ipstat add [intf] rule...
ipstat del N
ipstat rearrange [N]
```

Description:

The IP statistics gathering module provides for collecting information on data flows 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 "SLOTS" numeric parameter of an "ipstat enable SLOTS" 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 enable [incoming|outgoing|full] [detail] [SLOTS] | disable** – enables/disables ip statistics gathering. It can allow gathering only **incoming**/**outgoing** or **full** (both) data flows. **Detail** option switch on detailed ip statistics gathering including ports and protocols information. **SLOTS** option allows setting the maximum number of rows in the ipstat table.
- **ipstat clear -** clear accumulated statistical info

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 the "**ipstat strict**" option has been specified, then at the overflow condition the transit routing is disabled, but the router still responds to any protocol.

To get the gathered statistic info remotely one can use 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.

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 rearrange N** command renumbers all the ipstat rules with the given increment (default step is 1).

The **ipstat traf [detail] [bytes | total_bytes]** allows for visually inspecting statistics collection process in real time. **Detail** option switch on detailed ip statistics gathering including ports and protocols information. **Bytes(/total_bytes)** option sort ipstat output according to the number of transmittered bytes in the moment(/bytes transmitted for the whole period).

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);
}
```

14. Sflowagent (Sflow Agent)

Sflow Agent is a realization of a standard Sflow protocol agent.

Syntax:

```
Available commands are:
  sta[rt]                   Start Sflow agent
  sto[p]                    Stop Sflow agent
  wi[pe]                    Stop Sflow agent and clean all configuration
  add[instance] 'name'          Add instance (default 'ipstat')
  del[instance] 'name'          Delete instance (default 'ipstat')
  stat 'name'                   Show statistics for instance (default 'ipstat')
  cl[earstat] 'name'            Clear statistics for instance (default 'ipstat')
Available options are:
  -collector=IPaddress[:port] Set collector address
  -agent=IPaddress            Set agent address (default 0.0.0.0)
  -maxpacket=size             Set maximal datagram size (default 1500)
  -interval=number            Set statistics recieve interval, in seconds (default 5)
  -datagrams=number             Set datagrams per statistics interval (default 100)
  -rawheader={on|off}           Sends original ipV4 headers (default off)
  -debug={on|off}             Puts debug output to log (default off)
  -version -v                 Display Version
```

Description:

Sflow – protocol for monitoring computer networks. It is commonly used by Internet Providers to capture traffic data in switched or routed networks. *Sflowagent* command allows configuration of Sflow agent on the device.

- **sflow sta[rt]** – starts Sflow agent
- **sflow sto[p]** – stops Sflow agent
- **sflow wi[pe]** – stops Sflow agent and clears its configuration
- **sflow add[instance] 'name'** – adds statistics gathering component (if *'name'* parameter is not specified then '*ipstat*' component will be used)
- **sflow del[instance] 'name'** – deletes statistics gathering component (if *'name'* parameter is not specified then '*ipstat*' component will be used)
- **sflow stat 'name'** – shows statistics for a component (if *'name'* parameter is not specified then '*ipstat*' component will be used)

  Command output:

Parameter	Description
Total flow records	Number or records delivered from *Instance*.
Total flow samples	Number of grouped records delivered from *flow records*.
Overflow records	Number of records in *Instance* for all cases when *Instance* overflowed earlier then *interval* period had ended.
Overflow count	Number of times when *Instance* overflowed earlier then *interval* period had ended.
Total cycles	Overall number of gathering statistics success cycles.
Total datagrams	Overall number of sent datagrams.
Unused datagrams	Number of datagrams that could be created in compliance with *datagrams* parameter but was not used.
Bytes sent	Overall number of transmitted data by Sflow protocol.
Lost flow samples	Number of *flow samples* that were discarded because of *maxpacket*, *interval* and *datagrams* parameters low values.
Lost flow records	Number of *flow records* that were discarded because of *maxpacket*, *interval* and *datagrams* parameters low values.
Lost overflow records	Number of times when *Instance* overflowed earlier then *interval* period had ended and data were lost.

- **sflow cl[earstat] 'name'** – clears statistics for a component (if *'name'* parameter is not specified then '*ipstat*' component will be used)
- **sflow collector=IPaddress[:port]** – sets address of a collector that process sflow-packets. Default port is 6343.
- **sflow -agent=IPaddress –** sets agent's own address (device)
- **sflow -maxpacket=size** – sets maximum size of a Sflow-packet in bytes. 1472 bytes by default. Upper bound is limited by hardware and operational system capabilities. In case of its exceeding packet size will be decreased to acceptable value.
- **sflow -interval=number** – time in seconds equal to interval with which statistics is delivered from *instance*. Increasing of this parameter leads to increasing in overall system efficiency but in case of unexpected network activity splash data could be lost. 15 seconds by default.

- **sflow -datagrams=number** - maximum number of datagrams between times of receiving statistics from instance. The increase of this parameter leads to the decrease in datagram average size and increases in theoretical number of delivered statistics data. Reduces the load on the CPU but in the same time reduces overall system efficiency. However, reducing of system efficiency doesn't happen with low traffic. It is recommended to increase this parameter when decreasing *maxpacket* parameter and/or when increasing *interval parameter*. 100 by default. Maximum flow: *sflow= datagrams/interval* maxpacket, (Bytes/sec)*.
- **sflow -rawheader={on|off}** – sends original ipV4 headers in spite of statistics data (*off* by default). Used for compliance with traffic monitoring programs.
- **Sflow -debug={on|off}** – puts statistics information to log.

Example:

```
ipstat enable full detail 3000 # starting the process of gathering statistics
sflow add ipstat # adding gathering component
sflow -collector=1.2.3.4 start  # starting process of processing the statistics
```

15. Access Control Lists ("acl" command)

Syntax:

```
acl add $NAME TYPE params...
acl del $NAME [params...]
acl ren $NAME1 $NAME2
acl flush
Possible TYPES: net num
Predefined ACL names:
 $ACLOCAL       - Hosts (networks) permitted to configure the device.
```

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 symbols 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
```

Reserved access lists:

$ACLOCAL – reserved list for access limitation to the device via Telnet, FTP, HTTP and SSH protocols. Having "$ACLOCAL" access list in the configuration all attempts to establish a connection with the device from addresses (networks) that are not in this list will be rejected.

Example:

```
acl add $ACLOCAL net 10.0.0.0/8 192.168.0.0/16
```

16. 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 default.

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
```

17.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	Century (is added before Year)
yy	Year in abbreviated 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 200402100530.04
Tue Feb 10 05:30:04 2004
```

```
date
Tue Feb 10 05:30:10 2004
```

18. Erp command (Emergency Repair Procedure)

ERP (Emergency Repair Procedure) utility allows restoring lost system password to the device.

Syntax:

```
erp [options] [command]
[options]:
    -serial <n>     - device serial number
    -code <c>       - ERP code
    -ip <address>   - interface IP address
    -mask <mask>    - interface IP address mask

[command]:
    boot    - force continuing boot on device(s).
              Device serial number may be unspecified
              and means 'any device'.
    reset   - resets device's configuration.
              Serial number and ERP code must be specified
    ifup    - turns up device's interface and add IP address
              and mask alias to it. Serial number, IP address
              and IP address mask must be specidfied
    If command is not specified, then it's assumed the 'boot' command.
```

Description:

Options:

- **-serial <n>** - device serial number
- **-code <c>** - special ERP-code
- **-ip <address>** - IP-address of device’s Ethernet interface
- **-mask <mask>** - net mask of device’s Ethernet interface

Commands:

- **boot** – speeds up device boot
- **reset** – resets device configuration including system user name and password. Serial number and special ERP-code must be specified
- **ifup** – turns up device's Ethernet interface (eth0) and adds IP-address and net mask alias to it. Serial number, IP address and net mask should be specified

How it works

To do Emergency Repair Procedure one needs two InfiNet Wireless devices. First device is a device which should be repaired (device “a”), second – device on which ERP utility will be run to repair the first device (device “b”). Both of the devices should be connected to the same Ethernet segment via their Ethernet interfaces.

Procedure:

1. On the device “b” run ERP utility with “*-serial <n>” option*, where *n* – is a serial number of the device which should be repaired (device “a”).

```
erp –serial <n>
```

   Example:

```
erp –serial <36696>
```

   ERP will come to “hearing” mode waiting for device “a” to reboot.

2. Reboot device which should be repaired (device “a”) by disabling power and enabling it again. For example, by disconnecting and connecting again service cable on the IDU.
3. After device “a” is rebooted, on device “b” ERP will show «Sequence» parameter value and serial number of device “a”. Please contact InfiNet Wireless tech support and provide these values.
4. In return tech support will give you special ERP-code and new password for device “a”.
5. On device “b” run the following ERP command:

```
erp –serial <n> -code <c> reset
```

   where *n* – is a serial number of the device which should be repaired (device “a”) and *c* – is special ERP-code received from tech support

6. Reboot device “a” again
7. ERP utility will reset login, password and configuration on device “a” to default (use any characters as default login and password).
8. Login device “a” with a serial number as a user name and a new password that was received from tech support.

To change IP-address on Ethernet interface of a device “a” from device “” without login to device “a”:

```
erp –serial <SERIAL> -ip <address> -mask <mask> ifup
```

19. AAA (access control using RADIUS server)

AAA module allows access control configuration on the device using remote RADIUS server.

Syntax:

```
aaa [options] [command]
  where commands are:
    start - start service
    stop  - stop service
```

```
 where options are:
   -auth=ip[:port],secret[,identifier] - RADIUS server parameters,
                                          address    - Server IP Address
                                          secret     - shared secret
                                          identifier - NAS Identifier
                                        this option can be repeated.
   -remove=ip[:port]                  - Remove RADIUS server.
   -debug={on|off}]                   - on/off debug output.
```

Description:

AAA start/**stop** commands enables/disables AAA module operation.

Options:

- **-auth=ip[:port],secret[,identifier]** – sets parameters to access remote RADIUS server, where **ip[:port]** – IP-address and port of a RADIUS server, **secret** – password to access the server, **identifier** – NAS (Network Access Server) ID.
- **-remove=ip[:port]** – removes information about a RADIUS server from the configuration of the unit.
- **-debug={on|off}]** – enables/disables AAA module service information output to the system log.

20. License

This command manages operations with a license file on the device.

Syntax:

```
license [options]
 options are:
   --install=<url> - install new license
   --export=<url>  - export current license to external server
   --show          - show license info
   <url> = ftp://[login[:password]@]host/file
```

Description:

Install option uploads license file into the device from a remote server using FTP.

Export option downloads license file from the device to a remote server using FTP.

S**how** option displays license information on the screen.

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** is the IP_address of the remote server.

Examples:

li --export=ftp://ftp_login:ftp_password@192.168.145.1/license_file

li --show

21. Dport

Syntax:

dport BAUD

Description:

This command sets a bitrate of the console port. Available values are: 9600, 19200, 38400, 57600, 115200 Bit/sec. Default value is 38400 Bit/sec.

22. Mem

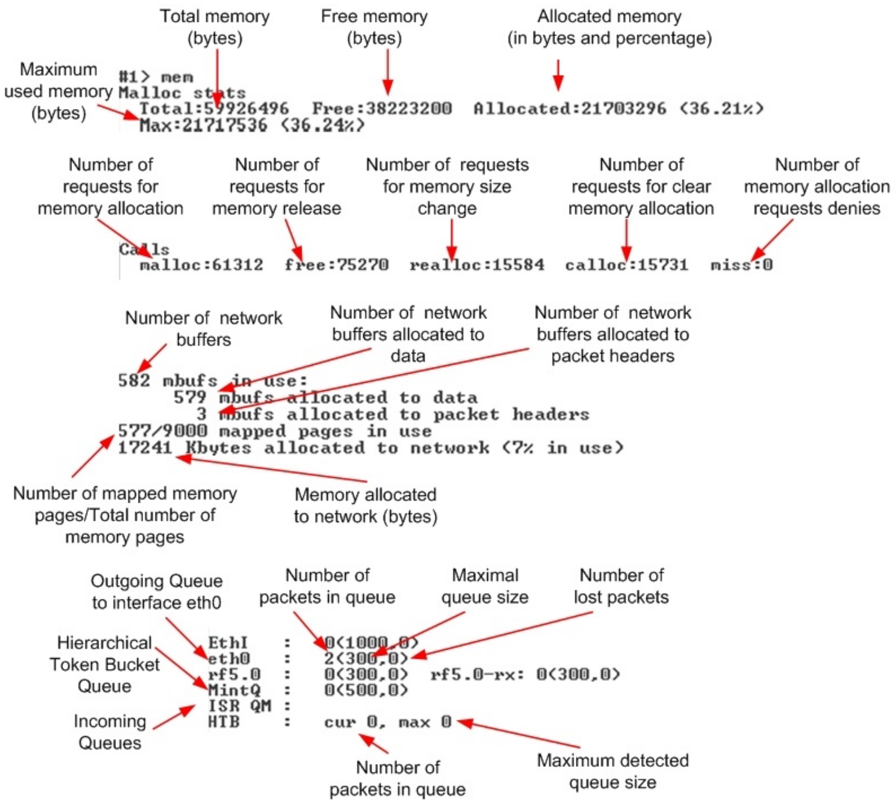
Syntax:

mem

Description:

This command show statistics for allocated device memory, network buffers, queues and drops on interfaces. Command output is described in the picture below.

Sample command output:



```
#1> mem
Malloc stats
  Total:59926496  Free:38223200  Allocated:21703296 (36.21%)
  Max:21717536 (36.24%)

Calls
  malloc:61312  free:75270  realloc:15584  calloc:15731  miss:0

582 mbufs in use:
     579 mbufs allocated to data
       3 mbufs allocated to packet headers
577/9000 mapped pages in use
17241 Kbytes allocated to network (7% in use)

EthI    :   0(1000,0)
eth0    :   2(300,0)
rf5.0   :   0(300,0)  rf5.0-rx: 0(300,0)
MintQ   :   0(500,0)
ISR QM  :
HTB     :   cur 0, max 0
```

Total memory (bytes)
Free memory (bytes)
Allocated memory (in bytes and percentage)
Maximum used memory (bytes)
Number of requests for memory allocation
Number of requests for memory release
Number of requests for memory size change
Number of requests for clear memory allocation
Number of memory allocation requests denies
Number of network buffers
Number of network buffers allocated to data
Number of network buffers allocated to packet headers
Number of mapped memory pages/Total number of memory pages
Memory allocated to network (bytes)
Outgoing Queue to interface eth0
Number of packets in queue
Maximal queue size
Number of lost packets
Hierarchical Token Bucket Queue
Incoming Queues
Number of packets in queue
Maximum detected queue size

23. Grep command

Grep is a console output filtering utility. It filters a command output with lines containing a match to the given regular expression.

⚠ *Attention! This command is not available on the H01/H02 platforms.*

Syntax:

grep [OPTIONS] [-e]PATTERN "command"

or

command | grep [OPTIONS] [-e]PATTERN

OPTIONS:

-e PATTERN, --regexp=PATTERN

-i, --ignore-case

-v, --invert-match

-w, --word-regexp

-x, --line-regexp

-c, --count

-m NUM, --max-count=NUM

-n, --line-number

-A NUM, --after-context=NUM

-B NUM, --before-context=NUM

-C NUM, --context=NUM

Description:

Grep utility searches the output of the given command for lines matching the given PATTERN and displays the result.

When used with "**-e**" option (**-e PATTERN** or **--regexp=PATTERN**) allows searching a PATTERN that begins with dash sign (-).

To ignore case distinction between capital and lowercase letters while searching **–i** (or **--ignore-case**) option is used.

To perform the invert filtering, i.e. find non-matching lines, **-v** (or **-invert-match**) option is used.

To find and display only those lines that matches only the whole world (**-w**) or line (**-x**) the corresponding options are used.

To count the number of the result lines (matching or non-matching if used together with **–v** option) **–c** option is used. Result lines themselves are not displayed.

The **–m** option (**-m NUM**) stops search after the specified number of matching lines.

All the command output lines are index numbered starting from 1. To show the matching lines with their index numbers **–n** option is used.

The **–A NUM**, **-B NUM** and **–C NUM** options are used to print the specified number (NUM) of nearby lines after, before or after and before the matching lines correspondingly. When using these options each matching line in the grep output will be printed together with the specified number of the nearby lines separated from other matching entries with a special line (---).

III. Layer 2 commands set – PHY and MAC

1. Rfconfig command (Radio interface configuration)

The command is used to configure a radio module.

Syntax:

```
rf interface parameters...
Interface rf5.0 parameters:
  band   XXX:  bandwidth (MHz) - {double (40)|full (20)|half (10)|quarter (5)}
  grid   B G:  frequency grid - <bw> freq1[-freq2[/step]],... | clear
  freq   XXX:  central frequency (MHz)
  bitr   XXX:  bitrate (Kbps)
  txpwr XXX:  tx power (dBm)
  sid    XXX:  system identifier - up to 8 hex digits.
  cap       :  RF capabilities
  dist   XXX:  distance in kilometers or auto
  txrt   XXX:  max transmit retries [15]
  txvrt  XXX:  max transmit retries in voice mode [5]
  [-]burst  :  burst mode
  [-]shortgi: short guard interval mode
  noise  XXX:  Noise floor threshold, dB [20]
  [-]pwrctl :  automatic TPC mode
  [-]wocd
  [modulation OFDM | CCK]
  [chntime XXX]
  [statistics]
  [-]bcsid
```

⚠ *Not all of the 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 radio interface) parameter, or displays them if executed without any optional parameter. Optional parameters are as follows:

- **band XXX**: this option allows choosing the bandwidth for transmission (in MHz). **Double** means 40MHz**, Full** - 20MHz, **half** – 10MHz, **quarter** – 5MHz. Units in one PTP or PTM connection must have the same bandwidth.

  Example:

rf rf5.0 band half

- **grid B G**: this option allows creating a customized frequency grid within the license restrictions. The frequency grid for every band is formed in the form of a list of sub ranges with the defined frequency step (start frequency–end frequency/step), or a list of comma delimited standalone frequencies.

Syntax:

```
IFNAME grid BAND FREQUENCY_RANGE_LIST
```

Example:

```
rf rf5.0 grid 40 4920-5940/5
rf rf5.0 grid 20 5310-5390/10,5450,5500-5580/20
rf rf5.0 grid 10 5480, 5500, 5520, 5540, 5560, 5580
```

The resulting frequency grid can be viewed in the “rf cap” command output and is used every time the frequency is setup using the “auto” command (roaming, dfs etc).

To remove a frequency grid:

```
rf IFNAME grid BAND clear
```

That will restore the default frequency grid supplied in the unit’s license.

- **freq XXX**: central operating frequency (in MHz). Must be equal at the both sides of the link. The list of allowed frequencies can be obtained by executing "**rf if-name cap**" command.

- **bitr XXX**: the bit transfer rate (in Kbit/s) of the radio link. Allowed values are:

  - o For CKK mode: 11000, 5500, 2000, 1000 Kbit/s
  - o For OFDM mode (standard models): 6000, 9000, 12000, 18000, 24000, 36000, 48000, 54000 Kbit/s.
  - o For OFDM mode (Xm models):
    - § **5 MHz:** 3250, 6500, 9750, 13000, 19500, 26000, 29250, 32500 Kbit/s
    - § **10 MHz:** 6500, 13000, 19500, 26000, 39000, 52000, 58500, 65000 Kbit/s
    - § **20 MHz:** 13000, 26000, 39000, 52000, 78000, 104000, 117000, 130000 Kbit/s
    - § **40 MHz:** 30000, 60000, 90000, 120000, 180000, 240000, 270000, 300000 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.

- **txpwr XXX**: sets the emitting power of the transmitter (in dBm). The acceptable transmit power values can vary depending on the type of the radio module installed. The acceptable transmit power values can be viewed using the "**rf <if-name> cap**abilities" command.

- **sid XXX**: 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.

- **cap**: displays the capabilities of the radio module including acceptable transmit the information on power levels, frequencies etc.

- **dist XXX**: 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 thus making possible to work on longer distances with smooth adjustment.

There are several ways to manage this parameter:

  - To set an exact value (in kilometers)

  - Use automatic distance calculation (by default). In this case the unit will calculate and set the link distance automatically. In the configuration an automatically calculated value is displayed after the **auto** parameter: **auto (XX)**.

  Auto mode is recommended!

  - When exact device's geographical coordinates are known (e.g. using GPS) you can specify their values in the **sys gpsxy** command and set the distance parameter 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 the distance to the remotest subscriber, and each subscriber will calculate the distance to the Base Station. If the subscriber has the link coordinates information it will use this information, otherwise it will use the **distance** parameter value got from the Base Station.

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

- **txrt XXX**: this parameter sets the maximum number of repeat requests (transmit retries) to be done when sending unicast packets. 15 by default.

- **txvrt XXX**: this parameter sets the maximum number of repeat requests for data packets (excluding voice packets) in voice mode. 5 by default. The maximum allowed value is 64. Voice mode is turned on automatically when VoIP traffic appears.

- **burst**: enables the BURST protocol. “**-burst**” disables the BURST protocol support. Enabled by default.

The 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 MINT 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.

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

- **shortgi**: enables the short guard interval mode. Using short guard interval allows the device to increase its throughput by reducing the time interval between symbols being transmitted. However, this may significantly increase the intersymbol interference and, thus, cause a higher errors rate.

To disable type "**-shortgi**". Enabled by default.

- **noise XXX:** sets Noise Floor Threshold for radio interface. Measured in decibel. By default Noise Floor Threshold is 20 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.

- **pwrctl:** enables Automatic Transmit Power Control (ATPC) on the per-packet basis. When it is enabled ("rf rf5.0 txpwr <power> pwrctl") the system automatically adjusts device's output power to the optimal value that is necessary and sufficient to maintain the maximum productivity of the link in the given conditions. Transmit power adjustment is estimated for each sending packet. The upper bound of output power values that can be set by ATPC is limited by the power value specified in "txpwr" parameter.

  "**-pwrctl**" disables ATPC. Enabled by default.

- **wocd**: disables carrier sense on the radio interface when sending data packets. 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. "-wocd" re-enables the standard mechanism of permanent carrier sense. Initial setting is "-wocd" (carrier sense enabled).

- **modulation**: 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**

- **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)

- **statistics**: displays the radio module's statistics with 1 second update 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
Decrypted frames	Number of successfully decrypted packets
Decrypted errors	Number of errors in the decryption process
Replay drops	Number of packet drops in an aggregate due to the packet sequence errors
Aggr Subframe Retries	Number of packet drops in an aggregate due to protocol excesses (for transmission)
Aggr Full Retries	Number of duplicate aggregates transmitted
Max aggr frames	Maximal detected number of packets in an aggregate
Max aggr bytes	Maximal detected bytes in an aggregate
Encrypted frames	Number of successfully encrypted packets

"rf stat" command output for 2.4GHz devices	
Parameter	**Description**
Beacons Received	number of received service packets (not used)
Beacons Transmitted	Number of transmitted service packets (not used)
Ack Packets Transmitted	Number of transmitted acknowledgement packets
RTS Packets Transmitted	Number of RTS packets transmitted
CTS Packets Transmitted	Number of CTS packets transmitted
PLCP CRC Errors	CRC errors counter
Single Collisions	Single collisions counter
PLCP Format Errors	PLCP format errors number
Polling cache aged	Expired cache lifetime packets number
PLCP Length Errors	PLCP length errors number
No Deferral	Non-deferred packets number
MAC CRC Errors	CRC check errors number
Deferred Protocol	Protocol deferred packets number
Partial Received	Number of partially received packets
Deferred Energy Detect	Number of deferred packets due to media unavailability
SSID Mismatches	Number of SSID mismatches (useless in our case)
Retry Long	Number of retries for the packets which exceeded RTC threshold
AP Mismatches	Access Point mismatches (useless in our case)
Retry Short	Number of retries for the packets which were lower than RTC threshold
Data Rate Mismatches	Data rate mismatches number
Authentication Rejects	Authentication rejects number
Ack Received	Number of received acknowledgements for packets transmitting
Authentication T/O	Number of authentication timeouts
No Ack Received	Number of not acknowledged packets when transmitting
Association Rejects	Association rejects number
CTS Received	Number of received CTS packets
Association T/O	Number of association timeouts
No CTS Received	Number of not received CTS packets
Packets Aged	Number of expired lifetime packets after receiving

- **bcsid**: enables SID broadcasting (in beacon service packets which are regularly 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 MINT protocol exclude the possibility of unauthorized network connection, such device's behavior leads to its unstable work, failures and communication link delays.

  "**-bcsid**" disables SID broadcasting. Disabled by default.

Examples:

```
rfconfig rf5.0 sid 1 bitr 130000 freq 2427 burst
rfconfig rf5.0 bitr 300000 freq 5280 sid 01020304 burst
rfconfig rf5.0 txpwr 18 pwrctl
```

2. MINT ("mint" command)

General Description

Attention! *MINT firmware is not compatible with earlier RMA version. Don't try to update your current network to MINT version from RMA without prior documentation studying and laboratory testing.*

MINT architecture gives a functionality to present a radio interface of a unit (as well as a network connected to it) as a traditional Ethernet in a bus topology. Therefore the unit can have several Ethernet interfaces and several pseudo-interfaces (tun, ppp, null etc). Any of Ethernet interfaces can be united in bridging groups which consist of two or more interfaces. Moreover, routing mode can also be used.

Full syntax:

```
mint IFNAME -type {mesh | master | slave}
mint IFNAME -mode {mobile | nomadic | fixed}
mint IFNAME -nodeid NUMBERID
mint IFNAME -name NAME
mint IFNAME -key SECRETKEY
mint IFNAME -authmode {public | static | remote}
mint IFNAME -[no]authrelay
mint IFNAME –[no]snmprelay
mint IFNAME -[no]replicate [$ACL]
mint IFNAME -[no]autobitrate [+/-DB] | -fixedbitrate
mint IFNAME  –autofactor 1..5 [3]
mint IFNAME -tpcmin dB -tpcmax dB  -tpcadj +/-dB
mint IFNAME -ratefall 0..8 [0]
mint IFNAME -minbitrate XX
mint IFNAME -meshextracost XX
mint IFNAME -extracost XX
mint IFNAME -fixedcost XX
mint IFNAME -maxlinks XX
mint IFNAME [-loamp XX] [-hiamp XX]
mint IFNAME -[no]crypt
mint IFNAME -airupdate {disable | {[active|passive]|force}} [fast|normal|slow]
mint IFNAME -[no]log [detail]
mint IFNAME -roaming {leader | enable [multiBS] | disable}
mint IFNAME profile N [-freq X[,Y,N-M,...] | auto] [-sid X[,Y,..]] [-bitr X]
        [-band {double | full | half | quarter}]
        [-type {master|mesh|slave}] [-key XXX] [-nodeid N]
        [{-minbitr XXX [-autobitr [+/-dB]] | -fixedbitr}]
        [-long {on|off}]
       [enable | disable | delete]
mint IFNAME addnode [-defgw X.X.X.X] [-defmask X.X.X.X]
mint IFNAME addnode -mac X:X:X:X:X:X [-key STRING] [-note STRING] [-maxrate XX]
              [-lip X.X.X.X] [-tip X.X.X.X] [-mask X.X.X.X]
              [-lgw X.X.X.X] [-tgw {X.X.X.X | none}]
              [-lcost XX] [-tcost XX] [{-setpri | -addpri} NN | -1]
              [-disable | -enable | -delete]
mint IFNAME delnode -mac X:X:X:X:X:X
mint IFNAME map [routes | full | swg] [detail] [-m]
mint -[no]colormap
mint IFNAME monitor [-s] [-i SEC] [MAC [MAC ...]] | -[no]audio [full] [-mac MAC]
mint IFNAME rcmd -node {ADDR|all} [-peer] [-self] {-cmd "CMD" | -file URL} [-key KEY] [-quiet]
mint IFNAME -rcmdserver {disable | enable} -guestKey STRING -fullKey STRING
mint IFNAME -odr hub
mint IFNAME -odr spoke [[-]connected [$ACL]] [[-]kernel [$ACL]]
mint IFNAME -odr disable | show
mint IFNAME start | stop | restart | clear
mint IFNAME poll {start [log] | stop | stat [clear]}
mint join IFNAME1 IFNAME2 ...
mint disjoin
mint vers
```

General commands description

Setting the node type

Syntax:

```
mint IFNAME –type {mesh | master | slave}
```

The command sets the type of node.

Three node types are available:

- **MASTER**:
  Master can establish connections with all other types of nodes. He is able to form a network of any topology with other masters or with nodes of **mesh** type.

  On master node a marker access (**polling**) can be enabled. Only one master in a network segment can have this option enabled by means of which forming a star-topology segment (point-to-multipoint). With this, all other nodes break their connections with their respective neighbors (with exception of connections formed by **join**)

  This type of nodes is usually used for static networks with no or small number of nomadic or mobile clients.

- **MESH**:
  This type of node can be a part of a network with any topology. Establishes connections with **master** and **mesh** nodes. The difference between master and mesh is that master nodes will try avoid sending traffic of core transport network (master-master) through mesh nodes by setting the cost of master-mesh connection (from master side) higher (**meshextracost** parameter). Thus, mesh type of nodes can be used on mobile units where connection's parameters might change quickly in time.

  Mesh nodes can work in a marker access node with master node being a polling master. With this, if master turns polling on, mesh node breaks all its connections with other nodes but the master (except join connections). If master node drops off or turns marker access off, mesh node restores its connections with its neighbors (if these connections existed).

- **SLAVE**:
  Can only connect to the node with **master** type. When connection is lost, the device attempts to restore the connection to the master node. Slave node can work with master node using marker access in a classical point-to-multipoint topology.

Example:

```
mint rf5.0 –type master
```

Setting the node mode

Syntax:

```
mint IFNAME –mode {mobile | nomadic | fixed}
```

The command sets the mode of the node. The mode is defined by the application of the node for the network. Modes description:

- **Fixed**. The network node has a fixed allocation and never moves and never is switched off. This is a infrastructure node of the network
- **Nomadic**. Node may change its physical allocation but all the data transmitting is made when the node is not moving (or moving very slowly)
- **Mobile**. The node may move and exchange data while moving

Example:

```
mint rf5.0 –mode nomadic
```

Setting node sequential number

Syntax:

```
mint IFNAME -nodeid NUMBERID
```

The command sets the sequential number for the node. By default, it is set equal to the device's serial number.

The number may be specified in the "XXX.YYY" format reflecting a part of the IP address (both "XXX" and "YYY" numbers can range from 1 to 255).

The parameter is optional.

Example:

```
mint rf5.0 –nodeid 5
mint rf5.0 –nodeid 123.112
```

Setting node name

Syntax:

```
mint IFNAME -name NAME
```

The command sets the name for the node. Node name will be displayed in "**mint map**" set of commands. Node name should not exceed 16 characters. Spaces in the node name are accepted if put between quotation marks.

Example:

```
mint rf5.0 –name My_node
mint rf5.0 -name "Master Unit"
```

Switching to marker access mode (polling)

Syntax:

```
mint IFNAME poll {start [log] | stop | stat [clear]}
```

The command turns on/off polling mode for the master station.

Polling mode is a method of accessing common radio channel under master station control, which consists in centralized distribution of transmission authorization markers by a master station to slaves. This mode greatly improves operational stability and throughput of master stations under conditions of heavy load and signal level misbalance between different slave units. It is particularly useful when slaves 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 mode 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.

The polling mode is enabled on the master station only. Configuration of client units needs not to be modified.

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

```
mint IFname poll start [mi=XX] [ub=XX] [mt=XX]
```

- **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.
- **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 response time for the first key pressed in telnet program. The value is within 3 and 1000ms.
- **MT** - Marker Timeout. The maximum waiting time for a client unit response to a marker or data packet. Expressed in milliseconds; default value is 120ms.

Example:

```
mint rf5.0 poll start ub=250
```

Wireless subscriber stations “isolation” mode

Syntax:

```
mint IFNAME -[no]replicate [$ACL]
```

This feature allows you to do "isolation" of wireless subscriber stations from direct exchange of information with each other in switching mode.

If **mint –noreplicate** option is enabled on the base station then the traffic entering into a wireless network from wired segment of a subscriber station and coming to the base station from this subscriber station won't be transmitted back to the wireless segment by the base station. He may return to the wireless segment only through an external wired switchboard connected to the base station.

The direct exchange is allowed by default (**mint -replicate**).

In addition **$ACL** list of "num" type may be specified (**acl add $ ISOLATE num N1 N2 ...**) with a list of switching group's numbers for which you should enable or disable the listed feature (for all by default).

This feature applies only to traffic entering a wireless network from the wired segment of a subscriber station. Inside a wireless network nodes are all accessible to each other at all times.

Switching to automatic bitrate control mode

Syntax:

```
mint IFNAME –[no]autobitrate [+/-DB]
```

Enables/disables an automatic speed management mode.

In autobitrate mode every device controls the connection parameters independently (amplitude of the received signal, number of ARQs on transmitting, errors, SNR on the opposite side etc) and chooses such transmitting 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.

When no autobitrate is used transmitting speed will be set according to the setting of "bitr" parameter of "rfconfig" command. When autobitrate is used, transmitting speed will be automatically adjusted according to current link conditions. The ranges of speed will be in between the setting of "bitr" parameter in "rfconfig" command (maximal speed) and "minbitrate" parameter (see below). If no "minbitrate" is specified the minimal RF interface speed will be taken as a lowest possible transmitting speed.

Minimal transmitting speed for autobitrate mode can be set with a help of the following command:

```
mint IFNAME -minbitrate BITRATE
```

Example:

```
mint rf5.0 -autobitrate
mint rf5.0 –minbitrate 9000
```

+/-DB option influences the autobitrate function sensitivity. Autobitrate can be forced to set a higher bitrate (mint IFNAME –autobitrate - <number in dB>) even if the signal level is lower than expected on the specified number of dB. Or not to set a higher bitrate (mint IFNAME –autobitrate + <number in dB>) till the signal level won't become higher than expected on the specified number of dB.

To disable the autobitrate mode the following command is used:

```
mint rf5.0 -fixedbitrate
```

In the fixedbitrate mode the actual bitrate is set with the "bitr" parameter of the "rfconfig" command.

mint IFNAME -ratefall 0..8 command allows influencing autobitrate mechanism in the following way: it sets upper bitrate index threshold below which errors and retries checks are not performed, just energetic ability to upper bitrate is taken into consideration. Bitrate indexes are from 1 to 8 and correspond with bitrates available on the device's radio interface (to see bitrate list use «*rf rfX cap*» command). "0" ratefall's value cancels the command.

Example:

```
mint rf5.0 –ratefall 4
```

Setting the connection factor

Syntax:

```
mint IFNAME  –autofactor 1..5
```

This command sets the sensitivity of the device to establish a connection with a candidate via its radio interface. The more the "autofactor" value is the better should be the characteristics of the radio channel between the device and the candidate for establishing a connection.

Default value is 3.

Setting signal levels thresholds

Syntax:

```
mint IFNAME [-loamp XX] [-hiamp XX]
```

- **loamp**. This option sets the minimal signal level for the neighbor. Signal level is measured in dB above the noise threshold for the current bitrate. If the level gets lower than specified value the connection w**ith** a neighbor will be lost. Default value - 2
- **hiamp**. This option sets the minimal SNR for a new neighbor. Signal level is measured in dB above the noise threshold for the current bitrate. If neighbor's signal level is equal or higher than a specified value the node will consider this neighbor to be a candidate. Default value – 6

Example:

```
mint rf5.0 –loamp 2
```

Setting ATPC thresholds

Syntax:

```
mint IFNAME -tpcmin dB -tpcmax dB  -tpcadj +/-dB
```

This command allows controlling ATPC (automatic transmit power control) function behavior. ATPC function is enabled/disabled by "rf <interface> pwrctl" command (see "rfconfig" command description).

- **tpcmin dB**. This option sets the minimal transmit power level in dB ATPC function is allowed to set on the radio interface.

- **tpcmax dB**. This option sets the maximal transmit power level in dB ATPC function is allowed to set on the radio interface.
- **tpcadj +/-dB**. This option influences the optimal power level to be set on the radio interface by the ATPC function. The ATPC can be forced to set higher (tpcadj + <number in dB>) or lower (tpcadj - <number in dB>) power levels compared to the values it estimates itself.

Creating local nodes database

Syntax:

```
mint IFNAME addnode [-defgw X.X.X.X] [-defmask X.X.X.X]
mint IFNAME addnode -mac X:X:X:X:X:X [-key STRING] [-note STRING] [-maxrate XX]
                [-lip X.X.X.X] [-tip X.X.X.X] [-mask X.X.X.X]
                [-lgw X.X.X.X] [-tgw {X.X.X.X | none}]
                [-lcost XX] [-tcost XX] [{-setpri | -addpri} NN | -1]
                [-disable | -enable | -delete]
```

This set of parameters defines the nodes with which a node can work with. The following parameters can be specified:

- **mac**. This parameter is mandatory. X:X:X:X:X:X is a MAC-address of the node with which a connection can be established.
- **key**. Unique unit's key (key word or phrase up to 64 characters long; if contains spaces should be put into quotes). Used in authentication procedures. The same key should be specified in the settings of the connecting unit ("mint IFNAME –key").
- **lip**. Local IP-address. This address will be assigned to this unit when the connection with a remote is established
- **tip** and **mask**. Target IP-address and mask. This address will be assigned to the remote side when a connection is established. The mask is applied to both Local IP and Target IP. If mask is not specified these addresses will not be used
- **lgw**. Local gateway IP-address (will be assigned to the local node once connection is established)
- **tgw**. Target gateway IP-address (will be assigned to the remote node once connection is established). **None** option forbids providing information about default gateway (that is set by "addnode –defgw" command) to the remote node.
- **lcost**. Local cost of the connection to this neighbor from current node. If not specified, MINT will automatically calculate the cost
- **tcost**. Target cost of the connection from this neighbor to the current node. If not specified, MINT will automatically calculate the cost. If **lcost** and **tcost** parameters are set on a pair of neighbors, **lcost** has a higher priority.
- **enable/disable/delete**. Self-explanatory – enables, disables or deletes a record in a local database.
- **maxrate.** Target node maximum bitrate in kilobit per second.

- **setpri | addpri.** This options allows setting/increasing the priority of packets passing through to the specified node. “**Setpri**" parameter is used to change a priority to the value specified in the command. When using “-1” value a package priority is dropped to the lowest priority. "**Addpri**" is used to change a priority only in case it is higher than the previous one (Note: the smaller is the value the higher is the priority). So you can only increase priority using "**addpri**" parameter.
- **note.** This option allows making some word note (description) for the specified node.

Example:

```
mint rf5.0 addnode –mac 000028BAF234 –lip 1.1.1.1 –tip 1.1.1.2 –mask
255.255.255.252 –lcost 120
```

For easy subscriber nodes definition in the local database of the given node (Base Station) "mint addnode" command is updated with two options: "**-defgw X.X.X.X**“ and “**-defmask X.X.X.X**".

- **-defgw X.X.X.X**. Sets default gateway.
- **-defmask X.X.X.X**. Set default mask.

When mask or gateway values are not defined for the subscriber node then default gateway or default mask will be used for this node. Thus, to add a subscriber node to the local database it is enough to define MAC-address (mac), target IP-address (tip) and a key:

```
mint rf5.0 addnode -mac 000435567322 -tip 10.1.1.1 -key SecretKey1
```

If a key is not specified for a subscriber node then Base Station’s key is assigned to this node.

⚠ Information about default gateway (that is set by «addnode -tgw / addnode –defgw» commands) is not provided to a subscriber node in case IP-address and network mask is not specified.

Deleting a node from local database

Syntax:

```
mint IFNAME delnode -mac X:X:X:X:X:X
```

The command deletes a record created using “addnode” command with a corresponding MAC-address.

Example:

```
mint rf5.0 delnode –mac 000028BAF234
```

Managing MINT protocol

Syntax:

```
mint rf5.0 start|stop
```

The command starts or stops MINT protocol.

Example:

```
mint rf5.0 start
```

Fixed cost setting

Syntax:

```
mint IFNAME –fixedcost XX
```

This command will force all the costs for all the units connected to this unit to be fixed at the value specified in the command.

Example:

```
mint rf5.0 –fixedcost 120
```

Extra cost setting

Syntax:

```
mint IFNAME –extracost XX
```

extracost – is configured for the interface. Sets an extra cost for all connections on this interface. The value of the parameter is added to the cost automatically calculated by MINT protocol. Value of this parameter can only be positive. Zero value disables the parameter.

Example:

```
mint rf5.0 –extracost 60
```

Mesh extra cost setting

Syntax:

```
mint IFNAME –meshextracost XX
```

meshextracost – is configured for the interface. Sets an extra cost for all connections of master node with its mesh nodes. 500 – by default.

Example:

```
mint rf5.0 –meshextracost 300
```

Join cost setting

Syntax:

```
mint IFNAME –joincost XX
```

joincost – is configured for the interface. Sets the cost of all connections on the interface which were established by means of join (3 – by default). Zero value disables the parameter

Example:

```
mint rf5.0 –joincost 60
```

MINT log settings

The following command is used to control log settings for MINT protocol:

```
mint IFNAME -[no]log [detail]
```

Three different modes are available:

- No logging. “**-nolog**” option is used
- Limited logging. “**-log**” option is used. The messages on connecting/disconnecting neighbors will be put to the system log
- Detailed logging. “**-log detail**” option is used. Along with the messages from limited logging mode, messages on changing costs of the routes and changing bitrates (in autobitrate) mode will be put to the system log

Example:

```
mint rf5.0 –log detail
```

This command will turn full logging on.

MINT protocol version

Syntax:

```
mint vers
```

The command shows current version of MINT protocol.

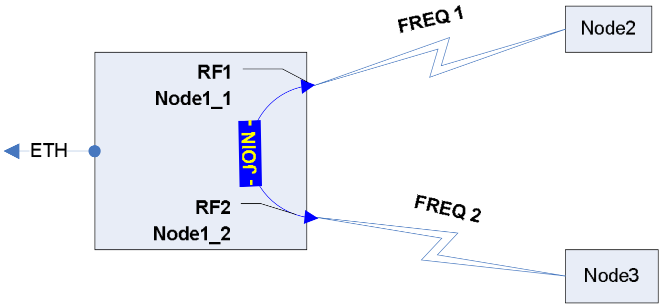
Joining interfaces

Syntax:

```
mint join IFACE1 IFACE2 ...
```

Among all of exiting features of MINT architecture one can mention **join** capability which allows joining several interfaces of one unit into one mesh network.

Some units can have two or more radio interfaces of different types. Each of these interfaces may act as an independent MINT network node. However, the nodes from different networks will not be able to connect to each other as radio interface parameters will be different (frequencies, modulations or other parameters) and other limitations (authentication parameters, secret keys etc) will take place. **JOIN** function allows for two or more radio-interfaces of one unit to interconnect with each other as if they are two nodes of one network.



```
mint join rf4.0 rf4.1
mint map
================================================
Interface rf4.0, node 000000000011 "Node1_1" id:11 (mesh)

 2 Neighbors:
  00020 Node2           000000000002, Cost=40 , I/O=24/27 <36/36> /mesh/
  00012 Node1_2          000000000012, Cost=3  , I/O=0/0    <0/0>  /join/

Interface rf4.1, node 000000000012 "Node1_2" id:12 (mesh)

 2 Neighbors:
------------
  00020 Node3           000000000003, Cost=40 , I/O=24/27 <36/36> /mesh/
  00030 Node1_1          000000000011, Cost=3  , I/O=0/0    <0/0>  /join/
```

As you can see from the example, each interface shows two neighbors. In reality the exchange of data between interfaces does not involve their physical activity, energetic parameters of the connection (signal levels and data rates) are not shown (zero). And this kind of connection has a constant and a very low cost.

In order to disjoin interfaces to make them independent, the following command is used:

```
mint disjoin
```

Pseudo radio interface (prf)

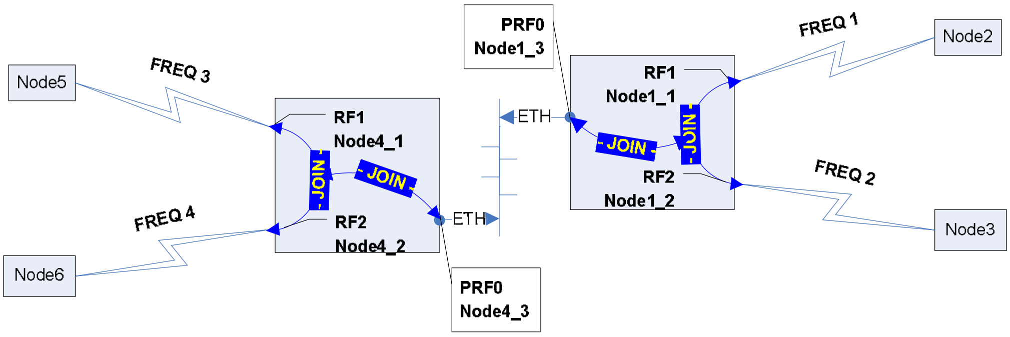
MINT protocol can work not only via radio but via wired Ethernet interface. “**prf**” pseudo radio-interface is used for this purpose. This interface can be “attached” to the physical interface like it’s done with vlanX interfaces.

```
prf 0 parent eth0
ifconfig prf0 up
```

Please find more information on prf interface in “**prf**” command description.

Such **pseudo radio-interface** can be configured as a MINT network node and even can be joined with other interfaces. From MINT protocol point of view, this pseudo interface will look like a usual radio interface through which a node can find neighbor and establish a connection with it.

```
mint prf0 start
mint join rf4.0 rf4.1 prf0
```



In this example we have joined several distributed (possibly even geographically distributed) network segments. Combining “joins” for different radio interfaces and pseudo radio interfaces the network will obtain a good number of alternative paths in any direction thus providing with the best delivery quality and less bottlenecks.

Important. If several interfaces are used with “join” function, when configuring switch groups only one of those joined interfaces should be mentioned in switch group configuration.

```
mint join rf4.0 rf4.1
switch group 1 add eth0 rf4.0
```

Nodes authentication

Setting the secret key:

```
mint IFNAME –key SECRETKEY
```

This command sets the secret key for the current node. See different authentication modes descriptions below to learn how it is being used. The key can be up to 64 characters long and should not contain spaces (or should be put in quotes).

```
mint IFNAME -authmode {public | static | remote}
```

The command sets the type of nodes authentication.

There are three types of nodes authentication available:

- **public** – all nodes have the same key (password) for access. The simplest case of authentication. It can be used for small workgroups, point-to-point connections, mass public access networks and for MINT architecture testing purposes. Any two nodes of the network can establish a connection (given other settings are suitable) if their keys are equal. In public mode, having found a potential neighbor a node check for its information in the local database (defined by "mint IFNAME addnode" commands). If requested information is found, a key from a local database will be used. Otherwise, it is assumed that neighbor's key corresponds with node's own key ("mint IFNAME –key" parameter)

- **static** – every node has a full list of nodes (including their parameters and access keys) with which a connection can be established. This mode is suitable for an autonomous area of service with no need of centralized management and monitoring. Obviously, nodes that are included in each others access lists (local databases) should have a physical ability to connect to each other in order to establish a connection. In static mode each node must have a list of all permitted neighbors in a local database formed by a set of "mint IFNAME addnode" commands. If no information on the neighbor is found in the database the connection is being rejected.

- **remote** – centralized authentication mode with remote server (e.g. RADIUS or relay). In this mode any node can request the information from a remote authentication server (remote authentication server parameters are set using "AAA" command). This means that the node must have an access to this server (e.g. using IP).

A node having a local database of its neighbors or having an access to a remote authentication server can be configured as an **authentication relay**. For this purpose the following command is used:

```
mint IFNAME  –[no]authrelay
```

The information about authentication relay will be automatically distributed throughout the MINT network. Nodes which use remote mode of authentication but both do not have access to the remote server and do not have the information in their local database will use authentication relay in order to obtain the keys of potential neighbors.

```
mint IFNAME  –[no]snmprelay
```

The information about SNMP relay will be automatically distributed throughout the MINT network. Nodes will use remote SNMP services.

Example 1:

Nodes A and B use the same key and can establish a connection with each other in public authentication mode.

Node A:

```
mint rf5.0 –key SECRETKEY
mint rf5.0 –authmode public
```

Node B:

```
mint rf5.0 –key SECRETKEY
mint rf5.0 –authmode public
```

Example 2:

Nodes A and B have different keys but they can establish a connection with each other using their local databases.

Node A:

```
mint rf5.0 –key SECRETKEY
mint rf5.0 –authmode public
mint rf5.0 addnode -mac B:B:B:B:B:B -key KEY2
```

Node B:

```
mint rf5.0 –key KEY2
mint rf5.0 –authmode public
mint rf5.0 addnode -mac A:A:A:A:A:A -key SECRETKEY
```

Moreover, each of these two nodes can set up connections with other nodes working in public mode if their keys correspond with each other.

Example 3:

Node A has a local database and acts as an authentication relay. Node B does not have a database and uses a relay in remote mode

Node A:

```
mint rf5.0 –key KEY1
mint rf5.0 –authmode static
mint rf5.0 –authrelay
mint rf5.0 addnode -mac B:B:B:B:B:B -key KEY2
mint rf5.0 addnode -mac A:A:A:A:A:A -key KEY3
```

Node B:

```
mint rf5.0 –key KEY2
mint rf5.0 –authmode remote
```

Node B will be getting neighbors' information via relay (Node A).

If Node A is switched to remote mode and there is no information in the local database, the authentication request will be forwarded to the remote server (if specified and accessible) or to another authentication relay.

Automatic over-the-air firmware upgrade

```
mint IFNAME -airupdate {disable | {[active|passive]|force}} [fast|normal|slow]
```

This set of commands manages the Automatic Over-the-air Firmware Update system.

What is it?

The AirUpdate system provides with an easier ways of massive firmware upgrade in the MINT network for a big number of the nodes (same type). In order to do that only one unit of each type should be manually (or through the scheduler) upgraded – other units will get new firmware automatically.

How does it work?

Every unit can be configured for AirUpdate in passive or active mode. Active units periodically (every 30 minutes) announce the information about their firmware to the MINT network. The information includes the version of the firmware and the time of uninterrupted (without reboots) work with this version. All units of MINT network (both active and passive) receive and accumulate the information from active units choosing the source with the latest version of firmware with which the source has worked for the longest period of time.

After the period of information accumulation passive units send their requests for the new firmware to the chosen source. Active units form a list of requests and perform a group distribution of the firmware using special MINT multi-address distribution protocol.

The period of information accumulation can be changed using **fast**, **normal** and **slow** parameter of the command.

In **fast** mode the unit will wait for the potential source of the firmware to work with new version within two hours with no reboots. Only after two hours the request will be sent.

In **normal** mode the waiting period is 7 hours; in **slow** – 24 hours

By default, **passive normal** mode is turned on.

For immediate firmware upgrade there is a special option "**force**". The command is not saved in the configuration and acts as a signal for all units to send their requests for upgrade regardless the work mode and information accumulation time period.

If during the process of new firmware distribution an error occurs (or link loss) the passive unit will stop the upload process and will resend its request after getting an announcement.

Examples

The unit is in the active mode sending announcements about new firmware version. If units with newer firmware version are found on the network, the upload request will be sent in no less than 7 hours after uninterrupted work of the announcement source:

```
mint rf5.0 –airupdate active normal
```

The unit is the passive mode waiting for the source of the latest firmware version to work with it during no less than 24 hours:

```
mint rf5.0 –airupdate passive slow
```

The operator decides to immediately upgrade all the units with new firmware:

```
mint rf5.0 –airupdate force
```

The unit is not participating in AirUpdate process. It does not send announcements and does not generate requests for upgrades:

```
mint rf5.0 –airupdate disable
```

Over-the-air encryption

All MINT protocol messages are automatically encrypted. This feature is always turned on.

AES-128 CCMP hardware encryption is used on the device.

For full data encryption the following command is used:

```
mint IFNAME -[no]crypt
```

The command turns on/off over-the-air encryption. Encryption has the following features:

- Each node has its own key for outgoing traffic encryption

- There is no restrictions on the number of nodes that use outgoing traffic encryption in the MINT network
- Every three minutes the key is dynamically changed

Example:

mint rf5.0 –crypt

Learning the network neighbors

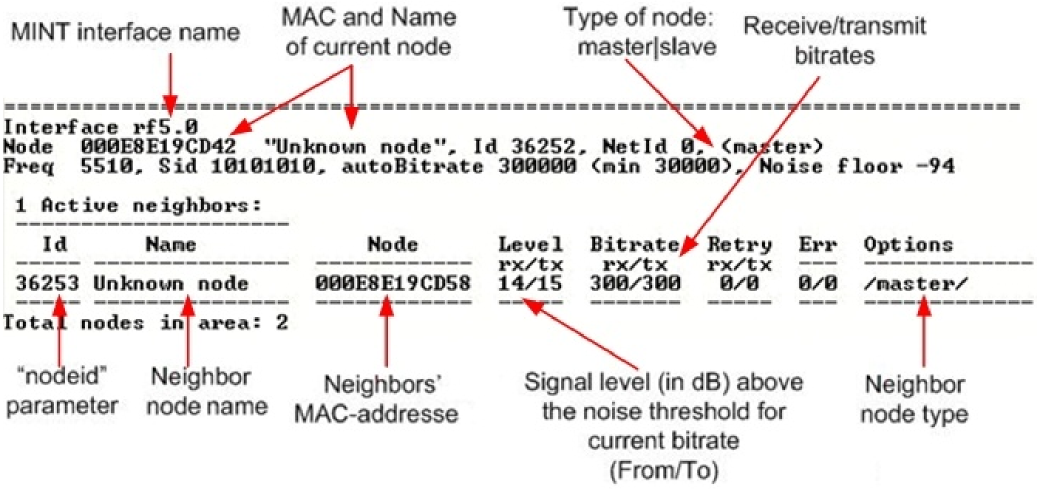
The following command is used to learn the state of the connections with neighbors:

mint IFNAME map [routes | full | swg] [detail] [-m]

mint -[no]colormap

Three options are used:

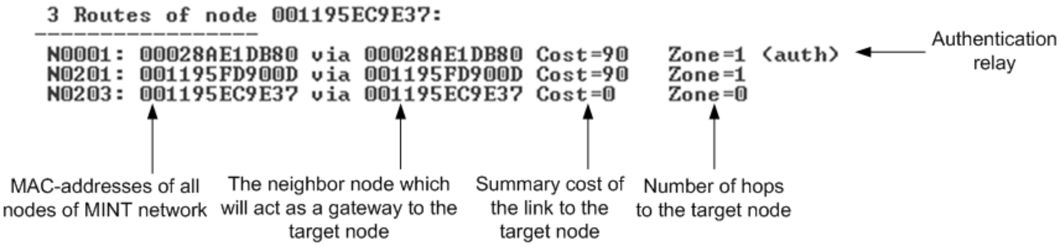
- The default output is displayed:



```
==========================================================================
Interface rf5.0
Node   000E8E19CD42  "Unknown node", Id 36252, NetId 0, (master)
Freq   5510, Sid 10101010, autoBitrate 300000 (min 30000), Noise floor -94

 1 Active neighbors:
 ----------------------
   Id       Name              Node        Level  Bitrate  Retry  Err  Options
 ----- ---------------  ------------  rx/tx   rx/tx   rx/tx  ---  -------------
 36253 Unknown node     000E8E19CD58  14/15  300/300   0/0   0/0  /master/
 ----- ---------------  ------------  -----  -------  -----  ---  -------------
Total nodes in area: 2
```

MINT interface name
MAC and Name of current node
Type of node: master|slave
Receive/transmit bitrates
"nodeid" parameter
Neighbor node name
Neighbors' MAC-addresse
Signal level (in dB) above the noise threshold for current bitrate (From/To)
Neighbor node type

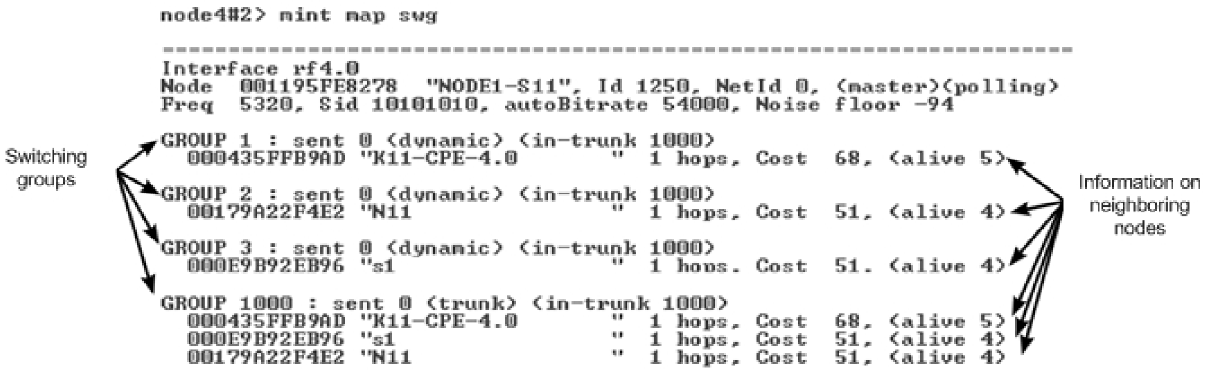
- Routes. The following output is displayed:



```
 3 Routes of node 001195EC9E37:
-------------------
 N0001: 00028AE1DB80 via 00028AE1DB80 Cost=90   Zone=1 (auth)
 N0201: 001195FD900D via 001195FD900D Cost=90   Zone=1
 N0203: 001195EC9E37 via 001195EC9E37 Cost=0    Zone=0
```

Authentication relay
MAC-addresses of all nodes of MINT network
The neighbor node which will act as a gateway to the target node
Summary cost of the link to the target node
Number of hops to the target node

- Full. A combination of "neighbors" and "routes" modes
- Swg. This option is used when switching groups are created in MINT network. It shows in which switching groups neighbor nodes are included:

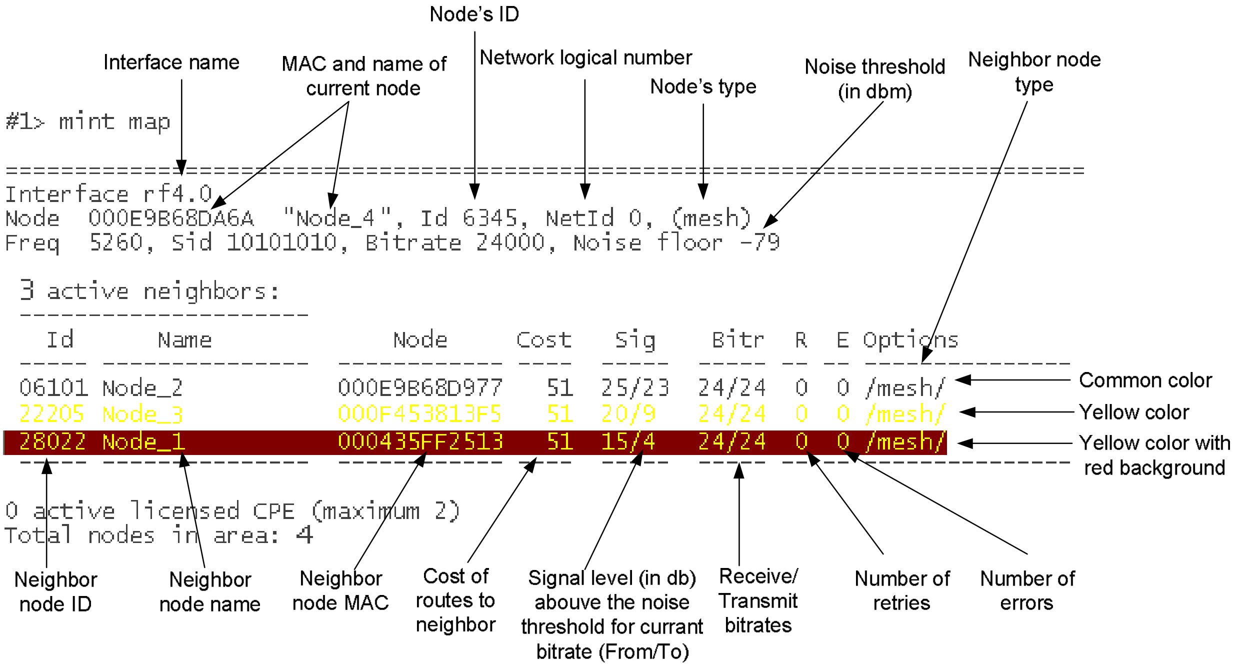


```
node4#2> mint map swg

===========================================================================
Interface rf4.0
Node   001195FE8278   "NODE1-S11", Id 1250, NetId 0, (master)(polling)
Freq   5320, Sid 10101010, autoBitrate 54000, Noise floor -94

GROUP 1  : sent 0 (dynamic) (in-trunk 1000)
  000435FFB9AD "K11-CPE-4.0          "  1 hops, Cost  68, (alive 5)

GROUP 2  : sent 0 (dynamic) (in-trunk 1000)
  00179A22F4E2 "N11                  "  1 hops, Cost  51, (alive 4)

GROUP 3  : sent 0 (dynamic) (in-trunk 1000)
  000E9B92EB96 "s1                   "  1 hops. Cost  51. (alive 4)

GROUP 1000  : sent 0 (trunk) (in-trunk 1000)
  000435FFB9AD "K11-CPE-4.0          "  1 hops, Cost  68, (alive 5)
  000E9B92EB96 "s1                   "  1 hops, Cost  51, (alive 4)
  00179A22F4E2 "N11                  "  1 hops, Cost  51, (alive 4)
```

Parameters:

- **-detail** – shows the following information for each link with a neighboring node: distance do the neighboring node in kilometers, load on receive/transmit in Mbps, on receive/transmit in packets per second, link «Cost», main IP-address of the neighboring node.**-**
- **-m** – shows I/O signal levels relative to minimal receive/transmit bitrate. (Without **–m** relative to current bitrates).

Mint –[no]colormap option enables/disables color indication of the command.



```
#1> mint map

=========================================================================
Interface rf4.0
Node   000E9B68DA6A   "Node_4", Id 6345, NetId 0, (mesh)
Freq   5260, Sid 10101010, Bitrate 24000, Noise floor -79

 3 active neighbors:
 ---------------------
   Id       Name              Node       Cost   Sig    Bitr   R   E Options
 ----- ---------------- ------------- ---- ----- ----- -- -- ----------------
 06101 Node_2           000E9B68D977    51  25/23  24/24   0   0 /mesh/
 22205 Node_3           000F453813F5    51  20/9   24/24   0   0 /mesh/
 28022 Node_1           000435FF2513    51  15/4   24/24   0   0 /mesh/
 ----- ---------------- ------------- ---- ----- ----- -- -- ----------------

0 active licensed CPE (maximum 2)
Total nodes in area: 4
```

Color indication of «mint map» command output:

- **Common color** identifies neighbor nodes that have acceptable characteristics of a link to the current node.
- **Yellow color** identifies neighbor nodes that potentially may have problems with sustainability and quality of a link to the current node. In this case link quality can be improved through the change of certain parameters (for example, lowering bitrates).
- **Yellow color with red background** identifies neighbor nodes that have unsatisfactory characteristics of a link to the current node. For example, neighbor nodes that have low characteristics of a link on the

lowest possible bitrate or have errors are marked this way. In this case link quality can be improved by such actions as antenna alignment, cable connectivity testing and so on.

When neighbor nodes are marking with certain color style it is not only signal level but also number of retries and errors are taken into consideration.

Continuous signal levels monitoring

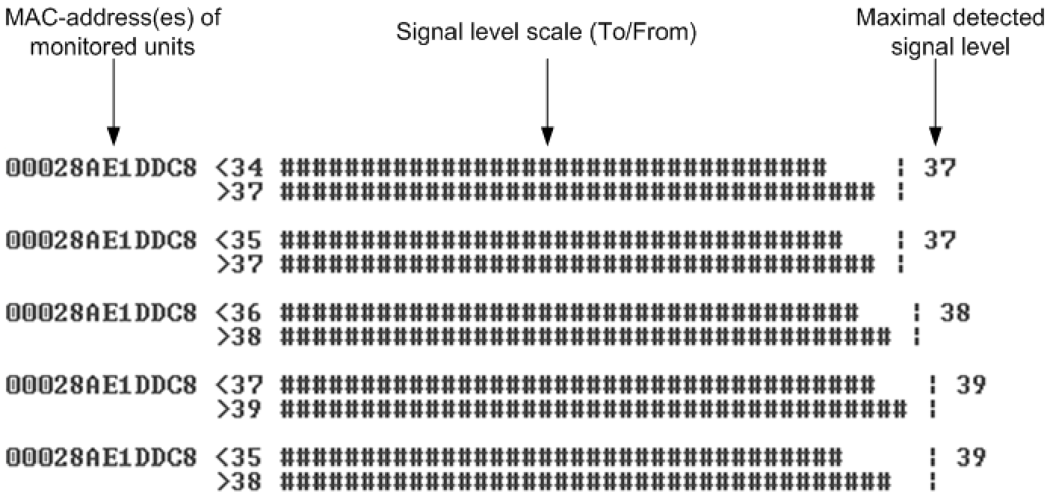
Syntax:

mint IFNAME monitor [-s] [-i SEC] [MAC [MAC ...]]

If no MAC-addresses are specified, the output of command will contain the information about all neighbors and candidates of the current node.

Instead of MAC-addresses “nodeid” and/or “name” of the node can be specified.

The sample output of the command is presented below.


MAC-address(es) of monitored units
Signal level scale (To/From)
Maximal detected signal level

```
00028AE1DDC8 <34 ##################################    ¦ 37
             >37 ##################################### ¦

00028AE1DDC8 <35 ###################################   ¦ 37
             >37 ##################################### ¦

00028AE1DDC8 <36 ####################################   ¦ 38
             >38 ###################################### ¦

00028AE1DDC8 <37 #####################################   ¦ 39
             >39 ####################################### ¦

00028AE1DDC8 <35 ###################################     ¦ 39
             >38 ######################################  ¦
```

“**-s**” option keeps the output within one screen without line-by-line output

“-i SECONDS” set the interval for information output in seconds

“Audiomonitor” mode:

Syntax:

mint IFNAME monitor -[no]audio [full] [-mac MAC]

"**-audio**" option enables sound indication (AudioMonitor mode). "**-noaudio**" turns off the sound indication.

In "**full**" mode, audio monitoring is performed according to a maximal receiving level among neighbors and candidates. If "full" parameter is not specified an arithmetical mean of receive and transmit signal is calculated (only for neighbors). The neighbor for which this value is maximal will be indicated in audiomonitor.

“-**mac MAC**” option allows to perform audio monitoring only for one neighbor with the specified MAC-address.

☝ **mint IFNAME monitor -audio** *command starts working immediately and can be saved in the configuration which allows using it even after unit's reboot*.

Frequency roaming

For a flexible management of frequency resource, higher noise immunity and throughput optimization InfiNet Wireless equipment supports frequency roaming capability based on MINT protocol.

Roaming is turned off by default – that means that the unit works using fixed radio interface configuration.

Any node of the network can be set up as a **roaming leader.** Roaming leader will define required radio frequency parameters of the wireless network. Roaming leader also works with a fixed radio interface parameters, however its radio parameters configuration is transmitted over the network in special packets so every node of the network knows whether it is connected to the roaming leader or to the network that has a roaming leader. If the network has several roaming leaders, their parameters should be identical. Roaming leader also supports DFS and Radar Detection features (if a special license is installed for selected countries).

Other network nodes can use roaming in order to search for the roaming leader or the network having a roaming leader (**roaming enable**). The search is implemented by switching between different sets of radio parameters that are defined in profiles. Each profile contains a fixed set of radio interface parameters which are set on each iteration of the search. Heuristic search algorithm can quickly evaluate general air media parameters and chooses the profile which defines the most suitable network.

The "**multiBS**" option enables the slave node to constantly check the link quality and try to find another BS if the quality become worse. When the option is disabled then if the link breaks the node will firstly try to reconnect to the same BS regardless of the link quality.

Profile parameters:

- **freq X[,Y,N-M,...] | auto** – radio interface frequency or list of frequencies. **Auto** keyword can be used - in this case all frequencies that the unit supports will be used
- **sid X[,Y,..]** – SID of the radio interface (or list of SIDs)
- **bitr X** – bitrate of the radio interface. Acts as a top limit for the bitrate if autobitrate mechanism is turned on
- **band {double|full|half|quarter}** – defines the channel width for the profile. If profiles use different channel widths, **auto** mode for frequency cannot be used
- **type {master|mesh|slave}** – node type
- **key XXX** – secret key
- **nodeid N** – node ID
- **fixedbitr** – sets fixed bitrate for the node
- **minbitr XXX** – minimum bitrate for operation in "autobitrate" mode
- **autobitr [+/-dB]** – operation mode with automatic bitrate control. **[+/-dB]** parameter allows to manage bitrate control sensitivity
- **long {on|off}** – enables/disables compulsory "long" mode
- **enable | disable | delete** – enables, disables or deletes the profile.

Syntax:

```
mint IFNAME -roaming {leader | enable [multiBS] | disable}
mint IFNAME profile N [-freq X[,Y,N-M,...] | auto] [-sid X[,Y,..]] [-bitr X]
        [-band {double | full | half | quarter}]
        [-type {master|mesh|slave}] [-key XXX] [-nodeid N]
```

```
[{-minbitr XXX [-autobitr [+/-dB]] | -fixedbitr}]
[-long {on|off}]
[enable | disable | delete]
```

Samples:

```
mint rf5.0 profile 1 -freq 5920 -sid ABCDE
mint rf5.0 profile 2 -freq 5960 -sid ABCDE disable
mint rf5.0 profile 3 -freq auto -sid DEAD
mint rf5.0 roaming enable
```

Remote command management

Remote command management allows one MINT node to perform commands on one other or all MINT nodes in the network.

Options:

- **-rcmdserver {disable | enable}** – disables/enables remote control management mode (enable by default)
- **-guestKey STRING** – guest key. Guest key allows to perform read only commands on the node
- **-fullKey STRING** – full key. Full key grants full access to the node (all commands can be performed)
- **-node {ADDR|all}** – Mac-address of the destination node or access to all MINT nodes
- **[-peer]** – performs commands only on the nodes that is connected to the given device directly
- **[-self]** – performs commands also on the device itself
- **{-cmd "CMD" | -file URL}** – command to be performed on the remote unit or root to a command txt file by ftp
- **[-key KEY]** – access key
- **[-quiet]** – disables writing replies from remote devices to a system log.

Syntax:

```
mint IFNAME rcmd -node {ADDR|all} [-peer] [-self] {-cmd "CMD" | -file URL} [-key KEY] [-quiet]
mint IFNAME -rcmdserver {disable | enable} -guestKey STRING -fullKey STRING
```

Samples:

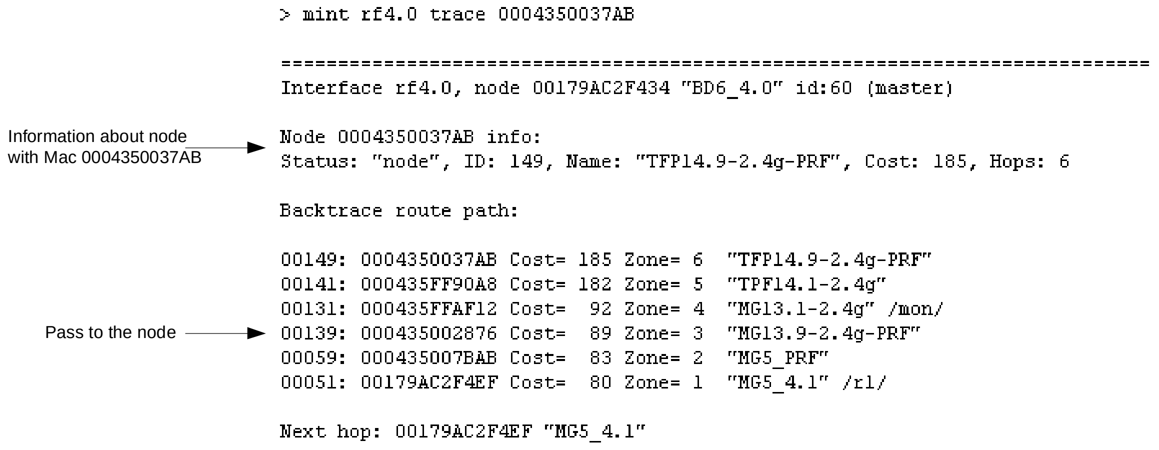
```
mint rf5.0 rcmd -node all -cmd "co sh"
mint rf5.0 rcmd -node all -file ftp_name:ftp_pswd@192.168.100.21/1.txt
```

Trace command

Syntax:

mint IFNAME trace MAC

Trace command allows viewing information about node by its MAC address: its status, ID, name, cost, number of hops. This command also shows information about cost optimal pass to the node at the current time.



```
> mint rf4.0 trace 0004350037AB

============================================================================
Interface rf4.0, node 00179AC2F434 "BD6_4.0" id:60 (master)

Node 0004350037AB info:
Status: "node", ID: 149, Name: "TFP14.9-2.4g-PRF", Cost: 185, Hops: 6

Backtrace route path:

00149: 0004350037AB Cost= 185 Zone= 6  "TFP14.9-2.4g-PRF"
00141: 000435FF90A8 Cost= 182 Zone= 5  "TPF14.1-2.4g"
00131: 000435FFAF12 Cost=  92 Zone= 4  "MG13.1-2.4g" /mon/
00139: 000435002876 Cost=  89 Zone= 3  "MG13.9-2.4g-PRF"
00059: 000435007BAB Cost=  83 Zone= 2  "MG5_PRF"
00051: 00179AC2F4EF Cost=  80 Zone= 1  "MG5_4.1" /rl/

Next hop: 00179AC2F4EF "MG5_4.1"
```

On-Demand Routing

On-Demand Routing (ODR) protocol is a structure of the MINT protocol. It provides dynamic routing capabilities in stub networks without the use of any routing protocol.

The main advantage of using ODR is the increase in the available bandwidth of your network by eliminating the service traffic of a separate routing protocol whilst still maintaining dynamic routing functionality. The ODR protocol propagates IP prefixes on the Layer 2 using the MINT protocol.

ODR is applicable only for stub networks of the hub-and-spoke topology, when all nodes (spokes) are connected only to a hub node. An example of the hub-and-spoke network is a simple Point-to-Multipoint network where each subscriber station has the only wireless connection to the Base Station.

To set a unit as a hub the following command is used:

mint IFNAME -odr hub

To set a unit as a spoke:

mint IFNAME -odr spoke [[-]connected [$ACL]] [[-]kernel [$ACL]]

Where “connected” option allows announcing IP-addresses/networks set on the spoke’s own interfaces. “Kernel” option allows announcing static routes (set with the “route add” command).

Also one can specify a list of IP-addresses and networks in the command using Access Control List (“$ACL”). Please see the Access Control List description in the corresponding chapter of this manual.

To show the current state of the protocol and established connections the following command is used:

mint IFNAME -odr show

To disable the on-demand routing on the unit the following command is used:

```
mint IFNAME -odr disable
```

Upgrading current RMA network to MINT

Attention! *It is recommended to study MINT technology features and test your basic configuration skills on the test devices before complying this instruction.*

It is also recommended to consider principles of new network building in advance: wil it be mesh or point-to-multipoint network, how and what routing will be used, will it be a switched network?

1. Perform the following actions on each client device:

- Note down or remember MAC address of radio interface which is used to connect to base station. You can view it using command:

```
Ifc rf5.0
```

- Save current configuration (it is supposed that client device is connected to base station in the moment) using command:

```
config save
```

- Upload MINT firmware using command:

```
fl get user:password@server/file
```

2. Perform restart of all the client devices (one after another) using command:

```
  restart y
```

3. Upload MINT firmware on the base station and restart base station.

After the restart all devices will start with MINT firmware. Upon discovering the old configuration (RMA) new MINT firmware will start neighbor search protocol on all radio interfaces of the device with default parameters (master, autobitrate, hiamp=4).

Radio interface parameters will be delivered from the old configuration (at the time of its last saving).

This is enougth for device to connect to MINT Network. The devices will lose IP address assigned by RMA.

4. Ensure all devices have connected to base station using “mint map” command.

5. Assign appropriate IP address on base station radio interface, for example:

```
ifc rf5.0 10.0.0.254/24 up
```

6. Using "mint rcmd" command assign IP addresses on radio interfaces of all client devices addressing them using MAC adress (you can miss colon in MAC addresses), for example:

```
mint rf5.0 rcmd -node 17:9a:c2:f4:34 -cmd "ifc rf5.0 10.0.0.1/24 up; co save;"
mint rf5.0 rcmd -node 17:9a:c3:ad:46 -cmd "ifc rf5.0 10.0.0.2/24 up; co save;"
mint rf5.0 rcmd -node 179ab1f391      -cmd "ifc rf5.0 10.0.0.3/24 up; co save;"
```

...

and so on.

After that all client devices will be accessible from base station by IP (if it wasn't rejected by configuration options).

7. Using telnet perform the rest necessary configuration on each device for smooth network functioning (routing and so on).

If IP access is not obtained, it is always possible to verify or correct configuration of client devices using "mint rcmd" command:

```
mint rf5.0 rcmd -node 0023113231 -cmd "co show" -reply
```

with **–reply** parameter command result will appear in a local system log (sys log show).

3. Prf

MINT architecture protocol can work not only via radio interfaces but through wired Ethernet interfaces. For this purpose, "**prf**" interface (pseudo radio interface) is used which can be attached to the physical interface like **vlanX** interfaces.

Syntax:

```
prf N parent IFACE [hwmtu N] [channel N]
prf N clear
```

"**prf**" driver allows sending network frames of a proprietary format through Ethernet network. When sending frames to Ethernet, the driver encapsulates them into IEEE802.3 frames.

If specified maximal length of Ethernet frame is not enough to send a fully encapsulated original frame, this frame will be fragmented and then assembled at the remote side. Fragmentation algorithm uses the feature of Ethernet to avoid a delivery of frames out of order.

There is a possibility to create up to four isolated pseudo radio networks within one Ethernet segment. Each **prf** interface can be assigned a number from 0 to 3. Frames sent to Ethernet with a specific channel number can be received on other nodes only be **prf** interfaces with the same channel number.

prf command options:

- parent IFACE

IFACE – interface through which encapsulated (and, possibly, fragmented) packets will be sent. Only Ethernet-type interfaces should be configured.

- hwmtu N

N – maximal length of Ethernet frame sent to the parent interface.

- channel N

N – channel number with which the frames are sent and received by parent interface.

- clear – deletes **prf** with a specified number

More information on **prf** interface usage one can find in “**mint**” command description.

4. Link Aggregation (“lag” command)

Link Aggregation is using multiple physical channels in parallel as one logical channel to increase the total capacity of the link and provide redundancy.

For example, it can be used to set up a high-capacity device-to-LAN backbone utilizing two Ethernet interfaces of the device. The total link speed of such backbone will be a sum of the speeds of the Ethernet interfaces. Moreover, in case of one Ethernet connection failure its traffic will be passed to the other one without stooping the services. If later the fallen Ethernet connection will be reestablished the traffic will be sent over both Ethernet interfaces again.

Infinet Wireless LACP (Link Aggregation Control Protocol) realization fully complies with the IEEE 802.3ad standard for link aggregation. In addition, special proprietary LACP mode can be enabled that provides enhancement of the efficiency of the process over the standard mode.

For LACP configuration on IW devices the “lag” command is used.

Syntax:

```
lag N [command] [(port|-port) (IFNAME | IFNAME...)]
  where [command] is:
  status      - show lag interface status
  migrate      - enable session migrate
  -migrate     - disable session migrate
  balance      - enable session-oriented load balancing
  -balance     - disable session-oriented load balancing
  mode         - switch lag inteface mode. Modes: (stand|fast)
  clear       - remove lag interface from configuration
  loadm        - show realtime per-port load status
```

Description:

To enable link aggregation on the device the following command is used:

```
lag N port (IFNAME | IFNAME...)]
```

This command creates logical “lagN” interface in the system and assigns parent physical interfaces to it.

For example:

```
lag 1 port eth0 eth1
```

In this example “lag1” interface is created and two Ethernet interfaces of the device are assigned.

To delete one ore more parent interfaces from LACP configuration the “-port” option is used. For example:

```
lag 1 -port eth1
```

The following commands are used to manage LACP operation and get the statistics:

- **lag N status** – allows to view the aggregation id, the load on each of the parent interfaces and its status
- **lag N migrate / lag N -migrate** – allows/disallows existing sessions migration between the available parent interfaces in case of the difference in the overload. By default, migration is allowed
- **lag N balance / lag N -balance** – allows/disallows load-depending dispersion of new-coming sessions between the available parent interfaces. By default, balancing is enabled.
- **lag N mode** – allows to set "standard" or "fast" LACP mode. In "standard" mode LACP acts in accordance to the IEEE 802.3ad standard. In "fast" mode LACP uses IW proprietary extension that enhances its speed of response on the link state changes, optimizes the overall aggregate system performance and delivers more accurate statistics
- **lag N clear** – removes the "lagN" interface from the device configuration
- **lag N loadm** – shows the load statistics in real time on each of the parent interfaces.

5. Ltest (radio link test)

Ltest is a utility for radio link testing. It is recommended for using in the antenna alignment process when installing a new radio link or for testing of the existing radio channel. It allows:

- Real-time monitoring of signal levels, retries and errors in the radio link, thus, making it possible to increase the quality of the link by antenna re-alignment, device re-location or upgrade of the wireless system
- Radio link throughput testing.

Syntax:

```
ltest IFNAME target [-r rate[,reply_rate]]
                    [-s packet_size[,reply_size]] -- default size 1024, max 1794
                    [-b]                          -- send as broadcast
                    [-p priority]                 -- set priority (0 to 16)
                    [-a (l|r|m)]                  -- enable audiomonitor,
                                                  l - local amp, r - remote amp
                                                  m - min(local,remote) amp
                    [-auto (l|r|m)]               -- like -a, but run in background
                                                  and autostart on boot
                    [-align [L[,R]]]              -- MIMO antenna alignment mode
                                                  L,R - local/remote tx antenna
                                                  (0/1 or V/H)
```

```
                 [-evm]                  -- display Error Vector Magnitude
                 [-tu [seconds]]         -- unidirectional throughput test
                 [-tb [seconds]]         -- bidirectional throughput test
                 [-load N[m|k]           -- limit throughput to N:
                                         m - Mbps, k - Kbps
                 [-mint]                -- do throughput test through MINT
ltest -key [PASSWORD]
ltest (-stop|-noauto)                   -- stop all running tests
                                        (-noauto - cancel test on boot)
ltest (-disable|-enable)               -- disable/enable ltest
```

Description:

- **IFNAME** – radio interface on which testing will be performed
- **target** – MAC-address of a target device on the other side of a tested radio link
- **-r rate[,reply_rate]** – sets bitrates for transmitting test packets from the local device and toward it. This parameter is optional. There are two situations when this parameters are not configured:
  - Local device is tested with its neighboring node, i.e. we can view remote device and **tx/rx bitrate** values for it in a «**mint map**» command output. In this case **tx/rx bitrate** values from «**mint map**» command output are taken for **rate** and **reply rate** parameters.
  - Local device doesn't consider remote device as a neighboring node. In this case **rate** and **reply rate** parameters will be equal to minimal possible **bitrate** of the local device for current bandwidth (for example, 13 Mbps for 20 MHz bandwidth, 6.5Mbps for 10 MHz, 3.25 Mbps for 5 MHz).
- **-s packet_size[,reply_size]** – establishes test packet size from the local device and toward it. Test packet size by default is 1024 bytes. Maximal possible test packet size is 1810 bytes.
- **-b** – transmitting broadcast test packets
- **-a (l|r|m)** – enabling sound indication mode (AudioMonitor).

  AudioMonitor is intended to be used while preliminary link establishing and antenna alignment procedure in order to obtain the maximum signal level from the remote side. When having AudioMonitor plugged into the console port of the device with headphones plugged in, the sound signals will be sent to the headphones. The AudioMonitor or the console cable must be plugged in when the device is powered off. No return packets state on the testing device input (no link) with described testing mode turned on is indicated by seldom tones in the headphones. If there is a link established, the signal level is indicated by the set of clicks. The higher the signal level, the higher is the frequency of the clicks and their number in the set. The durability of the set and the interval between the sets is always the same and has a value of 2 and 0.5 seconds correspondingly. The indication cycle always corresponds with a PREVIOUS measurement cycle. That means that the sound indication has a delay of 2.5 seconds. The maximum signal level is indicated by a permanent tone.

⚠ ***Attention! AudioMonitor is NOT included in a standard device set - it should be ordered separately.***

l – When this option is set you will hear a sound indication of a signal that is received on a local device, i.e. a signal which is transmitting from remote device to local device

r – When this option is set you will hear a sound indication of a signal that is received on a remote device i.e. a signal which is transmitting from local device to remote device

m – When this option is set you will hear a sound indication of a signal that is minimal among two previous signals (for example, if a signal that is received on a local device is lower than a signal that is received on a remote device then a signal that is received on a local device will be indicated)

- **-auto (l|r|m)** – automatic test in a sound indication mode that begins after device reboot. After device reboot link test will start in the sound indication mode(AudioMonitor) only, i.e. you will not be able to view test output on the screen. Parameters are the same as «–a(l|r|m)»
- **-key [PASSWORD]** – sets password for testing. If two devices have different passwords they can't perform testing with each other
- **-noauto** – disables automatic test after device reboot, i.e. test won't stop immediately, but after device reboot it won't start
- **-stop** - stops the test almost immediately
- **-disable|-enable** – disables/enables ability to perform link test. Enable by default
- **-align [L[,R]]** – special "ltest" command mode for antenna alignment for «Xm» series devices (for example, R5000-Om). It allows aligning each antenna of the device independently. *L* parameter sets which antenna will be used to transmit test frames from the local device. *R* parameter sets which antenna will be used to transmit test frames from the remote device from the other side of the link. If *L* parameter is not specified then average signal level value between two antennas will be shown. If *–r* is not specified then test can be performed even when only one antenna is attached to each side of the link. If *–r* is specified then *L* and *R* parameters are ignored.

  L and *R* parameters can have the following values: 0 – antenna with vertical polarization, 1 – antenna with horizontal polarization.
- **-evm** – indicates the measured input signal quality (Error Vector Magnitude). It should be as high as possible. The recommended level is not less than 21.

All parameters (including specified by default) keep its values the same as they were at the beginning during the whole test.

Examples:

```
lt rf5.0 00179AC2F3E6
```

This command illustrates the simplest way to start link test. «**lt**» command with all parameters undefined (default parameters) starts test of a local device with a remote device which have "00179AC2F3E6" MAC-address.

```
lt rf5.0 00179AC2F3E6 –r 24000
```

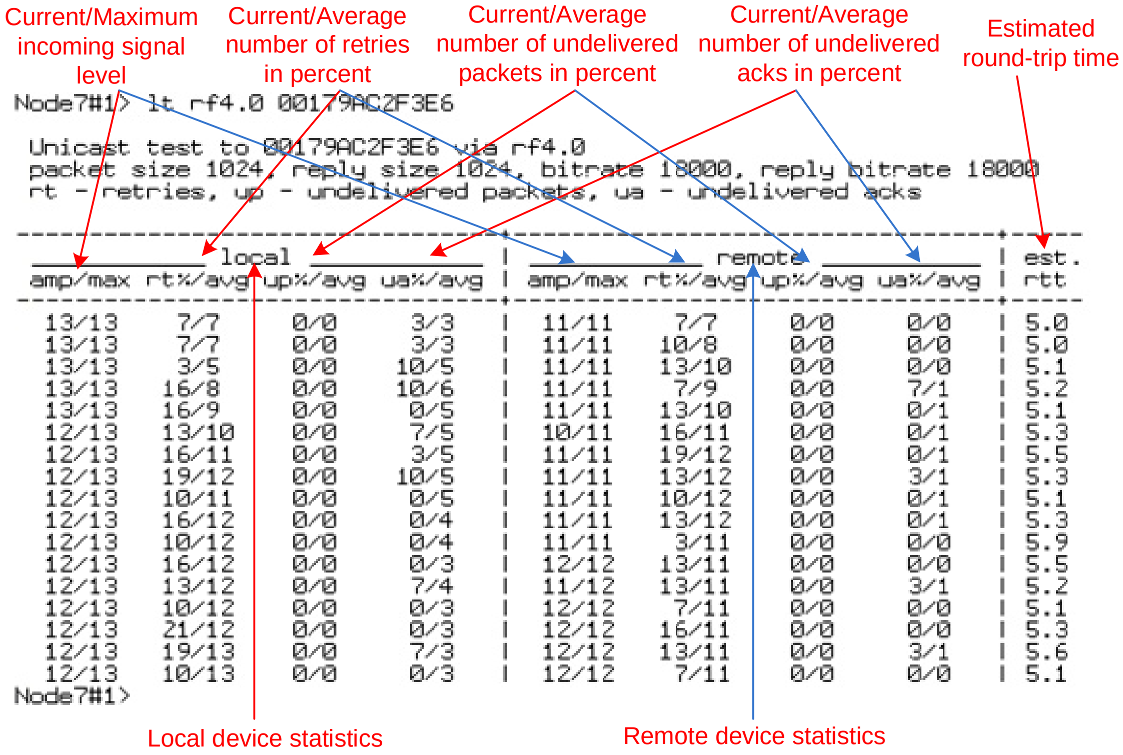
This command starts link test with **rate** parameter 39 Mbps. The **reply rate** parameter will be considered the same.

```
lt rf5.0 00179AC2F3E6 –a l
```

This command starts link test in a sound indication mode when sound signal represents a signal that is received on a local device.

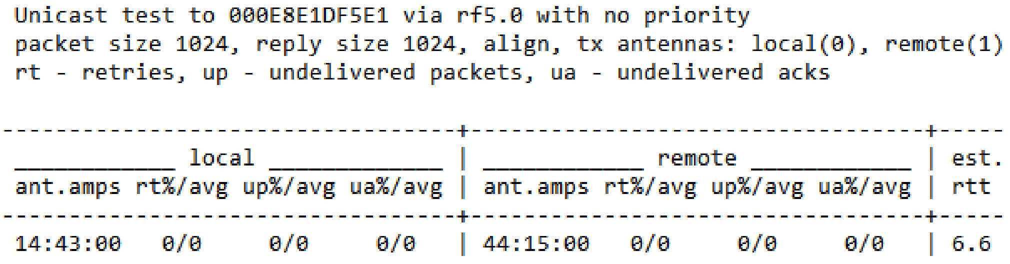
«ltest» command recommendations:

When «**ltest**» command starts it will show you output information that contains testing results (except **auto** mode). You can see command output below:


Current/Maximum incoming signal level
Current/Average number of retries in percent
Current/Average number of undelivered packets in percent
Current/Average number of undelivered acks in percent
Estimated round-trip time
Local device statistics
Remote device statistics

```
Node7#1> lt rf4.0 00179AC2F3E6

 Unicast test to 00179AC2F3E6 via rf4.0
 packet size 1024, reply size 1024, bitrate 18000, reply bitrate 18000
 rt - retries, up - undelivered packets, ua - undelivered acks

---------------------------------+--------------------------------+-----
 ____________ local _____________ | ____________ remote ____________ | est.
 amp/max rt%/avg up%/avg ua%/avg | amp/max rt%/avg up%/avg ua%/avg | rtt
---------------------------------+--------------------------------+-----
   13/13     7/7     0/0     3/3 |  11/11     7/7     0/0     0/0 | 5.0
   13/13     7/7     0/0     3/3 |  11/11    10/8     0/0     0/0 | 5.0
   13/13     3/5     0/0    10/5 |  11/11   13/10     0/0     0/0 | 5.1
   13/13    16/8     0/0    10/6 |  11/11     7/9     0/0     7/1 | 5.2
   13/13    16/9     0/0     0/5 |  11/11   13/10     0/0     0/1 | 5.1
   12/13   13/10     0/0     7/5 |  10/11   16/11     0/0     0/1 | 5.3
   12/13   16/11     0/0     3/5 |  11/11   19/12     0/0     0/1 | 5.5
   12/13   19/12     0/0    10/5 |  11/11   13/12     0/0     3/1 | 5.3
   12/13   10/11     0/0     0/5 |  11/11   10/12     0/0     0/1 | 5.1
   12/13   16/12     0/0     0/4 |  11/11   13/12     0/0     0/1 | 5.3
   12/13   10/12     0/0     0/4 |  11/11    3/11     0/0     0/0 | 5.9
   12/13   16/12     0/0     0/3 |  12/12   13/11     0/0     0/0 | 5.5
   12/13   13/12     0/0     7/4 |  11/12   13/11     0/0     3/1 | 5.2
   12/13   10/12     0/0     0/3 |  12/12    7/11     0/0     0/0 | 5.1
   12/13   21/12     0/0     0/3 |  12/12   16/11     0/0     0/0 | 5.3
   12/13   19/13     0/0     7/3 |  12/12   13/11     0/0     3/1 | 5.6
   12/13   10/13     0/0     0/3 |  12/12    7/11     0/0     0/0 | 5.1
Node7#1>
```

«**ltest**» output when using "*-align*" parameter:



```
Unicast test to 000E8E1DF5E1 via rf5.0 with no priority
packet size 1024, reply size 1024, align, tx antennas: local(0), remote(1)
rt - retries, up - undelivered packets, ua - undelivered acks

----------------------------------+----------------------------------+-----
____________ local _____________ | ____________ remote ____________ | est.
ant.amps rt%/avg up%/avg ua%/avg | ant.amps rt%/avg up%/avg ua%/avg | rtt
----------------------------------+----------------------------------+-----
14:43:00    0/0      0/0      0/0 | 44:15:00    0/0      0/0      0/0 | 6.6
```

The difference of this output from the standard one is that «ant.amps» column is used instead of «amp/max». «Ant.amps» column indicates signal levels from 0, 1 and 2 antennas divided by ":" correspondingly.

For successful radio link establishment the following factors have to be considered:

1. It is recommended to start antenna alignment with searching maximum signal level on a minimal possible bitrate. Afterwards automatic MINT mechanisms will set the most appropriate bitrate if **autobitrate** mode will be enabled.

2. Current incoming signal level in «amp/max» columns (see "ltest" command output) must be between 12 and 40.

When it is more than 40 it is recommended to lower amplifier power.

If maximal signal level is less than 12 it is recommended to lower bitrate or channel width (for example, from 20MHz to 10MHz on the both sides of the radio link).
In some cases signal level that is less than 12 may be enough for radio link operation. In this case one has to be guided by such parameters as number of retries, number of undelivered packets and number of undelivered acks. If the number of undelivered packets and the number of undelivered acks is null, the number of retries is small and all these parameters are constant in time then the radio link, most often, will be operating properly.

3. Number of retries value in «rt%» columns must be as close to zero as possible.

4. Number of undelivered packets value in «up%» columns must be zero; if this value is not zero then the radio link couldn't be exploit.

5. Number of undelivered acks value in «ua%» columns must be zero; if this value is not zero then the radio link couldn't be exploit. If this value is constantly not less then 50 then most probably «**distance**» parameter is set with a wrong value. If radio link distance is more than 20 km then «**long**» mode must be enabled.

ALL described parameters must be observed in the both (**Local** and **Remote**) sections of the «**ltest**» command output.

Radio link bandwidth test (Bandwidth meter):

Bandwidth meter is used to test the following radio link characteristics: speed in kilobits per second, speed in packets per second, number of retries and errors.

Use the following «*ltest*» command options for testing:

- **-tu [seconds]** – Unidirectional test: packets are transmitted only from the current side to the specified address ("*target*" option)
- **-tb [seconds]** – Bidirectional test: packets are transmitted in both directions

Packet size by default - 1536 bytes (to change packet size use «-s» option).

"*Seconds*" parameter allows setting test period (5 seconds by default). Maximum value is 60 seconds.

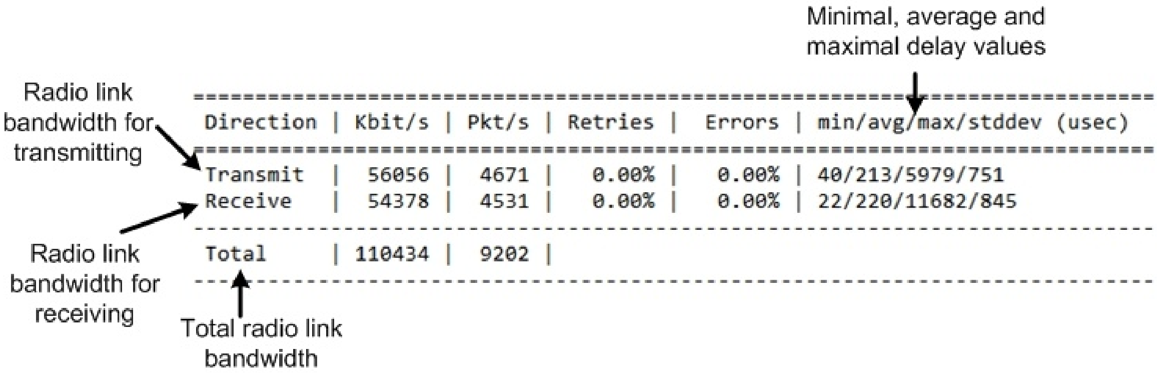
-load N[m|k] option allows setting a limit on the maximal tested channel bandwidth. By default, *N* parameter is measured in Megabits per second. If *k* parameter is specified then in kilobits per second (for example, 10m - 10 Mbps, 500k - 500 Kbps).

"**-mint**" option allows performing MINT-enabled tests when all the traffic and link parameters are controlled and managed by MINT functions such as ATPC and autobitrate. In this mode the statistics for errors abd retries is not available.

When using "**-mint**" option "*target"* parameter can point to any MINT node's MAC-address including nodes that are not direct neighbors of the current node (multi-hop).

"**-mint**" option is available only with *-tu* or *–tb* options. When using "**-mint**" option *-r* option is ignored.

“Ltest” command output in Bandwidth meter mode:



Minimal, average and maximal delay values
Radio link bandwidth for transmitting
Radio link bandwidth for receiving
Total radio link bandwidth

```
==============================================================================
 Direction | Kbit/s | Pkt/s | Retries |  Errors | min/avg/max/stddev (usec)
==============================================================================
 Transmit  |  56056 |  4671 |   0.00% |   0.00% | 40/213/5979/751
 Receive   |  54378 |  4531 |   0.00% |   0.00% | 22/220/11682/845
------------------------------------------------------------------------------
 Total     | 110434 |  9202 |
------------------------------------------------------------------------------
```

Example:

lt rf5.0 00179AC2F3E6 -tb

This command starts bidirectional link bandwidth test of a local device with a remote device which have “00179AC2F3E6” MAC-address.

6. Muffer command (Environment analyzer)

The muffer module is used to analyze the electromagnetic environment.

Syntax:

muffer IFNAME [-tXX] [-lXX] review [FREQ1 [FREQ2 ...]]

| sid | { mac[2|3]|mynet|scan [MAC]}

muffer IFNAME sensor [record=SEC] [F1 [F2] [BW STEP]]

muffer IFNAME sensor [replay]

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.

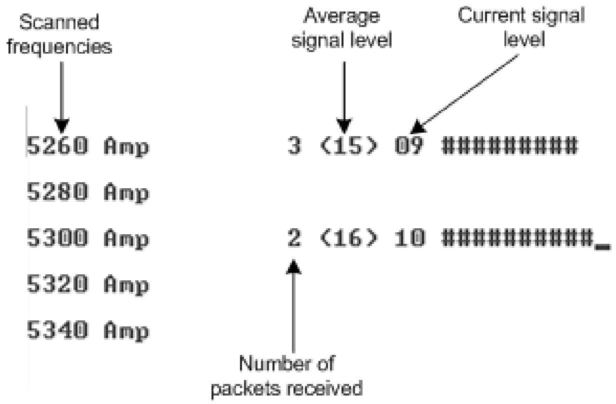
Several operating regimes of the **muffer** module provide for different levels of details in test results.

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. Several frequencies (separated by spaces) subject to analysis may be specified as parameters [FREQ1 [FREQ2 ...]]. Only two last digits of the frequency values shall be given.

Example:

muffer rf5.0 review



```
5260 Amp        3 (15) 09 #########
5280 Amp
5300 Amp        2 (16) 10 ##########_
5320 Amp
5340 Amp
```

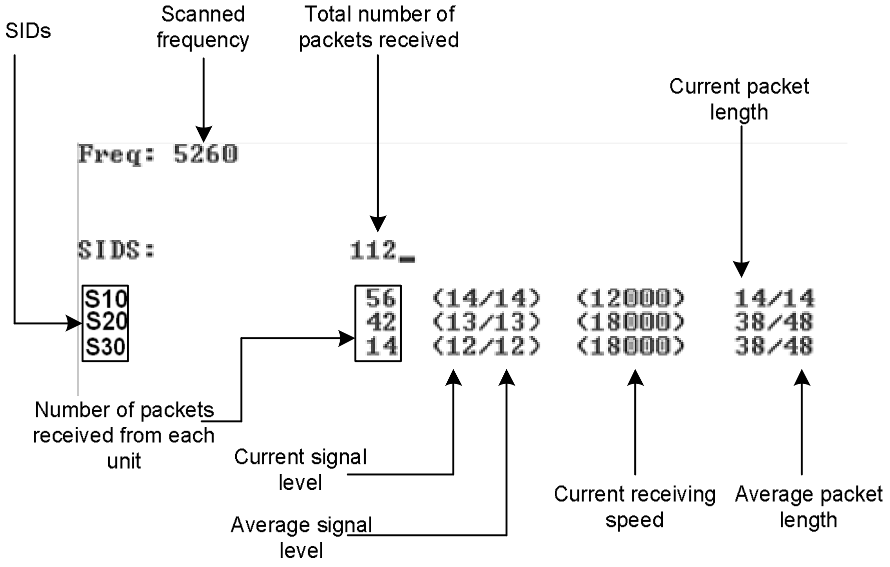
The picture above shows the output of review mode.

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.

Example:

muffer rf5.0 sid



```
Freq: 5260

SIDS:            112_

S10              56  (14/14)  (12000)  14/14
S20              42  (13/13)  (18000)  38/48
S30              14  (12/12)  (18000)  38/48
```

MAC|MAC2|MAC3|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. **[MAC]** option allows carrying the air link analysis in MAC|MAC2|MAC3|MYNET modes for the specified MAC-address.

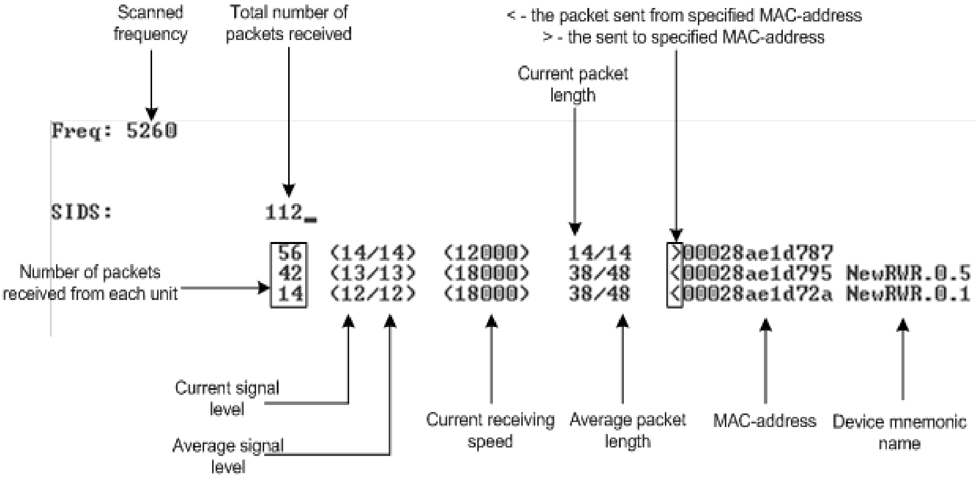
The **mac** mode checks only data packets, while in the **mac2** mode the link-level ACK messages sent by protocol support devices are also taken into account whenever possible. Compared to "mac2" mode, **mac3** mode also includes calculation of impulse noisy signals. It shows number of detected pulses, their average signal level and pulses per second information.

Finally, the **mynet** mode performs the radio testing taking into account only packets from within the given network.

Example:

muffer rf5.0 mac2

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



```
Freq: 5260

SIDS:          112_
               56  (14/14)  (12000)  14/14  >00028ae1d787
               42  (13/13)  (18000)  38/48  <00028ae1d795 NewRWR.0.5
               14  (12/12)  (18000)  38/48  <00028ae1d72a NewRWR.0.1
```

Scanned frequency
Total number of packets received
< - the packet sent from specified MAC-address
> - the sent to specified MAC-address
Current packet length
Number of packets received from each unit
Current signal level
Average signal level
Current receiving speed
Average packet length
MAC-address
Device mnemonic name

Scan mode

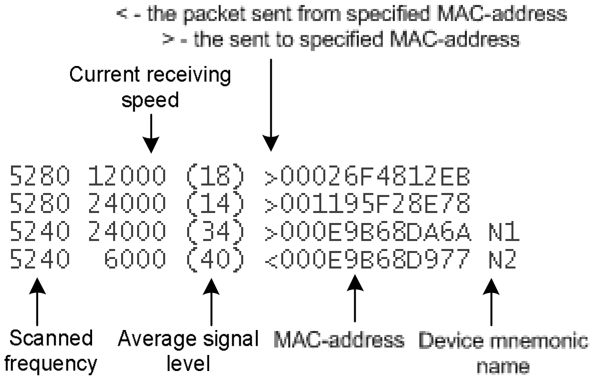
The scanning regime is enabled by a **muf scan** command, and provides for deep analysis of radio emission sources within the given network's territory. In this regime, the device scans the radio spectrum on all frequencies and for all modulation types.

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.

Example:

muffer rf5.0 scan



```
5280 12000 (18) >00026F4812EB
5280 24000 (14) >001195F28E78
5240 24000 (34) >000E9B68DA6A N1
5240  6000 (40) <000E9B68D977 N2
```

Supplementary options for all the above regimes:

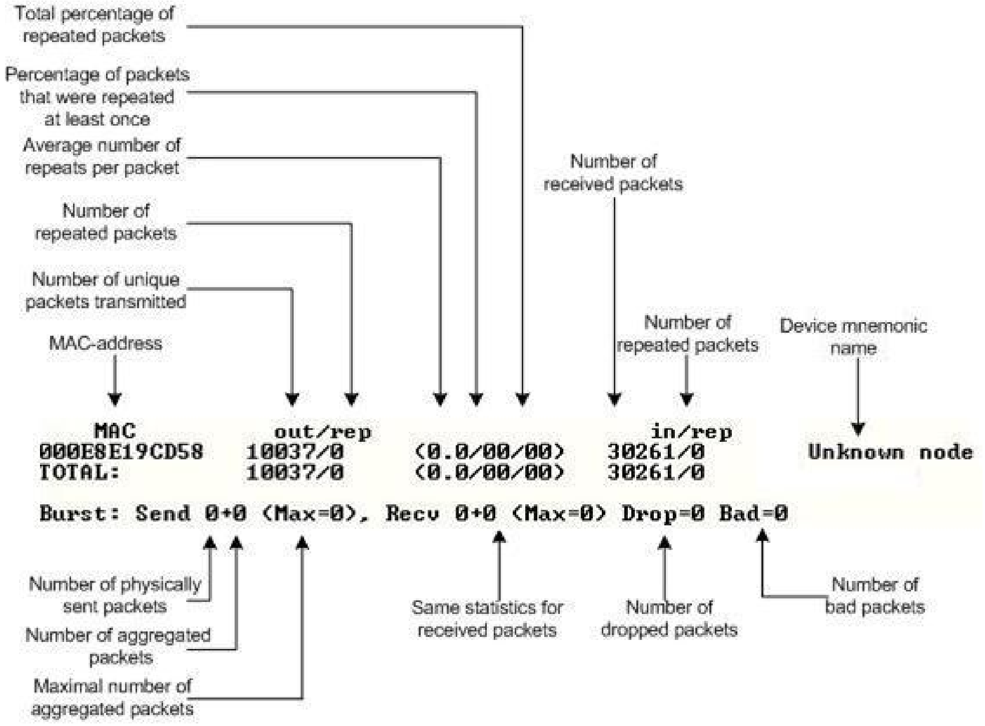
- **-tXX** specifies the duration of time, in seconds, for which the test regime is enabled (2 minutes by default). The value 0 in this field enables a test regime for illimited time.
- **-lXX** specifies the number of lines on the screen for displaying test results (24 lines by default)

For termination of 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.


Total percentage of repeated packets
Percentage of packets that were repeated at least once
Average number of repeats per packet
Number of repeated packets
Number of unique packets transmitted
MAC-address
Number of received packets
Number of repeated packets
Device mnemonic name
Number of physically sent packets
Number of aggregated packets
Maximal number of aggregated packets
Same statistics for received packets
Number of dropped packets
Number of bad packets

```
    MAC            out/rep                           in/rep
000E8E19CD58    10037/0       (0.0/00/00)    30261/0          Unknown node
TOTAL:          10037/0       (0.0/00/00)    30261/0

Burst: Send 0+0 (Max=0), Recv 0+0 (Max=0) Drop=0 Bad=0
```

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:

***muf**fer stat*

To reset all counters please type

***muf**fer stat clear*

Spectrum Analyzer mode

The Spectrum Analyzer mode is enabled by a **muf sensor** command and provides deep analysis of radio emission sources. In this mode the device scans the radio spectrum on all available frequencies.

Information is displayed on the screen in a visual-digital format.

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

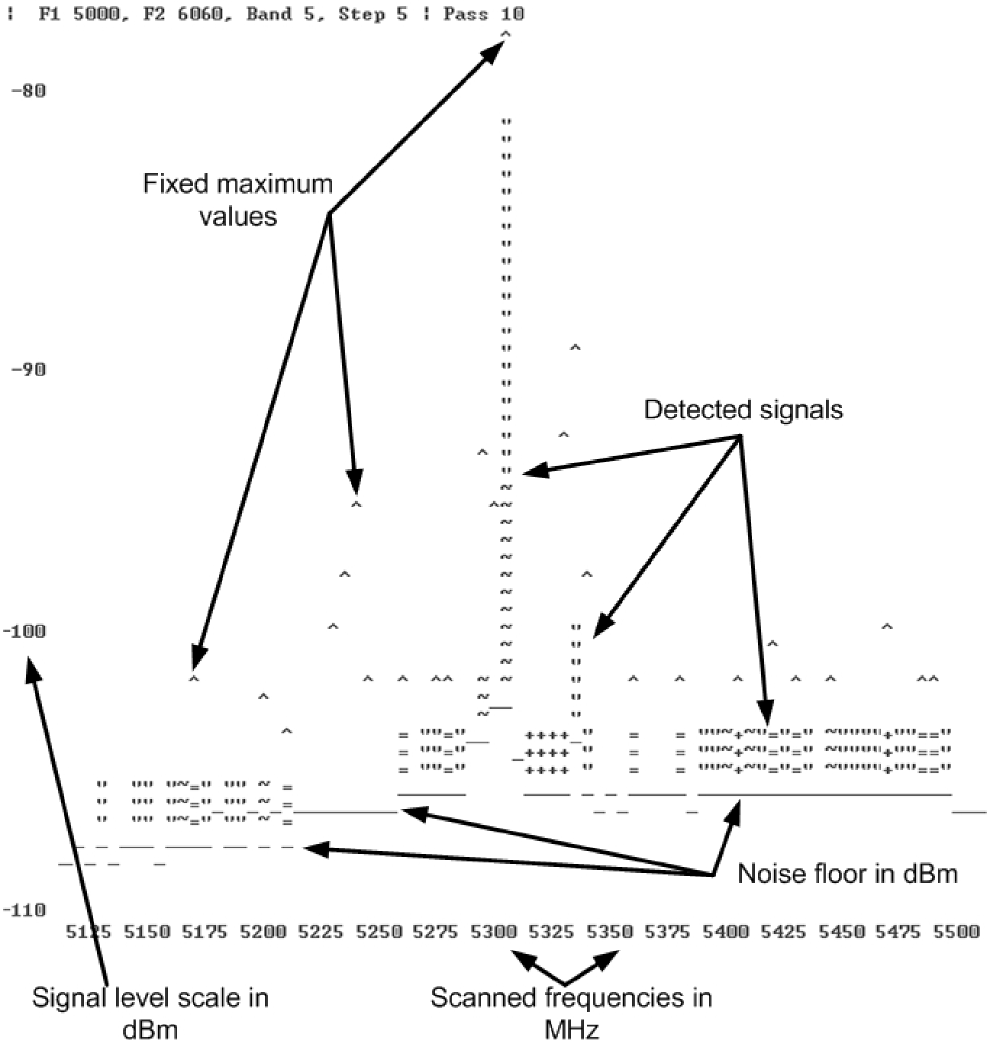
It is recommended to use Graphical Spectrum Analyzer in Web-interface (please see IW "Technical User Manual" for instructions).

Running Spectrum Analyzer mode disturbs normal operation of the radio module and makes it impossible to access the unit via radio.

Example:

```
muffer rf5.0 sensor
```

The picture below shows **muf sensor** output:



```
!  F1 5000, F2 6060, Band 5, Step 5 ! Pass 10
```

Supplementary options for "muf sensor" command:

- **F1 –** sets the initial frequency for scanning in MHz. Minimal available frequency for the given equipment model is used by default.

- **F2 –** sets the ending frequency for scanning in MHz. Maximal available frequency for the given equipment model is used by default. The actual shown ending frequency is limited by the size of the program window.
- **BW** – sets bandwidth in MHz. Allowed values are 1/5/10/20 MHz depending on the concrete equipment mode.
- **STEP** – sets frequency changing step in MHz. Allowed values are 1/5/10/20 MHz but no more than defined bandwidth value.
- **record=SEC** – starts Spectrum Analyzer in the background mode with the specified duration time in seconds. The result is stored in the memory till the unit's reset.
- **replay** – allows viewing the result of the last Spectrum Analyzer scan.

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

7. 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 apartment 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.

Quiet option allows switching off message logging to the system log.

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.

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.

8. 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

o **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
```

9. 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 radio interface with active protocol MINT. Be careful when entering this command via telnet.

arp [-]proxyall [$ACL]

In **proxyall** 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
```

10. Switch command

This command is used to configure MAC Switch.

Syntax:

```
________ LIST commands ___________________________________
 switch list LISTNAME [{iface | mac | numrange | match}]
        {add | del} [VALUE ...]
        dump [WILDCARD]
        rename  NEWNAME
        file   FILENAME
        [ flush|remove]
________ GROUP commands ____________________________________
 switch group ID {add | del} IFNAME[:{TAG|0}] ...
 switch group ID {repeater|trunk|uncoupled} {on|off}
 switch group GORUPID {(up|down)stream} {SCID|0}
 switch group ID [x]vlan {TAG|LIST|0} [[no]bidir]
 switch group ID nvlan {[on]|off}
 switch group ID info INFO_STRING
 switch group ID setid NEWID
 switch group ID stp { off | on | dump }
 switch group ID stp priority [PRIO]        #(default: 32768, step: 4096)
 switch group ID stp forwarddelay [DELAY]        #(default: 15 sec)
 switch group ID stp maxage [TIME]              #(default: 20 sec)
 switch group ID stp port IFNAME priority [PRIO] #(default: 128,step 16)
 switch group ID stp port IFNAME cost [COST] #(default: 200000(RSTP),
                                                   65535(STP))
 switch group ID igmp-snooping { off | on }
 switch group ID order N
 switch group ID
      [ setpri|addpri prio ]
      {deny | permit | showrules | showblack}
 switch group ID
        [dump [interface] [WILDCARD]]
        [dbdelete   MACADDRESS]
        {start | stop | remove}
 switch group ID in-trunk [{ID|0}]
 switch admin-group {ID|0}
________ RULES commands ____________________________________
 switch {group ID | interface IFNAME} rule NUMBER
        [set NEWNUMBER]
        [src   LIST] [dst   LIST] [vlan  LIST]
        [iface LIST] [proto LIST] [match LIST]
        [ setpri|addpri prio ]
        [ deny | permit ]   [ remove ]
_______  CONTROL commands _________________________________
 switch keeptag [(on|off)]
 switch resynchronize
 switch local-tag TAG
 switch trace { off | on | verbose | filter "pcap expr"}
 switch stptrace { off | on }
 switch stpblock { off | on }
 switch {dump [WILDCARD]|MACADDRESS}


 switch igmp-snooping dump [detail]
 switch igmp-snooping lmqt Value
 switch igmp-snooping gmi Value
 switch igmp-snooping static-add MCAST IF_NAME [MAC]
 switch igmp-snooping static-del MCAST IF_NAME [MAC]
 switch igmp-snooping srcip IP
 switch igmp-snooping querier group N [source X] [mcast X [,Y,...]] [vlan N]
{start|stop|clear}


 switch {start | stop | restart | destroy | dead-interval DEAD_INTERVAL |strict-
 admin [(on|off)]}
 switch statistics [(clear|help)]
 switch maxsources (MAXSOURCES|0)
```

ATTENTION!

Starting from 1.22.0 firmware version, "switch" is partially incompatible with other firmware versions. It is highly recommended to perform firmware upgrade for units working in switch mode. Compatibility for MINT protocol and routing is not disturbed.

"Over The Air Firmware Upgrade" feature also can be used.

Wildcard format

Wildcards are used in different commands to filter printed information. As a difference from standard wildcards, in special cases the following characters can be used:

- * - any number of any symbols (or empty).
- ~ - any symbol (just one).

Example:

```
rf~.~
```

This filter includes the strings like rf5.0, rf5.0 etc.

```
#1> switch group 1 dump eth~

Bridge group 1(normal), READY STARTED Interfaces : eth0(F) eth1(F) rf5.0(F)
Total records 5
  DST MAC        L   Int.   GateWay MAC   GT Cost   UsCNT    Dead    HashC
========= = ====  ========= == === ===== ===== ====
=======
001111144693      eth0   000000000000       0      3987       300       1
000435018822    * eth0  000000000000       0       0           0       1
000435118822    * eth1  000000000000       0       0           0       1
```

This filter displays group statistics for all Ethernet interfaces.

List configuration commands

Syntax:

```
switch list LISTNAME [{iface | mac | numrange | match}]
        {add | del} [VALUE ...]
        dump [WILDCARD]
        rename  NEWNAME
        file   FILENAME
        [ flush|remove]
```

Lists are used as a set of acceptable values for **rules**. Each list must have a unique name and must be of one of the types: iface, mac, numrange, match. List name may consist of letters and digits. List name should not start with a digit. List name is case-insensitive.

Command parameters:

- **LISTNAME** – list name. If list name contains spaces, it should be put in quotes.
- **iface** – list type which consists of network interfaces names.
- **mac** – list type which consists of a set of MAC-addresses
- **numrange** – list type that consists of a set of ranges of positive integer numbers. The range of numbers is specified as **<min>[-<max>]**. The range may consist of one number if <min>=<max>. If a range of numbers is added to existing list and two ranges values intersect, these ranges will be concatenated.
- **match** – by context, *match* expressions are identical to expressions lists but should consist of one element – the expression itself. The expression should be written in PCAP format (see **tcpdump** utility). If an expression has spaces it should be put into quotes.

Keywords **add** and **del** add or delete values to the specified list correspondingly.

VALUE – one or several (except for **match**) values to be added or deleted from the list.

Examples:

```
switch list my_iface iface add eth0 rf5.0
```

Here a list of **iface** type is created with a name of **my_iface.** Interfaces eth0 and rf5.0 are added to this list.

```
switch list vlans numrange add 10 20-30 40
```

A range of numeric values are added to a list with a name of **vlans** and with a type of **numrange.** Values added are 10, the range from 20 to 30 and a value 40.

```
switch list ip_mynet match add ‘net 195.38.45.64/26’
```

A list-expression of **match** type is created. In this case when using filter its effect will cover all types of packets (ip, arp and т.д.) from 195.38.45.64/26 network.

```
switch list ip_mynet match add ‘ip net 195.38.45.64/26’
```

In this example a list-expression of **match** type is also created but now only ip packets from 195.38.45.64/26 network will be affected when using filter.

A source file can be specified for the list. The source file should contain the list of values with each value taking one line. The file is retrieved using FTP protocol.

Example:

```
switch list MACGROUP1 file ftp://1.2.3.4/switches/list/macgroup1.txt
```

With this macgroup1.txt file might contain the following information:

```
#The list of computers in HR department
00:01:02:03:04:05       # Smith
00:11:12:13:14:15       # Johnson
<EOF>
```

Values are loaded from the file automatically after switch is started, or when a source file name is modified or when the following command is executed:

```
switch synchronize
```

```
switch list LISTNAME remove
```

This command deletes the list with LISTNAME name from the switch configuration.

```
switch list LISTNAME flush
```

Clears the contents of LISTNAME name.

```
switch list OLDLISTNAME rename NEWLISTNAME
```

Renames the list with OLDLISTNAME to NEWLISTNAME.

```
switch list LISTNAME  dump [WILDCARD]
```

Prints the contents of the list LISTNAME. If WILDCARD parameter is specified, the command prints only those values from the list which satisfy the WILDCARD.

Groups configuration commands

Syntax:

```
switch group ID {add | del} IFNAME[:{TAG|0}] ...
switch group ID {repeater|trunk|uncoupled} {on|off}
switch group GORUPID {(up|down)stream} {SCID|0}
switch group ID [x]vlan {TAG|LIST|0} [[no]bidir]
switch group ID nvlan {[on]|off}
switch group ID info INFO_STRING
switch group ID setid NEWID
switch group ID stp { off | on | dump }
switch group ID stp priority [PRIO]        #(default: 32768, step: 4096)
switch group ID stp forwarddelay [DELAY]        #(default: 15 sec)
switch group ID stp maxage [TIME]              #(default: 20 sec)
switch group ID stp port IFNAME priority [PRIO] #(default: 128,step 16)
switch group ID stp port IFNAME cost [COST] #(default: 200000(RSTP),
                                                  65535(STP))
switch group ID igmp-snooping { off | on }
switch group ID order N
switch group ID
    [ setpri|addpri prio ]
    {deny | permit | showrules | showblack}
switch group ID
     [dump [interface]] [WILDCARD]]
     [dbdelete   MACADDRESS]
     {start | stop | remove}
switch group ID in-trunk [{ID|0}]
switch admin-group {ID|0}
```

```
switch group ID {add | del} IFNAME[:{TAG|0}] ...
```

The command adds or deletes specified interfaces to/from the switching group.

- **ID** – numeric switching group identifier (1-4095)
- **add**|**del** – these commands add/delete specified interfaces to/from the switching group. If “add” keyword is used and there is no switching group with ID identifier, it will be automatically created.
- **IFNAME –** network interface name which should be added or deleted from the switching group.
- **TAG.** This option allows different manipulations with VLAN tags of the packet when the packet is sent through this interface. The following options are available:
    - **TAG** is specified for the interfaces and its value is >0. That means that any packet forwarded to the interface by the switch will be tagged with a VLAN tag **TAG.** If the packet already had a tag, this tag will be retagged to **TAG.**
    - **TAG** is not specified. This means that the packet stays unmodified.
    - **TAG** is specified and its value is zero. This means that the packet sent through this interface will be untagged if it was previously tagged or sent without any changes if it was not tagged.

Example:

```
switch group 3 add rf5.0:10 eth0:0
```

In this example, all packets switched by group 3 will be tagged with VLAN TAG 10 when sending through rf5.0 interface and will be untagged when sent through eth0 interface.

☝ All packets destined for the switch are always untagged.

switch group ID {repeater|trunk|uncoupled} {on|off}

This command turns on/off the **repeater**, **trunk** or **uncoupled** modes**.**

In the **repeater** mode the group switches the packets simply by sending them to all the device's interfaces except the one the packet was received from.

In the **trunk** mode, the group switches all the packets received through eth* interfaces in such a way that when packets are sent to rf* interfaces, these packets are places in a group with a number corresponding to the packet's VLAN TAG. When receiving the packet from rf* interfaces, trunk group sends these packets to eth* interface tagging them with a switch group number this packet was received from.

If a Ring/redundant network is connected to a CORE network in multiple points, STP loops can be formed in the CORE network. Thus, STP-enabled switches may block some of the links. Switching groups with "**uncoupled on**" parameter blocks the traffic between each other even if they have the same switching group number. This does not affect the traffic to come into the wireless network. For the incoming traffic intermediate nodes only use the closest uncoupled node. This improves the effectiveness of network utilization.

Example:

switch group 12 trunk on

If trunk group which will provide transmission of multiple VLAN flows in different directions is enabled on device then **in-trunk** option should be used on a subscriber station for exact instruction of what trunk group is the group:

switch group ID in-trunk [{ID|0}]

For example, if a Group №100 on a subscriber station is a member of a trunk Group №5 (Group №100 was formed as a result of conversion of VLAN ID №100 into the Group №100), subscriber station switch configuration should have the following command: ***switch group 100 in-trunk 5***

This option allows creating multiple disjoined trunk groups in the same network with the same VLAN flows inside.

switch group ID vlan {TAG|LIST|0} [[no]bidir]

This command defines that the group will switch the packets which are tagged with **TAG** VLAN tag or with VLAN tags specified is a LIST of **numrange** type. In order to cancel this VLAN filtration, TAG should be specified as zero.

The **bidir** option enables two-way traffic classification by VLAN ID (from and into the wired segment). The option can be useful for a Ring (or redundant) topology network transmitting multiple VLANs when the traffic with certain VLAN IDs is picked up at junction points.

Important. When enabling this VLAN tag filter other rules (see below) do not work.

Example:

```
switch group 5 vlan 5
```

switch group ID xvlan {TAG|LIST|0} [[no]bidir]

This command unlike the "vlan {TAG|LIST|0}" rule allows groups to handle also not tagged packets.

Examples:

```
switch list MYNET numrange add 100 200 300
switch group 10 xvlan MYNET
 switch group 10 trunk on
```

Group №10 would handle packets tagged with VLAN IDs 100, 200, 300 as well as not tagged packets. Not tagged packets will be sent to MINT network with its own group number (in this case 10), tagged packets – with group numbers concurred with VLAN ID.

```
switch list MYNET numrange add 100 200 300
switch group 20 vlan MYNET
switch group 20 trunk on
```

Group №20 handles only tagged packets from the MYNET list and transmits them upgrading VLAN ID number to appropriate group (and vice versa).

```
switch list MYNET numrange add 100 200 300
switch group 30 vlan MYNET
switch group 30 trunk off
```

Group №30 handles only tagged packages from the MYNET list and transmits them without changing with the group number 30.

switch group ID nvlan {[on]|off}

This command defines that group will switch only the packets not tagged with VLAN tag.

switch group ID info INFO_STRING

This command allows adding comments to switch group description.

switch group ID setid NEWID

This command changes ID of the switching group to NEWID.

Example:

```
switch group 3 setid 7
```

switch group ID

```
[dump [interface] [WILDCARD]]
[dbdelete   MACADDRESS]
{start | stop | remove}
```

Here:

- **dump** – prints the database of all known MAC-addresses
- **interface** – prints the database of all known MAC-addresses by grouping them according to interfaces
- **WILDCARD** – the output will be filtered according to the WILDCARD criteria.
- **dbdelete MACADDRESS** – deletes all records from MAC-address database connected with a specified MACADDRESS
- **start|stop** – starts/stops a specified switching group.
- **restart –** restarts the switching group (same as "switch group ID start; switch group ID start" set of commands). The command is used to clean the switching group database.
- **remove** – deletes a specified switching group from the switch configuration.

Examples:

```
switch group 3 dump eth0
switch group 5 start
```

In order to access PCs which are connected wirelessly from eth* interfaces (e.g. workstations which are connected using wired interfaces to one of the units) on such units (border units of the wireless network) one of the groups should be selected as **admin group**. All packets destined for any of the switches in wireless network will be sent by this group.

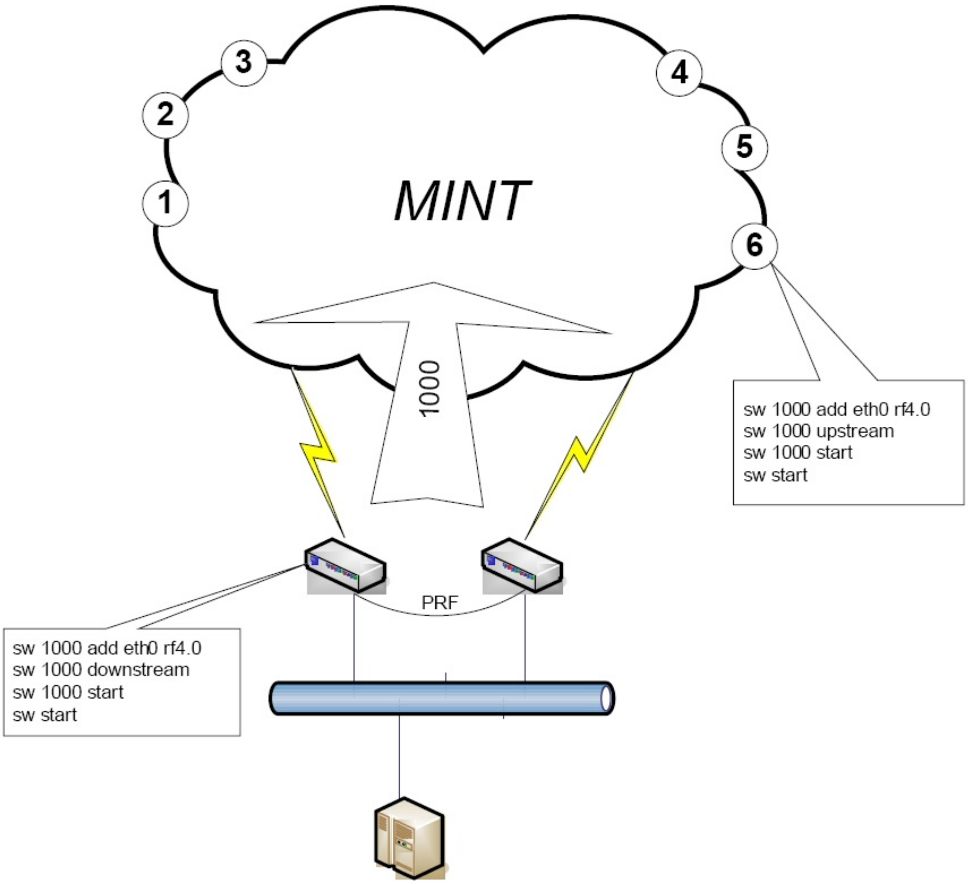
```
switch admin-group {ID|0}
```

In order to deal with upstream multicast flows in video surveillance systems two additional parameters are introduced - *upstream* and *downstream*.

```
switch group ID {(up|down)stream} {SCID|0}
```

Example:

For example, we have nodes with numbers 1, 2, 3, 4, 5 and 6 that are connected to digital cameras which broadcast video traffic using multicast packets. All of these flows need to be transferred to a video server the best way without flooding the network with unnecessary broadcast packets.



sw 1000 add eth0 rf4.0
sw 1000 upstream
sw 1000 start
sw start

sw 1000 add eth0 rf4.0
sw 1000 downstream
sw 1000 start
sw start

Entire downstream (from server to camera) traffic, if any, is transferred in group number 1000 in which all the nodes are located. But upstream flows from each camera are transmitted directly to the nearest hub of the group.

A feature of this solution is the ability to set multiple concentrators with the same number of the group. To address the problem broadcast storm that could arise from the fact that the concentrators are included in the various ports switch of one wire in MINT restricted - Broadcast and downstream concentrators never use each other to carry traffic. Furthermore, the availability of options "upstream" ensures that the terminal nodes will choose to send packages only one hub, but it is the shortest way to the nearest hub.

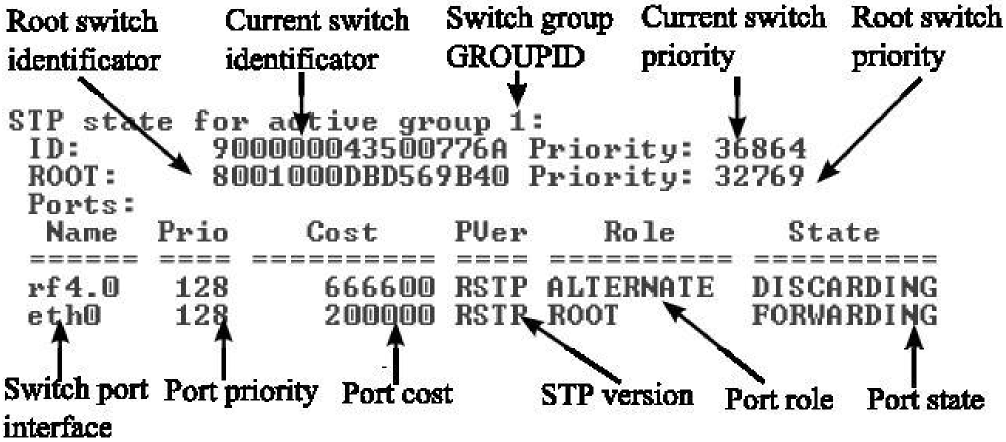
A distinctive feature of such solution is a possibility to use different hubs with the same group number. To eliminate broadcast storm that could have happened because of the connection of several hubs to different ports of the same switch – trunk and downstream hubs never use each other for traffic transmission. Moreover, *upstream* option guarantees that nodes will choose only one hub for packet delivery (the shortest route to the nearest hub).

MAC Switch supports **STP protocol**, namely two its versions: STP and RSTP. To implement this feature the following switch commands are introduced:

switch group ID stp { off | on | dump }

This command with *off/on* options enables or disables STP for the group. *Dump* option allows to see STP state of the group.

"switch group ID stp dump" command output:



```
STP state for active group 1:
 ID:        900000043500776A Priority: 36864
 ROOT:      8001000DBD569B40 Priority: 32769
 Ports:
  Name   Prio     Cost    PVer     Role       State
  ====== ==== ========= ==== ========= =========
  rf4.0   128    666600 RSTP ALTERNATE  DISCARDING
  eth0    128    200000 RSTP ROOT       FORWARDING
```

---

switch group ID stp priority [PRIO]

This command sets STP priority of a switch, where *[PRIO]* – priority value. If priority is not specified then default value 57344 is set. When setting priority value one should take into consideration that it will be automatically rounded down to a value divisible by 4096 (*step 4096*).

---

switch group ID stp forwarddelay [DELAY]

This command sets STP parameter *«forward delay»* which determines a time that switch spend in “listening” and “learning” states, where *[DELAY]* – time value in seconds. If not specified default value is set that is equal to 15 seconds.

---

switch group ID stp maxage [TIME]

This command sets STP parameter *«MAX age»* which determines time for switch to deliver BPDU-packet, where *[TIME]* – value of this parameter in seconds. If not specified default value is set that is equal to 20 seconds.

---

switch group ID stp port IFNAME priority [PRIO]

This command sets STP switch, where *IFNAME* – port interface name, *[PRIO]* – port priority value. If not specified default value is set that is equal to 128. When setting priority value one should take into consideration that it will be automatically rounded down to a value divisible by 16 (*step 16*).

---

switch group ID stp port IFNAME cost [COST]

This command sets STP parameter *«cost»* of a switch port which determines switch port cost, where *[COST]* – value oh this parameter. If not specified default value is set that is equal to 200000 for RSTP, 65535 for STP.

Example:

---

switch group 1 add eth0 rf5.0

switch group 1 stp priority 36864

switch group 1 stp on

switch group 1 start

In this example switch group «group 1» is configured. STP protocol support is enabled and STP switch priority is set to 36864 for this group.

switch group ID igmp-snooping { off | on }

This command disables/enables "IGMP snooping" function for the switching group.

Example:

switch group 1 igmp-snooping on

switch group ID order N

The logic of assigning switch groups to packets is the following:

- Groups are run over in the order of their appearance in a configuration.
- The first group that is suitable for a packet is chosen and the process is stopped.

The command sets the order in which the concrete group will be run over during the assigning process.

switch group ID [setpri|addpri prio]

This command allows setting/increasing the priority of packets passing through the group. "**Setpri**" parameter is used to change a priority to the value specified in the command. When using "-1" value a package priority is dropped to the lowest priority. "**Addpri**" is used to change a priority only in case it is higher than the previous one (Note: the smaller is the value the higher is the priority). So you can only increase priority using "**addpri**" parameter.

Example:

switch group 1 addpri 15

switch group ID {deny | permit}

This command permits or denies processing and sending out the packets which belong to this group.

switch group ID {showrules}

This command displays detailed information about the group's classification rules, including the hits counter for each rule.

switch group ID {showblack}

This command displays the list of MAC addresses that are blocked due to the indeterminacy of their owner.

Rules configuration commands

Rules are used for the following purposes:

- Selecting an appropriate switching group when packet is received through eth* interface. Packet will be switched only by that group to which rules it fully satisfies.
- When packet is chosen by the switching group and group decides whether this packet needs to be sent through one of the interfaces. The packet will only be sent if it satisfies the rules of this interface.

The rules consist of rules list and a decision by default (deny/permit). Each rule consists of a sequential number, condition and decision (deny/permit). While going through the list, the switch checks whether a packet matches the rule. If it matches the rule, the decision set for this rule is applied to the packet. Otherwise, the list of rules is viewed further. Rules are taken according to their sequential number in ascending manner. If a packet does not match to any rule, the default decision for this group or interface is taken.

The condition might consist of one or several parameters which are checked with the packet. Five packet parameters can be checked:

1. Source interface (iface)
2. Source MAC-address (src)
3. Destination MAC-address (dst)
4. VLAN tag (vlan)
5. Ethernet-level protocol number (proto)

For each parameter a corresponding **list** of values should be specified. Moreover, in the condition a PCAP expression may be present. This expression will be considered as a “pseudo parameter” of the packet and is called **match.** Therefore, the packet is considered to have matched the condition, if all its parameters match to the corresponding acceptable values from the lists and/or the packet satisfies to the expression of **match** type. One or more parameters might be missing in a condition clause – in this case it will mean that packet satisfies to that part of the condition which is missing. If the list of acceptable values is empty, non of the values of the corresponding parameter can match the condition even if this parameter is missing in the packet (for example, VLAN tag).

Rules configuration is implemented using the following command:

```
switch {group ID | interface IFNAME} rule NUMBER
        [set NEWNUMBER]
        [src   LIST] [dst   LIST] [vlan  LIST]
        [iface LIST] [proto LIST] [LIST]
        [ setpri|addpri prio ]
        [ deny | permit ]   [ remove ]
```

Here:

- *ID* and *IFNAME* – number of the group or interface.
- *NUMBER* – sequential rule number
- *set NEWNUMBER* – changes the number of the rule to NEWNUMBER
- *remove* – deletes the rule
- *deny | permit* – sets the decision for the corresponding rule
- *src, dst, vlan, iface, proto, match* – commands for specifying the **lists** of acceptable values for the corresponding parameter of the packet.

- *setpri|addpri prio* - command allows setting/increasing the priority of packets passing through the group. "**Setpri**" parameter is used to change a priority to the value specified in the command. When using "-1" value a package priority is dropped to the lowest priority. "**Addpri**" is used to change a priority only in case it is higher than the previous one (Note: the smaller is the value the higher is the priority). So you can only increase priority using "**addpri**" parameter.

Example:

```
switch list MACGROUP1 MACGROUP1 mac add 00:01:02:03:04:05 00:11:12:13:14:15
switch list VGROUP numrange add 10 20-30 40
switch list IP_NET3845 match add 'arp net 195.38.45.64/26 || ip net 195.38.45.64/26'
switch group 5 rule 10  src MACGROUP1 vlan VGROUP match IP_NET3845 deny
switch group 5 rule 20   dst MACGROUP1 vlan VGROUP match IP_NET3845 deny
switch group 5 permit
switch group 1 rule 1 setpri 10
```

In order to configure a default decision for group/interface the following command should be used:

```
switch {group ID | interface IFNAME}
        {deny | permit }
```

Control commands

Syntax:

```
switch keeptag [(on|off)]
switch resynchronize
switch trace { off | on | verbose | filter "pcap expr"}
switch stptrace { off | on }
switch stpblock { off | on }
switch {dump [WILDCARD]|MACADDRESS}
switch local-tag TAG

switch igmp-snooping dump [detail]
switch igmp-snooping lmqt Value
switch igmp-snooping gmi Value
switch igmp-snooping static-add MCAST IF_NAME [MAC]
switch igmp-snooping static-del MCAST IF_NAME [MAC]
switch igmp-snooping srcip IP
switch igmp-snooping querier group N [source X] [mcast X [,Y,...]] [vlan N] {start|stop|clear}
switch {start | stop | restart | destroy | dead-interval DEAD_INTERVAL | strict-admin [(on|off)]}
switch statistics [(clear|help)]
switch maxsources (MAXSOURCES|0)
```

```
switch resynchronize
```

Forces to reload lists which had an external file as a source

```
switch trace { off | on | verbose | filter "pcap expr"}
```

Disables (off) / enables (on) logging the service information into the system log.

Verbose option enables more detailed information to be written into the system log.

Filter "pcap expr" option enables tracing how packets of the given type are being processed by the switch.

Example:

```
sw trace filter "ether host 00:11:22:33:44:55"
sw trace filter "net 1.2.3.0/24"
sw trace off - disables the filter action
```

```
switch stptrace { off | on }
```

Disables (off) / enables (on) logging of the STP service information into the system log.

```
switch stpblock { off | on }
```

“Stpblock on” command prevents STP frames forwarding in the switch mode when STP support is disabled on the unit. To allow STP frames forwarding use “stpblock off” command.

Switch MAC-address database is a routing table of MAC-layer which contains information on how the packet should be delivered to its destination (dst). Each switching group has an independent database. Records in the database are formed automatically based on the source address of the packet which was received by one of the interfaces included into a switching group. Moreover, the database always contains records corresponding with interfaces included into the switching group. These records are called local records. Each records has its life span. If, during this life span, none of the interfaces have received a packet with a source address from this record, this record is deleted from the database. By default, life span is five minutes. To change this parameter, the following command can be used:

```
switch dead-interval <DEAD_INTERVAL_IN_SECONDS>
```

To start/stop/restart the switch, the following command can be used:

```
switch {start | stop | restart}
```

To clear the switch configuration please use the following command:

switch {destroy}

To view the switch statistics please use the following command:

switch statistics [(clear|help)]

The statistics shows the information on forwarded/flooded/dropped packets and records of the switch MAC-address table.

The "clear" option clears the switch statistic. The "help" option shows a list of the descriptions of the drooped packets reasons used in the switch statistics command output.

The following command allows setting the maximum allowed number of records in the switch MAC-address table:

switch maxsources (MAXSOURCES|0)

The default number of records is 5000. When the value "0" is used the number of records is set to minimum possible of 500.

Local packets generated by the device itself do not belong to any switch group. Therefore, by default, they are sent into the wired interface (eth*) untagged. The following command allows assigning the local packets with a VLAN tag:

switch local-tag TAG

The packets not belonging to any switch group will be sent into the wired interface with the VLAN value equal to the value of the "local-tag" parameter. And vice-a-versa, packets received via the wired interface and having their VLAN value equal to the "local-tag" parameter will be processed as local ones.

The "local-tag" parameter is also called a Management VLAN as it makes the device access and configuration possible only via the defined VLAN.

switch keeptag [(on|off)]

This command ("switch keeptag on") allows keeping the tag for the packets destined to the device itself and received via the wired interface (eth*). "Off" by default.

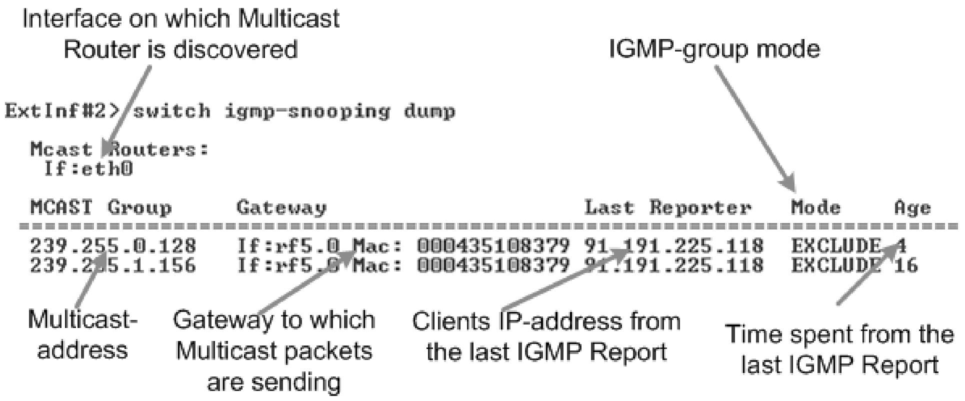
switch strict-admin [(on|off)]

By default, the device in switch mode process the packets that is destined to it itself regardless of the switch group they belong to (switch strict-admin off). However, by using "switch strict-admin on" command one can force the device to accept such packets when they come in the admin-group only.

switch igmp-snooping dump [detail]

This command allows to see a list of IGMP hosts which are subscribed on Internet Protocol multicast group.

«Switch igmp-snooping dump» command output:



```
ExtInf#2> switch igmp-snooping dump
  Mcast Routers:
   If:eth0
  MCAST Group      Gateway                         Last Reporter    Mode     Age
 ==========================================================================
  239.255.0.128    If:rf5.0 Mac: 000435108379 91.191.225.118   EXCLUDE 4
  239.255.1.156    If:rf5.0 Mac: 000435108379 91.191.225.118   EXCLUDE 16
```

Interface on which Multicast Router is discovered
IGMP-group mode
Multicast-address
Gateway to which Multicast packets are sending
Clients IP-address from the last IGMP Report
Time spent from the last IGMP Report

Parameter "**detail"** allows seeing detailed information on Multicast-subscribers.

switch igmp-snooping lmqt Value

This command sets «Last Member Query Time» value, i.e. the maximum time during which the switch will wait for the answer from active subscribers after receiving "IGMP leave". If no answer is received the switch will stop Multicast packets delivery to the particular Gateway. Gateway is an Ethernet interface or radio interface with a MAC-address of the device on the other side of the link.

switch igmp-snooping gmi Value

This command sets «Group Membership Interval» value, i.e. the amount of time that must pass before a Multicast Router decides there are no more clients subscribed to a Multicast group (no more "IGMP report" messages in the group).

switch igmp-snooping static-add MCAST IF_NAME [MAC]

This command creates static subscription on a Multicast-address.

switch igmp-snooping static-del MCAST IF_NAME [MAC]

This command removes static subscription on a Multicast-address.

switch igmp-snooping srcip IP

This command allows replacing a source IP-address in IGMP Report packets with the one specified in the "IP" parameter.

switch igmp-snooping querier group N [source X] [mcast X [,Y,...]] [vlan N] {start|stop|clear}

This command starts/stops (**start/stop**) «Querier» function operation. «IGMP Querier» substitutes the functions of Multicast Router when organizing video systems using «IGMP Snooping» services.

IGMP Querier parameters:

- **group N** – defines a switching group that uses «IGMP Snooping» services
- **source X** – sets source IP-address for Multicast packets

- **mcast X[,Y,...]** – sets concrete Multicast Group (or a number of groups) to be allowed for subscription
- **vlan N** – enables transmission of Multicast packets using Vlan.

The “**clear**” option deletes IGMP Querier configuration.

Sample configuration

```
switch list VGROUP numrange add 10 20-30 40
switch list ALL_VLAN numrange add 0-4095

switch group 5 add eth0 rf5.0
switch group 5 rule 10 vlan VGROUP permit
switch group 5 deny
switch group 5 start

switch group 15 add eth0 rf5.0
switch group 15 rule 10 vlan VGROUP deny
switch group 15 rule 11 vlan ALL_VLAN permit
switch group 15 deny
switch group 15 start

switch group 25 add eth0 rf5.0
switch group 25 rule 10 vlan ALL_VLAN deny
switch group 25 permit
switch group 25 start
switch admin-group 25

switch start
```

Here three switching groups are created. Group 5 switches the packets with VLAN tags 10, 20-30 and 40. Group 15 switches the packets with any VLAN tag with exception for those switched by group 5. Group 25 is switching all the packets without VLAN tags. Moreover, group 25 will be used to send the traffic to “outer” world.

TCPDUMP expressions (match)

Match expression selects which packets will fit into the group. Only packets for which expression is ‘true’ will be matched to the group.

The expression consists of one or more primitives. Primitives usually consist of an id (name or number) preceeded by one or more qualifiers. There are three different kinds of qualifier:

- **type** - qualifiers say what kind of thing the id name or number refers to. Possible types are **host**, **net, port** and **portrange**. E.g., 'host foo', 'net 128.3', 'port 20', 'portrange 6000-6008'. If there is no type qualifier, **host** is assumed.

- **dir** - qualifiers specify a particular transfer direction to and/or from *id*. Possible directions are **src**, **dst**, **src or dst** and **src and dst**. E.g., 'src foo', 'dst net 128.3', 'src or dst port ftp-data'. If there is no dir qualifier, **src or dst** is assumed. For some link layers, such as SLIP and the "cooked" Linux capture mode used for the "any" device and for some other device types, the **inbound** and **outbound** qualifiers can be used to specify a desired direction.

- **proto** - qualifiers restrict the match to a particular protocol. Possible protos are: **ether**, **fddi**, **tr**, **wlan**, **ip**, **ip6**, **arp**, **rarp**, **decnet**, **tcp** and **udp**. E.g., 'ether src foo', 'arp net 128.3', 'tcp port 21', 'udp portrange 7000-7009'. If there is no proto qualifier, all protocols consistent with the type are assumed. E.g., 'src foo' means '(ip or arp or rarp) src foo' (except the latter is not legal syntax), 'net bar' means '(ip or arp or rarp) net bar' and 'port 53' means '(tcp or udp) port 53'.

'fddi' is actually an alias for 'ether'; the parser treats them identically as meaning "the data link level used on the specified network interface." FDDI headers contain Ethernet-like source and destination addresses, and often contain Ethernet-like packet types, so you can filter on these FDDI fields just as with the analogous Ethernet fields. FDDI headers also contain other fields, but you cannot name them explicitly in a filter expression.

Similarly, 'tr' and 'wlan' are aliases for 'ether'; the previous paragraph's statements about FDDI headers also apply to Token Ring and 802.11 wireless LAN headers. For 802.11 headers, the destination address is the DA field and the source address is the SA field; the BSSID, RA, and TA fields aren't tested.]

In addition to the above, there are some special 'primitive' keywords that don't follow the pattern: **gateway**, **broadcast**, **less**, **greater** and arithmetic expressions. All of these are described below.

More complex filter expressions are built up by using the words **and**, **or** and **not** to combine primitives. E.g., 'host foo and not port ftp and not port ftp-data'. To save typing, identical qualifier lists can be omitted. E.g., 'tcp dst port ftp or ftp-data or domain' is exactly the same as 'tcp dst port ftp or tcp dst port ftp-data or tcp dst port domain'.

Allowable primitives are:

- **dst host** *host* - True if the IPv4/v6 destination field of the packet is *host*, which may be either an address or a name.

- **src host** *host* - True if the IPv4/v6 source field of the packet is *host*.

- **host** *host* - True if either the IPv4/v6 source or destination of the packet is *host*. True if either the IPv4/v6 source or destination of the packet is *host*. Any of the above host expressions can be prepended with the keywords, **ip**, **arp**, **rarp**, or **ip6** as in: **ip host** *host* which is equivalent to: **ether proto** *\ip* **and host** *host* If *host* is a name with multiple IP addresses, each address will be checked for a match.

- **ether dst** *ehost* - True if the Ethernet destination address is *ehost*. *Ehost* may be either a name from /etc/ethers or a number.

- **ether src** *ehost* - True if the Ethernet source address is *ehost*.

- **ether host** *ehost* - True if either the Ethernet source or destination address is *ehost*.

- **gateway** *host* - True if the packet used *host* as a gateway. I.e., the Ethernet source or destination address was *host* but neither the IP source nor the IP destination was *host*. *Host* must be a name and must be found both by the machine’s host-name-to-IP-address resolution mechanisms (host name file, DNS, NIS, etc.) and by the machine’s host-name-to-Ethernet-address resolution mechanism (/etc/ethers, etc.). (An equivalent expression is **ether host** *ehost* **and not host** *host* which can be used with either names or numbers for *host / ehost*.) This syntax does not work in IPv6-enabled configuration at this moment.

- **dst net** *net* - True if the IPv4/v6 destination address of the packet has a network number of *net*. *Net* may be either a name from /etc/networks or a network number (see *networks(5)* for details).

- **src net** *net* - True if the IPv4/v6 source address of the packet has a network number of *net*.

- **net** *net* - True if either the IPv4/v6 source or destination address of the packet has a network number of *net*.

- **net** *net* **mask** *netmask* - True if the IPv4 address matches *net* with the specific *netmask*. May be qualified with **src** or **dst**. Note that this syntax is not valid for IPv6 *net*.

- **net** *net*/*len* - True if the IPv4/v6 address matches *net* with a netmask *len* bits wide. May be qualified with **src** or **dst**.

- **dst port** *port* - True if the packet is ip/tcp, ip/udp, ip6/tcp or ip6/udp and has a destination port value of *port*. The *port* can be a number or a name used in /etc/services (see *tcp*(7) and *udp*(7)). If a name is used, both the port number and protocol are checked. If a number or ambiguous name is used, only the port number is checked (e.g., **dst port 513** will print both tcp/login traffic and udp/who traffic, and **port domain** will print both tcp/domain and udp/domain traffic).

- **src port** *port* - True if the packet has a source port value of *port*.

- **port** *port* - True if either the source or destination port of the packet is *port*.

- **dst portrange** *port1*-*port2* - True if the packet is ip/tcp, ip/udp, ip6/tcp or ip6/udp and has a destination port value between *port1* and *port2*. *port1* and *port2* are interpreted in the same fashion as the *port* parameter for **port**.

- **src portrange** *port1*-*port2* - True if the packet has a source port value between *port1* and *port2*.

- **portrange** *port1*-*port2* - True if either the source or destination port of the packet is between *port1* and *port2*. Any of the above port or port range expressions can be prepended with the keywords, **tcp** or **udp**, as in: **tcp src port** *port* which matches only tcp packets whose source port is *port*.

- **less** *length* - True if the packet has a length less than or equal to *length*. This is equivalent to: **len <=** *length*.

- **greater** *length* - True if the packet has a length greater than or equal to *length*. This is equivalent to: **len >=** *length*.

- **ip proto** *protocol* - True if the packet is an IPv4 packet (see *ip*(4P)) of protocol type *protocol*. *Protocol* can be a number or one of the names **icmp**, **icmp6**, **igmp**, **igrp**, **pim**, **ah**, **esp**, **vrrp**, **udp**, or **tcp**. Note that the identifiers **tcp**, **udp**, and **icmp** are also keywords and must be escaped via backslash (\), which is \\ in the C-shell. Note that this primitive does not chase the protocol header chain.

- **ip6 proto** *protocol* - True if the packet is an IPv6 packet of protocol type *protocol*. Note that this primitive does not chase the protocol header chain.

- **ip6 protochain** *protocol* - True if the packet is IPv6 packet, and contains protocol header with type *protocol* in its protocol header chain. For example, **ip6 protochain 6** matches any IPv6 packet with TCP protocol header in the protocol header chain. The packet may contain, for example, authentication header, routing header, or hop-by-hop option header, between IPv6 header and TCP header. The BPF code emitted by this primitive is complex and cannot be optimized by BPF optimizer code in *tcpdump*, so this can be somewhat slow.

- **ip protochain** *protocol* - Equivalent to **ip6 protochain** *protocol*, but this is for IPv4.

- **ether broadcast -** True if the packet is an Ethernet broadcast packet. The *ether* keyword is optional.

- **ip broadcast -** True if the packet is an IPv4 broadcast packet. It checks for both the all-zeroes and all-ones broadcast conventions, and looks up the subnet mask on the interface on which the capture is being done. If the subnet mask of the interface on which the capture is being done is not available, either because the interface on which capture is being done has no netmask or because the capture is being done on the Linux "any" interface, which can capture on more than one interface, this check will not work correctly.

- **ether multicast -** True if the packet is an Ethernet multicast packet. The **ether** keyword is optional. This is shorthand for '**ether[0] & 1 != 0**'.

- **ip multicast -** True if the packet is an IPv4 multicast packet.

- **ip6 multicast -** True if the packet is an IPv6 multicast packet.

- **ether proto** *protocol* - True if the packet is of ether type *protocol*. *Protocol* can be a number or one of the names **ip**, **ip6**, **arp**, **rarp**, **atalk**, **aarp**, **decnet**, **sca**, **lat**, **mopdl**, **moprc**, **iso**, **stp**, **ipx**, or **netbeui**. Note these identifiers are also keywords and must be escaped via backslash (\). [In the case of FDDI (e.g., '**fddi protocol arp**'), Token Ring (e.g., '**tr protocol arp**'), and IEEE 802.11 wireless LANS (e.g., '**wlan protocol arp**'), for most of those protocols, the protocol identification comes from the 802.2 Logical Link Control (LLC) header, which is usually layered on top of the FDDI, Token Ring, or 802.11 header. When filtering for most protocol identifiers on FDDI, Token Ring, or 802.11, *tcpdump* checks only the protocol ID field of an LLC header in so-called SNAP format with an Organizational Unit Identifier (OUI) of 0x000000, for encapsulated Ethernet; it doesn't check whether the packet is in SNAP format with an OUI of 0x000000. The exceptions are:
  - **iso -** *tcpdump* checks the DSAP (Destination Service Access Point) and SSAP (Source Service Access Point) fields of the LLC header;
  - **stp** and **netbeui -** *tcpdump* checks the DSAP of the LLC header;
  - **atalk -** *tcpdump* checks for a SNAP-format packet with an OUI of 0x080007 and the AppleTalk etype.

  In the case of Ethernet, *tcpdump* checks the Ethernet type field for most of those protocols. The exceptions are:

  - **iso**, **stp**, and **netbeui -** *tcpdump* checks for an 802.3 frame and then checks the LLC header as it does for FDDI, Token Ring, and 802.11;
  - **atalk -** *tcpdump* checks both for the AppleTalk etype in an Ethernet frame and for a SNAP-format packet as it does for FDDI, Token Ring, and 802.11;

  - **aarp -** *tcpdump* checks for the AppleTalk ARP etype in either an Ethernet frame or an 802.2 SNAP frame with an OUI of 0x000000;
  - **ipx -** *tcpdump* checks for the IPX etype in an Ethernet frame, the IPX DSAP in the LLC header, the 802.3-with-no-LLC-header encapsulation of IPX, and the IPX etype in a SNAP frame.

- **decnet src** *host* - True if the DECNET source address is *host*, which may be an address of the form "10.123", or a DECNET host name. [DECNET host name support is only available on ULTRIX systems that are configured to run DECNET.]

- **decnet dst** *host* - True if the DECNET destination address is *host*.

- **decnet host** *host* - True if either the DECNET source or destination address is *host*.

- **ifname** *interface* - True if the packet was logged as coming from the specified interface (applies only to packets logged by OpenBSD's **pf**(4)).

- **on** *interface* - Synonymous with the **ifname** modifier.

- **rnr** *num* - True if the packet was logged as matching the specified PF rule number (applies only to packets logged by OpenBSD's **pf**(4)).

- **rulenum** *num* - Synonomous with the **rnr** modifier.

- **reason** *code* - True if the packet was logged with the specified PF reason code. The known codes are: **match**, **bad-offset**, **fragment**, **short**, **normalize**, and **memory** (applies only to packets logged by OpenBSD's **pf**(4)).

- **rset** *name* - True if the packet was logged as matching the specified PF ruleset name of an anchored ruleset (applies only to packets logged by **pf**(4)).

- **ruleset** *name* - Synonomous with the **rset** modifier.

- **srnr** *num* - True if the packet was logged as matching the specified PF rule number of an anchored ruleset (applies only to packets logged by **pf**(4)).

- **subrulenum** *num* - Synonomous with the **srnr** modifier.

- **action** *act* - True if PF took the specified action when the packet was logged. Known actions are: **pass** and **block** (applies only to packets logged by OpenBSD's **pf**(4)).

- **ip**, **ip6**, **arp**, **rarp**, **atalk**, **aarp**, **decnet**, **iso**, **stp**, **ipx**, *netbeui* - Abbreviations for: **ether proto** *p* where *p* is one of the above protocols.

- **lat**, **moprc**, **mopdl -** Abbreviations for: **ether proto** *p* where *p* is one of the above protocols. Note that *tcpdump* does not currently know how to parse these protocols.

- **vlan** *[vlan_id]* - True if the packet is an IEEE 802.1Q VLAN packet. If *[vlan_id]* is specified, only true if the packet has the specified *vlan_id*. Note that the first **vlan** keyword encountered in *expression* changes the decoding offsets for the remainder of *expression* on the assumption that the packet is a VLAN packet. The **vlan** *[vlan_id]* expression may be used more than once, to filter on VLAN hierarchies. Each use of that expression increments the filter offsets by 4. For example: **vlan 100 && vlan 200** filters on VLAN 200 encapsulated within VLAN 100, and **vlan && vlan 300 && ip** filters IPv4 protocols encapsulated in VLAN 300 encapsulated within any higher order VLAN.

- **mpls** *[label_num]* - True if the packet is an MPLS packet. If *[label_num]* is specified, only true is the packet has the specified *label_num*.

Note that the first **mpls** keyword encountered in *expression* changes the decoding offsets for the remainder of *expression* on the assumption that the packet is a MPLS-encapsulated IP packet. The **mpls** *[label_num]* expression may be used more than once, to filter on MPLS hierarchies. Each use of that expression increments the filter offsets by 4. For example: **mpls 100000 && mpls 1024** filters packets with an outer label of 100000 and an inner label of 1024, and **mpls && mpls 1024 && host 192.9.200.1** filters packets to or from 192.9.200.1 with an inner label of 1024 and any outer label.

- **pppoed -** True if the packet is a PPP-over-Ethernet Discovery packet (Ethernet type 0x8863).

- **pppoes -** True if the packet is a PPP-over-Ethernet Session packet (Ethernet type 0x8864). Note that the first **pppoes** keyword encountered in *expression* changes the decoding offsets for the remainder of *expression* on the assumption that the packet is a PPPoE session packet. For example: **pppoes && ip** filters IPv4 protocols encapsulated in PPPoE.

- **tcp**, **udp**, **icmp -** Abbreviations for: **ip proto** *p* **or ip6 proto** *p* where *p* is one of the above protocols.

- **iso proto** *protocol* - True if the packet is an OSI packet of protocol type *protocol*. *Protocol* can be a number or one of the names **clnp**, **esis**, or **isis**.

- **clnp**, **esis**, **isis -** Abbreviations for: **iso proto** *p* where *p* is one of the above protocols.

- **l1**, **l2**, **iih**, **lsp**, **snp**, **csnp**, **psnp -** Abbreviations for IS-IS PDU types.

- **vpi** *n* - True if the packet is an ATM packet, for SunATM on Solaris, with a virtual path identifier of *n*.

- **vci** *n* - True if the packet is an ATM packet, for SunATM on Solaris, with a virtual channel identifier of *n*

- **lane -** True if the packet is an ATM packet, for SunATM on Solaris, and is an ATM LANE packet. Note that the first **lane** keyword encountered in *expression* changes the tests done in the remainder of *expression* on the assumption that the packet is either a LANE emulated Ethernet packet or a LANE LE Control packet. If **lane** isn't specified, the tests are done under the assumption that the packet is an LLC-encapsulated packet.

- **llc -** True if the packet is an ATM packet, for SunATM on Solaris, and is an LLC-encapsulated packet.

- **oamf4s -** True if the packet is an ATM packet, for SunATM on Solaris, and is a segment OAM F4 flow cell (VPI=0 & VCI=3).

- **oamf4e -** True if the packet is an ATM packet, for SunATM on Solaris, and is an end-to-end OAM F4 flow cell (VPI=0 & VCI=4).

- **oamf4 -** True if the packet is an ATM packet, for SunATM on Solaris, and is a segment or end-to-end OAM F4 flow cell (VPI=0 & (VCI=3 | VCI=4)).

- **oam -** True if the packet is an ATM packet, for SunATM on Solaris, and is a segment or end-to-end OAM F4 flow cell (VPI=0 & (VCI=3 | VCI=4)).

- **metac -** True if the packet is an ATM packet, for SunATM on Solaris, and is on a meta signaling circuit (VPI=0 & VCI=1).

- **bcc -** True if the packet is an ATM packet, for SunATM on Solaris, and is on a broadcast signaling circuit (VPI=0 & VCI=2).

- **sc -** True if the packet is an ATM packet, for SunATM on Solaris, and is on a signaling circuit (VPI=0 & VCI=5).

- **ilmic -** True if the packet is an ATM packet, for SunATM on Solaris, and is on an ILMI circuit (VPI=0 & VCI=16).

- **connectmsg -** True if the packet is an ATM packet, for SunATM on Solaris, and is on a signaling circuit and is a Q.2931 Setup, Call Proceeding, Connect, Connect Ack, Release, or Release Done message.

- **metaconnect -** True if the packet is an ATM packet, for SunATM on Solaris, and is on a meta signaling circuit and is a Q.2931 Setup, Call Proceeding, Connect, Release, or Release Done message.

- *expr relop expr* - True if the relation holds, where *relop* is one of >, <, >=, <=, =, !=, and *expr* is an arithmetic expression composed of integer constants (expressed in standard C syntax), the normal binary operators [+, -, *, /, &, |, <<, >>], a length operator, and special packet data accessors. Note that all comparisons are unsigned, so that, for example, 0x80000000 and 0xffffffff are > 0. To access data inside the packet, use the following syntax: *proto* **[** *expr* **:** *size* **]** *Proto* is one of **ether, fddi, tr, wlan, ppp, slip, link, ip, arp, rarp, tcp, udp, icmp, ip6** or **radio**, and indicates the protocol layer for the index operation. (**ether, fddi, wlan, tr, ppp, slip** and **link** all refer to the link layer. **radio** refers to the "radio header" added to some 802.11 captures.) Note that *tcp, udp* and other upper-layer protocol types only apply to IPv4, not IPv6 (this will be fixed in the future). The byte offset, relative to the indicated protocol layer, is given by *expr*. *Size* is optional and indicates the number of bytes in the field of interest; it can be either one, two, or four, and defaults to one. The length operator, indicated by the keyword **len**, gives the length of the packet.

  For example, '**ether[0] & 1 != 0**' catches all multicast traffic. The expression '**ip[0] & 0xf != 5**' catches all IPv4 packets with options. The expression '**ip[6:2] & 0x1fff = 0**' catches only unfragmented IPv4 datagrams and frag zero of fragmented IPv4 datagrams. This check is implicitly applied to the **tcp** and **udp** index operations. For instance, **tcp[0]** always means the first byte of the TCP *header*, and never means the first byte of an intervening fragment.

  Some offsets and field values may be expressed as names rather than as numeric values. The following protocol header field offsets are available: **icmptype** (ICMP type field), **icmpcode** (ICMP code field), and **tcpflags** (TCP flags field).

  The following ICMP type field values are available: **icmp-echoreply**, **icmp-unreach**, **icmp-sourcequench**, **icmp-redirect**, **icmp-echo**, **icmp-routeradvert**, **icmp-routersolicit**, **icmp-timxceed**, **icmp-paramprob**, **icmp-tstamp**, **icmp-tstampreply**, **icmp-ireq**, **icmp-ireqreply**, **icmp-maskreq**, **icmp-maskreply**.

  The following TCP flags field values are available: **tcp-fin**, **tcp-syn**, **tcp-rst**, **tcp-push**, **tcp-ack**, **tcp-urg**.

Primitives may be combined using:

- A parenthesized group of primitives and operators (parentheses are special to the Shell and must be escaped).

- Negation ('**!**' or '**not**').

- Concatenation ('**&&**' or '**and**').

- Alternation ('**||**' or '**or**').

Negation has highest precedence. Alternation and concatenation have equal precedence and associate left to right. Note that explicit **and** tokens, not juxtaposition, are now required for concatenation.

If an identifier is given without a keyword, the most recent keyword is assumed. For example,

```
not host vs and ace
```

is short for

```
not host vs and host ace
```

which should not be confused with

```
not ( host vs or ace )
```

Expression arguments can be passed to *tcpdump* as either a single argument or as multiple arguments, whichever is more convenient. Generally, if the expression contains Shell metacharacters, it is easier to pass it as a single, quoted argument. Multiple arguments are concatenated with spaces before being parsed.

EXAMPLES

To print all packets arriving at or departing from *sundown*:

```
tcpdump host sundown
```

To print traffic between *helios* and either *hot* or *ace*:

```
tcpdump host helios and \( hot or ace \)
```

To print all IP packets between *ace* and any host except *helios*:

```
tcpdump ip host ace and not helios
```

To print all traffic between local hosts and hosts at Berkeley:

```
tcpdump net ucb-ether
```

To print all ftp traffic through internet gateway *snup*: (note that the expression is quoted to prevent the shell from (mis-)interpreting the parentheses):

```
tcpdump 'gateway snup and (port ftp or ftp-data)'
```

To print traffic neither sourced from nor destined for local hosts (if you gateway to one other net, this stuff should never make it onto your local net).

```
tcpdump ip and not net localnet
```

To print the start and end packets (the SYN and FIN packets) of each TCP conversation that involves a non-local host.

```
tcpdump 'tcp[tcpflags] & (tcp-syn|tcp-fin) != 0 and not src and dst net localnet'
```

To print all IPv4 HTTP packets to and from port 80, i.e. print only packets that contain data, not, for example, SYN and FIN packets and ACK-only packets. (IPv6 is left as an exercise for the reader.)

```
tcpdump 'tcp port 80 and (((ip[2:2] - ((ip[0]&0xf)<<2)) - ((tcp[12]&0xf0)>>2)) != 0)'
```

To print IP packets longer than 576 bytes sent through gateway *snup*:

```
tcpdump 'gateway snup and ip[2:2] > 576'
```

To print IP broadcast or multicast packets that were *not* sent via Ethernet broadcast or multicast:

```
tcpdump ’ether[0] & 1 = 0 and ip[16] >= 224’
```

To print all ICMP packets that are not echo requests/replies (i.e., not ping packets):

```
tcpdump ’icmp[icmptype] != icmp-echo and icmp[icmptype] != icmp-echoreply’
```

11.CES command

Ces command is used for Infinet Wireless devices with IDU-5000-E1SCR, IDU-5000-E1RJ indoor units.

Ces command allows controlling parameters of TDM flow packing module (CES-module), as well as related WANFleX parameters for **CES-over-WLAN** mode.

Brief description of **CES-over-WLAN** mode (monopolistic):

CES-module (Master) of Infinet Wireless indoor unit receiving TDM flow from an external source accumulates a given number of TDM frames transforming them into Ethernet frame which is then transferred to outdoor unit. Outdoor unit in turn transmits a frame through a radio channel to an outdoor unit of another device. From outdoor unit of that device a frame comes to CES-module (Slave), which provides precise time restoration and original TDM frame transfer to external receiver.

In CES-over-WLAN mode outdoor unit of Infinet Wireless device process packets from CES-module ensuring their transmission to the other side of a radio link with a minimum delay. In this mode normal non-CES data is not transferred directly into the radio channel. For the non-CES data transfer does not interfere with CES packets transfer, non-CES data packets are broken into smaller parts and these parts can be added to each CES packet expanding it with a specified number of bytes as specified in the configuration. At a receiving side normal non-CES data parts of packets are taken from CES packets and gathered into complete packets.

Attention!!! CES-module only supports a Single Clock Domain. It is the customer's responsibility to feed the CES containing device with TDM ports, all of which are timed by the same clock source.

Syntax:

```
ces start
ces stop
ces clear
ces unit <addr>
ces mode (e1|t1) (line|internal|loopback|recovery)
ces ports <list>
ces frames <N>
ces maxjitter <N>
ces adjust (now|auto [clear]) <list>
ces stat [clear]
ces stat delay [clear]
ces eth media <MediaType>
ces eth bwlimit <Kbps>
ces eth stat [full] [clear]


MediaType: 100BaseTX-fullduplex, 100BaseTX-halfduplex,
           10BaseT-fullduplex, 10BaseT-halfduplex,
           auto
```

ces start – Starts a process which controls CES-module. If there are pre-configured ports TDM data transfer begins.

ces stop – Stops a process which controls CES-module.

ces clear – Clears **ces** configuration.

ces unit <addr> - Sets IP address of CES-module and reserves the corresponding network with the “24” mask . IP-addresses of this network are used for ODU-to-CES-module communication. Default value: 169.254.1.100 - 169.254.1.0/24 network is reserved.

ces mode (e1|t1) (line|internal|loopback|recovery) – Sets CES-module mode. **(e1|t1)** – type of interface: E1 or T1. **(line|internal|loopback|recovery)** – synchronization mode: **line** – input frequency of port 0 is used as an output frequency for all ports, **internal** – internal frequency generator is used, **loopback** – input frequency of each port is used as an output frequency for the port, **recovery** – frequency is restored from CES-over-WLAN packet flow.

ces ports <list> - List of port numbers divided by gaps.

Example:

```
ces ports 0 2 3
```

ces frames <N> - Number of TDM frames for one CES-over-WLAN packet. Number of packets per second which are being sent for each port depends on this parameter. Valid values are: for E1 – any value from 1 to 25 and from 26 to 44, for T1 – any value from 1 to 33 and from 34 to 60.

ces maxjitter <N> - Maximum expected jitter for given link. Valid values are: from 1 to 200 but not less than the option **frames** divided by 8.

ces adjust (now|auto [clear]) <list> - Parameters of jitter buffer self-tuning. **now –** jitter buffer self-tuning will be executed once, **auto** - jitter buffer self-tuning will be executing automatically and continuously, **auto clear** – stops automatic self-tuning. **<list> -** list of values format: <port> [/ threshold], where threshold sets threshold for current jitter filling deviation from given **maxjitter** value. It can be a number (may be fractional to three decimal places) in milliseconds or a percentage (for example: 45%). After exceeding the threshold jitter buffer self-tuning procedure will be started for given port. Threshold default value is 50%.

Example:

```
ces adjust auto 0/1.5 1/15% 2/15
```

In this example jitter buffer state monitoring will start on ports 0, 1 and 2. Jitter buffer self-tuning will be performed on port 0 if jitter filling deviation on this port is 1.5 milliseconds, on port 1 - if jitter filling deviation on this port is 15 percent, on port 2 - if jitter filling deviation on this port is 15 milliseconds.

ces stat [clear] – Shows **ces** statistics. Optional parameter **clear** allows to clear statistics.

ces eth media <MediaType> - Sets LAN port physical interface properties of an IDU. Allowed **<MediaType>** values: **100BaseTX-fullduplex**, **100BaseTX-**

halfduplex, **10BaseT-fullduplex**, **10BaseT-halfduplex**, **auto. Auto** value by default.

ces eth bwlimit <Kbps> - Limits LAN port physical interface speed of an IDU. Possible values: **10-100000**. Measured in Kbps. No limit by default. «**0**» value cancels the limit.

ces eth stat [full] [clear] – shows CES-module's Ethernet port statistics. Optional parameter **full** allows viewing detailed statistics. Optional parameter **clear** allows clearing statistics. When using dual-radio devices such as "CES multihop repeater" this command shows the same statistics as **ces stat delay** command does.

ces stat command output:

```
>ces stat

unit status: slave, active
time: 00:06:42 since started, 00:00:02 since stat cleared
mode: e1 recovery
peer: 00134676EEAC

master settings:
~~~~~~~~~~~~~~~~
mode: e1 loopback
jitterbuf: 40
frames: 16
datapad: 32
ports: 0

physical interface: 0 tx fifo underruns, 0 rx fifo overruns
~~~~~~~~~~~~~~~~~~
link status octets in/out frames in/out
port 0: up NoAlarm 37173184/39834496 1161662/1244828
port 1: up NoAlarm 0/39744608 0/1242019
port 2: down LossOfSignal 16632896/39654336 519778/1239198
port 3: down LossOfSignal 0/39554560 0/1236080

packet interface:
~~~~~~~~~~~~~~~~~
port 0:
tx: on   restarts: 0
rx: on   restarts: 0
jitter buffer: 7.769 cur (0.231 dev.), 6.953 min, 8.406 max
packets: total 39523, 500 per second, 100% valid
0 R bit, 0 L bit, 0 late, 0 lost,
0 out of order, 0 underrun, 0 overrun,
0 invalid sequence, 0 duplicate, 0 malformed
port 1: disabled
port 2: disabled
port 3: disabled
```

Where, **unit status** – shows CES-module status. **Slave/master** – common synchronization mode on CES-module. **Slave** is a recovery mode, master – non-recovery mode.

time - time from **ces start** command input, time from last statistics clearing.

mode – shows **ces mode** parameter configuration.

peer – Mac-address of neighboring device with which CES-over-WLAN transmission is performed.

master settings – shows Ces-Master-module configuration (only in **Slave** mode). This configuration is used on **Slave** module.

physical interface (between IW CES-module and external TDM equipment) – shows E1/T1 ports state.

packet interface (between CES-modules of point-to-point IW devices) – shows a state of CES-over-WLAN packet flows for each port. It is seen that receiving (rx) and transmission (tx) are functioning and there were no restarts for them.

jitter buffer – shows current state of jitter buffer. Immediate buffer filling correspond 7.769 milliseconds delay which deviate from operator set value (8) by 0.231 milliseconds. Minimum recorded value of this parameter since statistic clearing is - 6.953 ms, maximum - 8.406 ms.

packets – shows that 39523 CES-over-WLAN packets were processed altogether (**total**), current receiving speed of such packets is - 500 packets per second. **100% valid** means that all 500 expected packets per second were received without mistakes.

R bit – a number of packets having R bit set in check word CESoPSN

L bit - a number of packets having L bit set in check word CESoPSN

late – a number of packets which come too late for restoring synchronization

lost – a number of lost packets

out of order – a number of packets which were received out of order

underrun – a number of filling frames which were sent into physical port because jitter buffer was empty

overrun – a number of discarded packets because of repletion of jitter buffer

invalid sequence – a number of discarded packets because of considerable ordinal number deviation from expected value

duplicate – a number of duplicates of received packets

malformed – a number of packets which size didn't corresponds to expected

ces stat delay [clear] - Shows statistics on incoming CES-packets delays at ODU's radio interface. Optional parameter *clear* allows to reset statistics. When using dual-radio devices such as "CES multihop repeater" this command shows statistics on passing CES-packets delays.

Minimum configuration of Infinet Wireless devices with 4 E1/T1 ports:

Master Device:

```
rf rf5.0 freq 5200 bitr 36000 sid 10101010 txpwr 18
mint rf5.0 start
ces mode e1 loopback
ces ports 0 1 2 3
ces start
```

Slave Device:

```
rf rf5.0 freq 5200 bitr 36000 sid 10101010 txpwr 18
mint rf5.0 start
ces mode e1 recovery
ces start
```

In this example frequency, radio power and **sid** are configured on **Master** device. MINT starts. List of ports is appointed. And CES-over-WLAN mode starts with default settings: interface type **e1**, synchronization mode **loopback**.

On a Slave device frequency, radio power and **sid** is configured. MINT starts. Interface type **e1** and synchronization mode **recovery** is set. Other settings (except **ces adjust**) will be received from **Master** device. CES-over-WLAN mode starts.

12.WiFi (Wi-Fi AP configuration)

The command is used to configure remote Wi-Fi AP.

Syntax:

```
wifi IFNAME [options] [command]
  where IFNAME are WiFi Interface name (for example, wifi0)
  where options are:
      -name <"string">         - Set optional interface name
      -ssid <"string">         - Define Service Set Identification
      -freq <frequency>        - Set center frequency (MHz; see "wifi IFNANE cap"
for a list of supported frequencies)
      -power <power>           - Set Tx power (mW; see "wifi IFNANE cap" for a list
of supported values)
      -antenna <antenna>       - Select antenna (see "licence --show" for allowed
antenna types)
      -pass <key>              - Set authentication key. Key format:
[ascii|hex:]<string>.
      -auth <type>             - Select authentication type (see "wifi IFNANE cap"
for supported auth types)
      -wpa-encrypt <type>      - Select WPA encryption type (see "wifi IFNANE cap"
for supported encryption types; different encryption types can be combined with
comma as delimiter)
      -auth-server <address> - Set authentication server address. Use
<host>[:<port>],<secret> format for the address.
      -acct-server <address>  - Set accounting server address. Use
host>[:<port>],<secret> format for the address.
      -debug-radius on|off     - Enable/disable RADIUS conversation debugging.
      -hidden on|off           - Enable/disable hidden AP mode
      -wme on|off              - Enable/disable Wireless Multimedia Extensions
      (QoS)
      -isolation on|off        - Enable/disable client stations isolation
      -rsn-preauth on|off      - Enable/disable RSN/WPA2 pre-authentication
      -mcast-rate <rate>       - Set multicast packets bitrate
      -acct-interval <iterval>  - Set interim accounting update interval in seconds
 where commands are:
```

cap	*- Show IF capability*
status	*- Show status of IF*
stations	*- Show associated stations list*

Description:

For Wi-Fi AP activation use *ifc wifi0 up* command.

uname <"string"> - sets NAS identifier of the device used in Radius authentification process

ssid <"string"> - sets Service Set Identity name of Wi-Fi network, where *string* - Service Set Identity name.

freq <frequency> - sets AP frequency in MHz. Use "wifi IFNANE cap" command to view a list of supported frequencies.

power <power> - sets AP output in mW. Use "wifi IFNANE cap" command to view a list of supported values.

antenna <antenna> - allow to choose active antenna for Wi-Fi AP operation. In the *diversity* mode device will choose what antenna to use for each client automatically regarding link quality (only for devices with 2 antennas). Use "li --show" command to view a list of supported antenna types.

auth <type> - sets clients authentification type for connection to AP. Use "wifi IFNANE cap" command to view a list of supported authentification types.

pass <key> - sets clients authentification password key for connection to AP. **Key** parameter format: [ascii|hex:]<string>, where **ascii|hex** options defines the format of the password key and **<key>** parameter – password key.

wpa-encrypt <type> - sets traffic encryption type between client and AP when using WPA authentification. Use "wifi IFNANE cap" command to view a list of supported encryption types. Different encryption types can be combined with comma as delimiter.

-auth-server <address> - sets Radius authentication server address and password. Use <host>[:<port>],<secret> format for the address, where **<host>[:<port>]** options defines server IP address and port, **<secret>** option defines password.

-acct-server <address> - sets Radius accounting server address and password. Use <host>[:<port>],<secret> format for the address, where **<host>[:<port>]** options defines server IP address and port, **<secret>** option defines password.

-debug-radius on|off – enable/disable output of Radius server system messages.

-hidden on|off – enable/disable hidden AP mode. In a hidden mode AP do not appear in a search output of the available wireless networks.

-wme on|off - enable/disable Wireless Multimedia Extensions (QoS) support according to 802.11e protocol.

-isolation on|off – restricts/allows client stations of the given AP establish a connection between each other directly (without using external routers).

-rsn-preauth on|off - enable/disable RSN/WPA2 pre-authentication (quick roaming) support according to 802.11r protocol.

-mcast-rate <rate> - sets multicast packets bitrate.

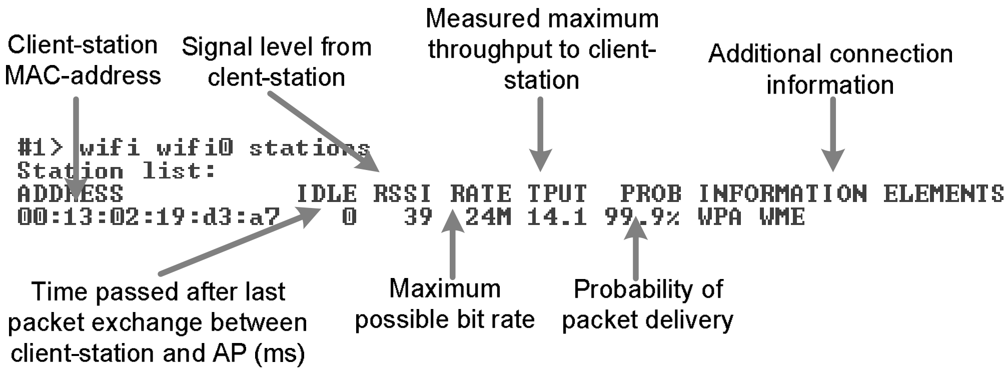
-acct-interval <iterval> - sets interim accounting update interval in seconds according to which Radius server traffic tracking messages are sending.

cap – shows list of AP's capabilities.

status – shows AP status.

stations – shows information on all client stations of the AP:

«Stations» command output:



```
#1> wifi wifi0 stations
Station list:
ADDRESS           IDLE RSSI RATE TPUT  PROB INFORMATION ELEMENTS
00:13:02:19:d3:a7   0   39  24M 14.1 99.9% WPA WME
```

Client-station MAC-address
Signal level from clent-station
Measured maximum throughput to client-station
Additional connection information
Time passed after last packet exchange between client-station and AP (ms)
Maximum possible bit rate
Probability of packet delivery

Sample configuration:

```
#WiFI remote AP configuration
  wifi wifi0 -ssid "IW"
  wifi wifi0 -freq 2432
  wifi wifi0 -power 63
  wifi wifi0 -antenna "diversity"
  wifi wifi0 -auth "WPA-PSK"
  wifi wifi0 -pass "SecretHotspot"
  wifi wifi0 -wpa-encrypt "RC4"
  wifi wifi0 -syslog on
```

13. Dfs (Dynamic Frequency Selection)

This command is used to configure DFS (Dynamic Frequency Selection) function of a radio interface.

Syntax:

```
dfs "interface_name" (dfsradar | dfsonly | dfsoff)
dfs "interface_name" freq { all |"frequency_list"}
dfs "interface_name" cot hh:mm
dfs "interface_name" scansec <seconds>
```

Description:

- dfs "interface_name" dfsonly – starts DFS on the device. In DFS mode device is selecting the most interference-free frequency channel by scanning all available frequencies. When scanning is done it sets the radio interface to operate on the frequency that has less presence of external interference sources.

While scanning and choosing the best channel the DFS function also takes into consideration the "density" characteristic of the radio environment. This characteristic indicates how much impulse interference was detected on the channel during the scan time. The "density" measurement results are added into the system log along with the other DFS scanning information.

- dfs "interface_name" dfsradar – starts DFS with radar detection. After choosing the most "clear" frequency channel the device is listening to radars that may work on the specified frequency. In case of detecting the radar it starts frequency selection process again.

- dfs "interface_name" dfsoff – stops DFS on the device.

- dfs "interface_name" freq {all |"frequency_list"} – sets list of frequencies ("*frequency_list"* parameter) that are allowed for choosing by DFS or allows DFS to use all radio interface enabled frequencies ("*all"* parameter).

- dfs "interface_name" cot {hh:mm | off} – allows doing DFS rescanning and choosing the most optimal frequency for using on the daily basis in the defined time that is set by "hh:mm" parameter. "Off" parameter disables this functionality.

- dfs "interface_name" scansec <seconds> – sets the time that is spent on scanning each available frequency in seconds. By default: 6 seconds.

DFS default operational characteristics:

- Channel occupation time: 24 hours
- Scanned time: 6 seconds for each available frequency
- Listening to the Radar on the chosen frequency: 1 minute

DFS Master/Slave configuration:

"DFS Master" is a unit that process actual frequency selection and radar detection functions. "DFS Slave" is a unit that does not choose the frequency itself but follows "DFS Master's" frequency settings. For example, in PTP link one unit should be configured as a "DFS Master" and another one as a "DFS Slave".

It is strongly recommended to set as a "DFS Master" the unit that is working in worthier interference conditions.

To set a unit as a "DFS Master" (example):

1. Set a unit as "Master":

```
mint rf5.0 –type master
```

2. Configure it as a "Roaming Leader":

```
mint rf5.0 roaming leader
```

Please refer to the "Frequency roaming" chapter for detailed description.

3. Start DFS (if not already started).

```
dfs rf5.0 dfsonly
```

The unit will perform DFS functions and send the chosen frequency information to the "DFS Slave" devices.

To set a unit as a “DFS Slave” (example):

1. Set a unit as “Slave”:

```
mint rf5.0 –type slave
```

2. Configure the corresponding “Roaming Profile” on the unit:

```
mint rf5.0 profile 1 –freq auto
mint rf5.0 roaming enable
```

Please refer to the “Frequency roaming” chapter for detailed description.

The unit will work on the same frequency as the “DFS Master” unit.

IV. Layer 3 Command Set – IP Networking

1. Ifconfig command (interfaces configuration)

The command is used to set and view the configuration of the unit's network interfaces.

Syntax:

```
ifconfig IFNAME
[info "TEXT up to 72 chars"]
[address[/netmask] [ [delete | -alias] [ up ] [ down ]
[mtu N]
[link0 | link1 | link2]
[media MediaType]]
[vlan TAG  [-]vlandev IFParent]
ifconfig –a

MediaType:
        1000BaseFX-fullduplex, 1000BaseFX-halfduplex,
        1000BaseTX-fullduplex, 1000BaseTX-halfduplex,
       100BaseTX-fullduplex, 100BaseTX-halfduplex,
       10BaseT-fullduplex, 10BaseT-halfduplex,
       auto
```

Description:

The command allows setting and viewing the configuration of the unit's interfaces specified by their ID numbers.

The command has the following parameters:

- **IFNAME**: specifies the name of an interface (to see all the unit's interface names, the **ifconfig -a** or **netstat -i** commands may be executed).
- **info**: allows adding a text note of up to 72 characters to the interface configuration.
- **address**: specifies the IP-address assigned to the interface. May be specified as:
  - IP-address/number of bits in the mask
  - IP-address:mask
  - IP-address

Example:

```
ifconfig eth0 192.168.1.1/26
ifconfig eth0 192.168.1.1:255.255.255.192
ifconfig eth0 192.168.1.1
```

delete| -alias: alias flag in the interface configuration 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 as an alias address.

For example, after executing the following commands:

```
ifconfig eth0 193.124.189.1/27 up
ifconfig eth0 10.0.0.1
```

There will be two IP-addresses from two different networks assigned to the same **eth0** interface.

To remove an IP-address from the unit's interface, the "ifconfig" command is executed with the **delete** or **-alias** option following the IP-address to be removed.

⚠ *When "ifconfig eth0 delete" command is executed the CES module (if connected) becomes inactive. Only unit's restart can reactivate the module.*

Example:

```
ifconfig eth0 193.124.189.1/27 -alias
```

The **[-]alias** option is applicable to any IP-address, that is to say, all IP-addresses assigned to the interface are considered as equivalent aliases. If the first (primary) address is removed, the next (in the order of their assignment) becomes primary.

up|down: enables/disables the interface.

<u>System limitations:</u>

The **lo0** interface cannot be set to the **down** state. Radio interfaces' (**rfX.X**) states are not saved in the configuration (after rebooting all radio interfaces of the unit are in the **up** state).

Examples:

```
ifconfig eth0 up
ifconfig eth0 1.1.1.1/24 up
ifconfig rf5.0 down
```

mtu N: sets the desirable MTU (Maximum Transfer Unit) size of the packet for the interface (in bytes). The allowed range is from 72 to 1580. Default value is 1500. 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 the tunnel interfaces.

⚠ *MTU paramater makes sense only on Layer 3 level(in routing mode).*

Media parameter allows specifying physical Ethernet interface 10/100/1000 type.

<u>Allowed MediaType values (model dependent):</u>

1000BaseFX-fullduplex, 1000BaseFX-halfduplex, 1000BaseTX-fullduplex, 1000BaseTX-halfduplex, 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 vlan1 -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).

One can enable **IEEE 802.1Q-in-Q** support on the unit. IEEE 802.1Q-in-Q allows adding an outer VLAN ID tag to IEEE 802.1Q tagged traffic forming so called “double-tagged” frames, thus, making it possible to encapsulate multiple штук VLANs within one single outer VLAN.

To configure IEEE 802.1Q-in-Q VLAN tagging on the unit the “**link0**” option of the “**ifconfig”** command is used. When this option is enabled the unit’s interface terminates the specified outer VLAN ID tag of the incoming doubled-tagged traffic and assigns the outer tag to the outgoing tagged traffic.

Example:

```
ifc vlan2 1.1.1.1/24 up
ifc vlan2 vlan 2 vlandev eth0 link0
```

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

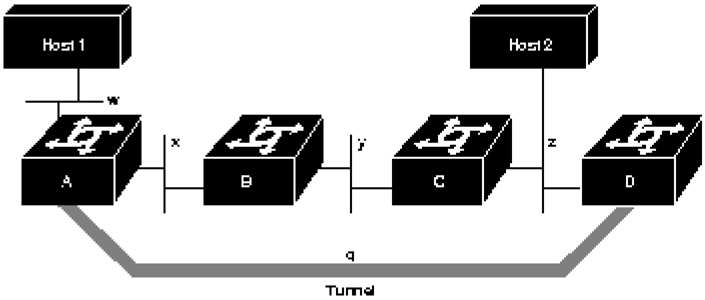
To see the configuration of all interfaces of the unit, use the **ifconfig -a** command.

2. Tun command (tunnels building)

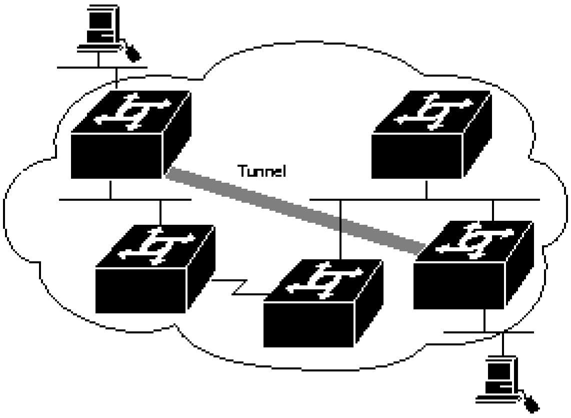
The command specifies the parameters of a software tunnel.

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 ADDR dst ADDR [mtu N] [[-]df] \ 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
```

When **-df** option is enabled the unit clears DF (Don`t Fragment) flag in the header of all IP packets passing through the tunnel.

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 device. 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.

Syntax:

```
qm option {[-]rtp [-]dot1p [-]tos [-]icmp [-]tcpack [-]strict}
qm classN {[max=N] [ceil=N] [ceilprio=N] [parent=N]} | {clear}
qm chN [max=N[%]|0] [[ceil=N[%]|0] [ceilprio=N|0] [latency=N|0] [pri=P] [[no]strict]] [pps=N|0] [to=addr] [vlan=N|-1] [dscp=N|-1] [dot1p=N|-1] [classL] / clear
qm stat [full] [clear]
del RULE_NUMBER
mov RULE_A RULE_B
rearrange [N]
qm add[out] [num] [ifname] chN [pass] rule...
```

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 max=N
```

This command creates a service class **#L**. It is used for dynamic bandwidth allocation between different channels. The “**max = N”** option defines the total bandwidth of the class that will be limited to a given value (thousands bps).

To delete the class:

```
qm classL clear
```

You can create a hierarchy of service classes where a "parent" class is used for the dynamic allocation of its bandwidth between its subsidiary classes. To do this **[ceil=N] [ceilprio=N] [parent=N]** parameters are used. The use of **[ceil=N] [ceilprio=N]** parameters is the same as in **qm chN** command. **[parent=N]** parameter defines "parent" class for the current class, where **N** – is a value of a "parent" class.

qm chN [max=N[%]|0] [[ceil=N[%]|0] [ceilprio=N|0] [latency=N|0] [pri=P] [[no]strict]] [pps=N|0] [to=addr] [vlan=N|-1] [dscp=N|-1] [dot1p=N|-1] [classL] / clear

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

- **max=N[%]** sets maximum data rate for the channel in Kbit/s. Value range: from 10 to 100000. It is also possible to set it in per cent (**max=N%**) of the parent class' total bandwidth. 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 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 class. When, there are several such channels competing for extra bandwidth, it is equally divided between them. (See examples below).

Exception: on the H02 platform, if there are several channels competing for extra bandwidth of their parent class, the bandwidth is divided between them proportionally to their respective predefined data rate limits.

- **ceil=N[%]** determines how much of the total bandwidth of the parent class L can be used by the channel when the class' bandwidth is not used entirely. Measured either in kilobits per second or per cent (**ceil=N%**) of the parents class' total bandwidth. To disable the parameter set its value to 0.

- **ceilprio=N** sets priority for the channel that is used when interface bandwidth can be used by several channels. There are 17 priorities from 0 (the highest) to 16 (the lowest). Default value is 0 therefore when setting another value it is possible only to lower the priority.

- **latency=N** determines the maximum time for the packets to stay in the channel. If a packet is waiting in a queue of the channel more than this time then it is discarded. Measured in milliseconds. To disable set the parameter to 0.

- **pri=P** Sets priority level of the specified channel (0...16). Smaller values correspond to greater priority levels. Two special values are available: "-1" – sets the lowest priority, "-2" - deletes the prioritization from the logical channel.

- **[no]strict** Please see the "strict" option description in the "Qm option" section below.

- **pps=N** Sets the limit for the packets per second for the specified channel. To disable set the parameter to 0.

- **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 "-1" deletes the parameter:

  - 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. If **vlan** and **dot1p** parameters are specified in the same command then **vlan** is processed first.

qm chN clear

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

qm add[out] [num] [ifname] chN [pass] rule ...

Specifies one or more rules for accepting incoming packets at the channel **#N**. When used with optional parameter **out** (**addout**) it specifies the rules for outgoing packets. Rules are specified using the same syntax as in the **ipfw** command.

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

All rules specified on a device 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.

The optional **num** parameter may be used to explicitly specify the number of the new filter in the list.

Each packet arriving to the device 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 (if not using **pass** parameter). 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 [full] [clear] command displays statistics of the specific channel (only for channels with specified speed limitation). "Full" option allows viewing enhanced statistics. "Clear" option zeroizes the statistics.

The "Qm stat" command displays PPS (Packets Per Second) statistics only if the limit for the packets per second is set for the specified channel (qm chN pps=N).

qm del RULE_NUMBER

This command deletes the specified rule from the list.

qm RULE_A RULE_B

Change the number of the rule from A to B.

qm rearrange [N]

Renumbers all the rules with the given increment (default is 5).

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 dscp=31 all from a to b
qm add ch1 dscp=42
```

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".

"**Qm option**":

```
qm option {[-]rtp [-]dot1p [-]tos [-]icmp [-]tcpack [-]strict}
```

- allows automatic prioritization management of data flows in the device. Command options **[-]rtp [-]dot1p [-]tos [-]icmp [-]tcpack** enable/disable automatic prioritization of real time packets, packets labeled with IEEE 802.1p priority (below is a compliance scheme of MINT and IEEE 802.1p priorities), packets labeled with TOS, ICMP (Internet Control Message Protocol) packets, TCP ACK (acknowledgments) packets.
- **[-]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=5 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=5
qm ch2 pri=10
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 –rtp tos
```

This command disables real time packets 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 ch3 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
```

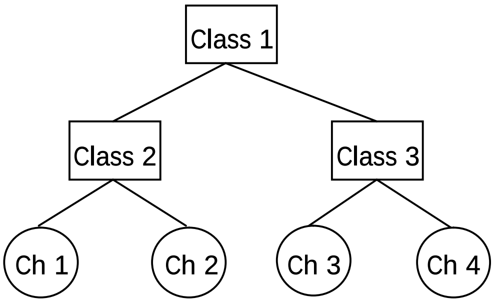
Packets which are coming from MINT network through eth0 interface and having DSCP label 11 is put into channel 25.

```
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
```

Example of using a hierarchy of service classes:



```
qm class1 max=1000
qm class2 max=600 ceil=1000
qm class3 max=300 ceil=1000 ceilprio=1
qm ch1 max=200 ceil=1000 class2
qm ch2 max=400 ceil=1000 class2
qm ch3 max=100 ceil=300 class3
qm ch4 max=200 ceil=300 class3
```

The result of these commands are a hierarchy of service classes (see figure) where channels (Ch1 and Ch2), members of the Class2, have a higher priority to use a bandwidth of 1000 kbps then channels (Ch3 and Ch4), since Class2 is of higher priority than Class3.

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. ARIP

Getting started

ARIP module is a realization of a standard routing protocol RIP.

ARIP routing module support two RIP *(Routing Information Protocol) versions* - **RIP-1** and **RIP-2**.

Module configuration is performed by **arip** command.

Command language. Basic principles

ARIP has its own command shell (CS). To start the ARIP module and enter the shell, execute the following commands:

```
#1> arip start

#1> arip

RIP>
```

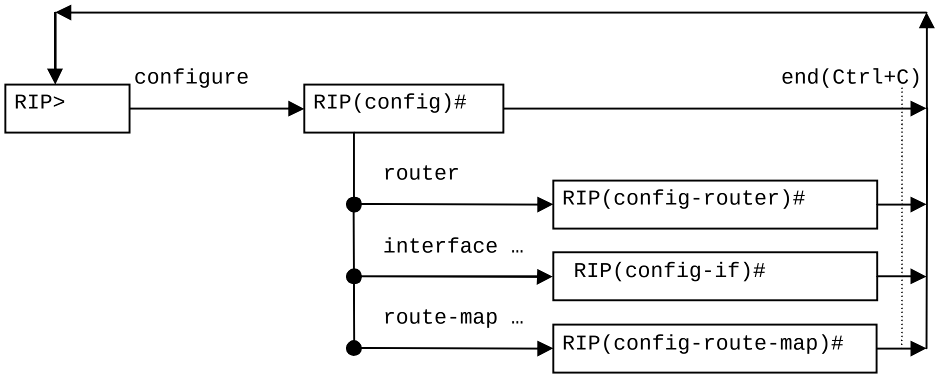
Commands entered in CS are not case-sensitive and can be shortened until ambiguity appears. To get a quick hint you can press “?” at any time:

```
RIP>?

  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

RIP>
```

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

```
RIP> configure
RIP(config)#
```

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



One can set the necessary mode or execute commands without specially entering into arip module. For example, if we consistently execute the following commands:

```
#1> arip configure
#1> arip router
#1> arip
RIP(config-router)#
```

While entering arip we will enter directly into necessary mode *config-router* (as it is shown in the example).

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 RIP 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:

```
RIP> config
RIP(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
  key          Authentication key management
  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
RIP(config)# interface?
  IFNAME  Interface's name
RIP(config)# interface eth0
RIP(config-if)#?
  authentication  Authentication control
  description     Interface specific description
  end             End current mode and change to root mode (CTRL+C).
  exit            Back to WANFleX command shell (CTRL+D).
  help            Print command list
  no              Negate a command or set its defaults
  receive         Advertisement reception
  send            Advertisement transmission
  show            Show running system information
  split-horizon   Perform split horizon
RIP(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 RIP

Start of the *RIP* router is executed by the following command:

```
arip start
```

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

```
stop (daemon|clear)
```

Example:

```
> arip
RIP> configure
RIP(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.

Filters

In many participating in the configuration parameters of the router 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 RIP 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 RIP 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 192.168.12.255, 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:

```
RIP(config)# access-list 1 permit 0.0.0.0 255.255.255.255
```

is equal to the command:

```
RIP(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:

```
RIP(config)# access-list 1 permit 192.168.12.150 0.0.0.0
```

is equal to the following command:

```
RIP(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:

```
RIP(config)# access-list TestList1 deny 192.168.1.0/24
RIP(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.

RIP configuration

The router can be enabled on the interface in several ways:

1. *By network specification.* RIP will be enabled on the interface with network address matching with the specified network. This can be performed by the following command in the *config-router* mode:

```
network A.B.C.D/M
```

Network is specified by its IP-prefix and mask.

2. *By interface name.* RIP will be enabled on the specified interface. This can be performed by the following command in the *config-router* mode:

```
network WORD
```

where **WORD** is interface name.

Example:

```
RIP>configure
RIP(config)# router
RIP(config-router)# network 4.7.8.0/24
RIP(config-router)# network  rf5.0
RIP(config-router)#
```

To cancel RIP on the interfacer use command:

```
no network A.B.C.D/M
no network WORD
```

In some cases not all routers understand multicast requests. To solve this problem, you can establish a direct link between routers. To implement this, use the command in *config-router* mode:

```
neighbor a.b.c.d
```

a.b.c.d – router's neighbor address. To cancel link between routers:

```
no neighbor a.b.c.d
```

To announce information from other routing protocols use the following command in *config-router* mode:

```
redistribute (kernel|connected|static|ospf) [metric <0-16777214>] [route-map WORD]
```

To define criteria according to which a router will announce information from some routing protocol, use the command in config-router mode:

```
distribute-list WORD direct ifname
```

where **WORD** – list name, **direct** – direction (values "**in**" or "**out**". When **direct** is "**in**" access list is adjusted to input packages, when "**out**" – to output packages). This command connects access list with the interface.

In the following example, the "eth0" allows only those packets that are routed to 10.0.0.0/8:

```
RIP(config-router)# distribute-list private in eth0
RIP(config-router)# access-list private permit 10 10.0.0.0/8
RIP(config-router)# access-list private deny any
```

Default metric is specified using the following command in the *config-router* mode:

```
default-metric <0-16>
```

If default metric is not defined, it equals 1.

In **redistribute kernel** mode the router will not make an advertisement into RIP system about having a *default route* (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 *default route* it is necessary to clearly force him to do that using a command in *config-router* mode:

```
default-information originate
```

To cancel advertising of the *default route* us the command:

```
no default-information originate
```

The following command enables "split horizon" algorithm at the device's ip interface in the *config-if* mode:

```
split-horizon [poisoned-reverse]
```

When the "split horizon" algorithm is enabled device doesn't announce routes through an interface from which they were obtained, thus reducing the likelihood of a local routing cycles.

If **poisoned-reverse** option is set device when removing the route still some time left it in the routing table and include it in the standard distribution announcement with special reference so that neighboring routers realize that the route is no longer used. Metrics of a route with the value 16 is used as a metrics for this.

"Split horizon" algorithm without **poisoned-reverse** option is enabled by default.

To cancel "split horizon" algorithm use command:

```
no split-horizon
```

Route map (route-map)

For more flexible configuration of metric type and its value, 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:

```
RIP> configure
RIP(config)# route-map testmap permit 10
RIP(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 belongs.

If a route matches to all record’s rules one can set values for route metric for this router using command in *config-route-map* mode:

```
set metric <0-4294967295>
```

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:

```
RIP> configure
RIP(config)# access-list AnyNetwork permit any
RIP(config)# access-list net200 permit 192.168.200.0/24
RIP(config)# route-map mapForConnected permit 10
RIP(config-route-map)# match address net200
RIP(config-route-map)# set metric 7
RIP(config-route-map)# route-map mapForConnected deny 11

RIP(config-route-map)# match address AnyNetwork
RIP(config-route-map)# router
RIP(config-router)# redistribute connected route-map mapForConnected
RIP(config-route-map)#
```

In this configuration the router will announce a route formed from the connected routes of the system routing table. With this, if a destination for this route is 192.168.200.0/24 network the formed route will have metric 7, any other destination will not lead to route announcing it.

Attention!!! For the interface to use the route-map which we have created before one have to use command **route-map** in the ***config-router*** mode:

```
route-map WORD (in|out) IFNAME
```

where **WORD** – name of the road-map which we have created before.

Authentication. Identity check

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

- **Password authentication.** 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.** Key is used while generation and check of message-digest signatures. Digital signature is built based on MD5 algorithm. 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 using the following commands in *config-if* mode:

1. Password authentication:

```
authentication mode text
authentication string LINE
```

where **LINE** – password, less than 16 symbols.

2. Cryptographic authentication:

```
authentication mode md5
authentication key-chain LINE
```

where **LINE** – name of the secret MD5 key

To configure the key which name is specified in **LINE** parameter use command in *config* mode:

```
RIP(config)# key chain WORD
RIP(config-keychain)# key <0-2147483647>
RIP(config-keychain-key)# key-string LINE
```

where

WORD – key chain name

<0-2147483647> - key ID

LINE – secret md5 key

Timers configuration

RIP protocol has several timers. User can configure those timers' values by timer's basic command. The default settings for the timers are as follows:

- The **update timer** is 30 seconds. Every update timer seconds, the RIP process is awakened to send an unsolicited Response message containing the complete routing table to all neighboring RIP routers.
- The **timeout timer** is 180 seconds. Upon expiration of the timeout, the route is no longer valid; however, it is retained in the routing table for a short time so that neighbors can be notified that the route has been dropped.
- The **garbage collect timer** is 120 seconds. Upon expiration of the garbagecollection timer, the route is finally removed from the routing table.

The following command in config-router mode allows the the default values of the timers listed above to be changed:

```
timers basic update timeout garbage
```

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

```
no timers basic
```

Router configuration view

To review RIP configuration there are several commands in the basic mode of CS:

```
show access-list
```

This command shows information about access lists.

```
show memory
```

This command shows information about memory usage.

```
show rip
```

This command shows current RIP configuration. Viewing Information about timers, filters, version, interfaces, on which RIP is enabled.

```
show route
```

This command lists route table.

6. 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 rf5.0 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 sub networks.

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 sub networks will appear aggregated in one routing path, and separated in another. Naturally, when choosing the route the most specific one (with the longest mask) will be used that could be undesirable.

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
```

7. 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>
```

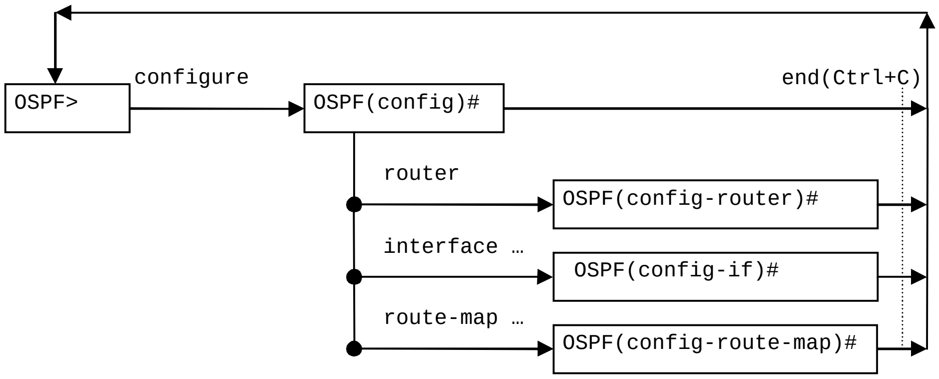
Commands entered in CS are not case-sensitive and can be shortened until ambiguity appears. To get a quick hint you can press "?" at any time:

```
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 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 destimation 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

The following command allows setting summary address for the external routes that are injected into the OSPF domain via redistribution:

ospf config router summary-address A.B.C.D/M [metric M] [metric-type T]

In case there is one or more external routes having network prefixes totally covered by the "summary-address" network (A.B.C.D/M) then only the summary route for the A.B.C.D/M network will be announced but not the external routes. Coupled with the accurate address space management this command allows to reduce the number of external network prefixes in the OSPF domain significantly.

To view the list of the possible summary networks and a number of the network prefixes that can be replaced with a summary route use the following command:

ospf show summary-address [detail]

“Detail” option enables detailed output of the command.

This command may be used as a tip while planning the network.

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.

o **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 links which were announced by transit network, and the advertising router 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
rf5.0 is up
  OSPF not enabled on this interface
OSPF>
```

8. Netstat command (Network statistics)

Display the network statistics

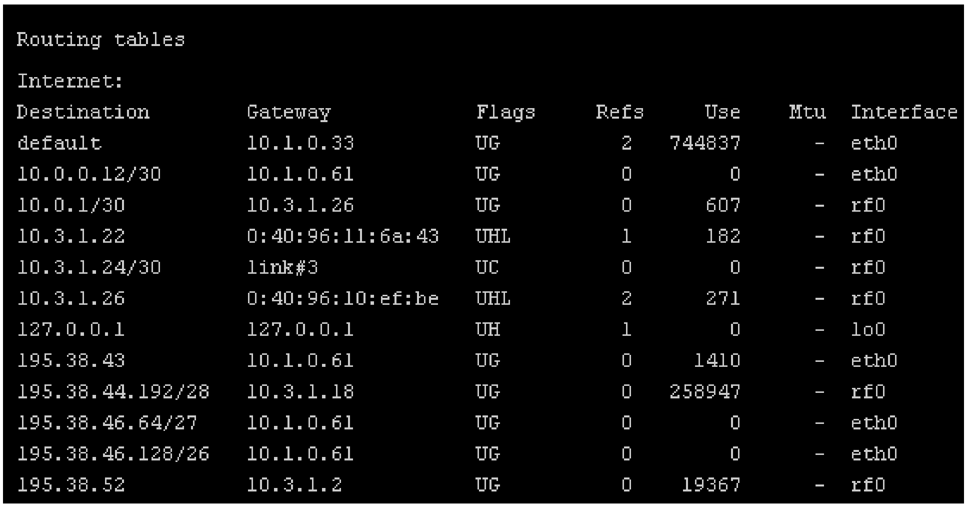
Syntax:

```
netstat -r
netstat -i
```

Description:

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

“**-r**” parameter displays system routing tables:



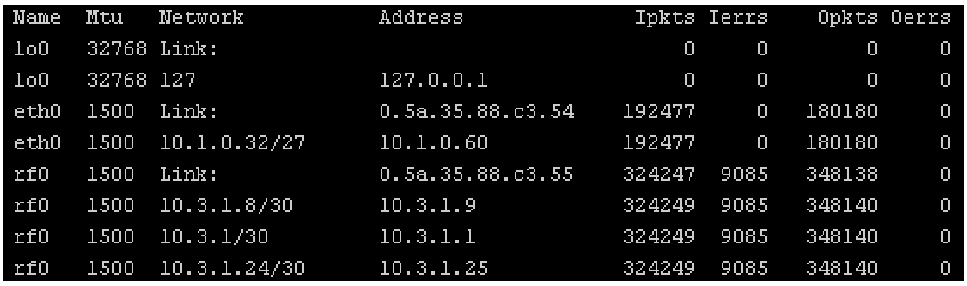
```
Routing tables

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
```

9. Ipfw command (IP Firewall)

General description

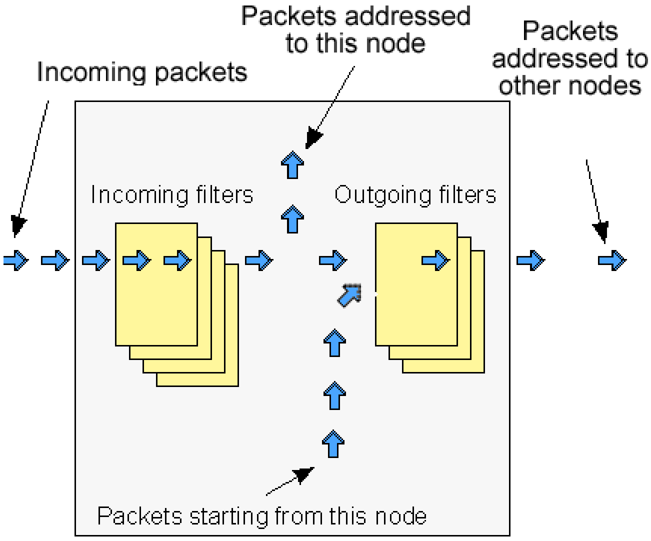
IP Firewall is a mechanism of filtering packets crossing an IP network node, according to different criteria. System administrator may define a set of incoming filters (**addincoming**) and a set of outgoing filters (**addoutgoing**). The incoming filters determine which packets may be accepted by the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


Incoming packets
Packets addressed to this node
Packets addressed to other nodes
Incoming filters
Outgoing filters
Packets starting from this node

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
show | reset
rearrange [N]
flush
quiet | -quiet
del num
mov num1 num2
add[out] [NUM] [IFNAME] rules...

rules: [{setpri|addpri}=[N]] accept|reject|rpfilter|pass [log]
```

[vlan={N|any|$ACL}] [dot1p=N] [swg=N] [ether={X|any}] [dscp=N|tos=N] [prf]

-f "pcap filter expression"

|

PROTO from [not] ADDR [PORTs] to [not] ADDR [PORTs]

PROTO: [all] | tcp | udp | icmp | arp | proto NUMBER

ADDR: IP | $LOCAL | $ROUTE | $ACL | mac {x:x:x:x:x:x}

PORTS: NUM[:NUM] [NUM] ...

Description:

ipfw list

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

ipfw show / reset

This command shows “ipfw” rules/resets “ipfw” rules counters.

ipfw flush

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

ipfw add [num] . . .

ipfw addout [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.

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.

ipfw mov num1 num2

Changes the filter’s number in the list: from num1 to num2.

ipfw rearrange [N]

Renumbers all the filter rules with the given increment (default is 5).

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:

[{setpri|addpri}=[N]] accept|reject|rpfilter|pass [log]

[vlan={N|any|$ACL}] [dot1p=N] [swg=N] [ether={X|any}] [dscp=N|tos=N] [prf]

-f "pcap filter expression"

|

PROTO from [not] ADDR [PORTs] to [not] ADDR [PORTs]

PROTO: [all] | tcp | udp | icmp | arp | proto NUMBER

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 rf5.0 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.

Setpri/addpri parameters allow setting/increasing priority for a packet when a packet is treated by the filter. “**Setpri**" parameter is used to change a priority to the value specified in the command. If empty value is used (**setpri=**) a package priority is dropped to the lowest priority. "**Addpri**" is used to change a priority only in case it is higher than the previous one (Note: the smaller is the value the higher is the priority). So you can only increase priority using "**addpri**" parameter.

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.

Module "**ipfw**" added with filter "**rpfilter**" (reverse path filter). This filter ensures that the sender of the package is accessible via the interface through which package it received in the system. If the filter fails, the packet processing continues, if not fails the packet is destroyed. This filter can be inserted into the list of rules first:

```
ipfw add rpfilter all from 0/0 to 0/0
```

One more possible value for **disp** field is “**pass**”. This value allows a packet to pass a rule executing the related actions of this rule and continue with other rules in the list.

Example:

```
ipfw add pass log tcp from 0/0 to 0/0
```

When a packet will face this rule it will continue moving further with other rules. Information about the packet will be logged.

Parameters **[vlan={N|any|$ACL}] [dot1p=N] [swg=N] [ether={X|any}] [dscp=N|tos=N] [prf]** are classifiers that allows analyzing VLAN ID, 802.1p priority, switch group number (SWitchGroup), packet type (EtherType) and also **ip_tos** field for having DSCP label value or IP precedence. **Prf** classifier enables filtration of the traffic outgoing from PRF interfaces.

One can specify several VLAN IDs using **$ACL** list (see ACL command description).

Any option of the **vlan** and **ether** classifiers enables the filter to accept all tagged packets regardless of their VLAN ID or packets of any EtherType correspondingly.

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.

Parameter **–f** allows using «pcap» filters.

Example:

```
ipfw add reject -f "icmp and host (1.1.1.1 or 1.1.1.5)"
```

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 inner host 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 rf5.0 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 rf5.0 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 rf5.0 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 rf5.0 reject all from 127.0.0.0/8 to 0/0
```

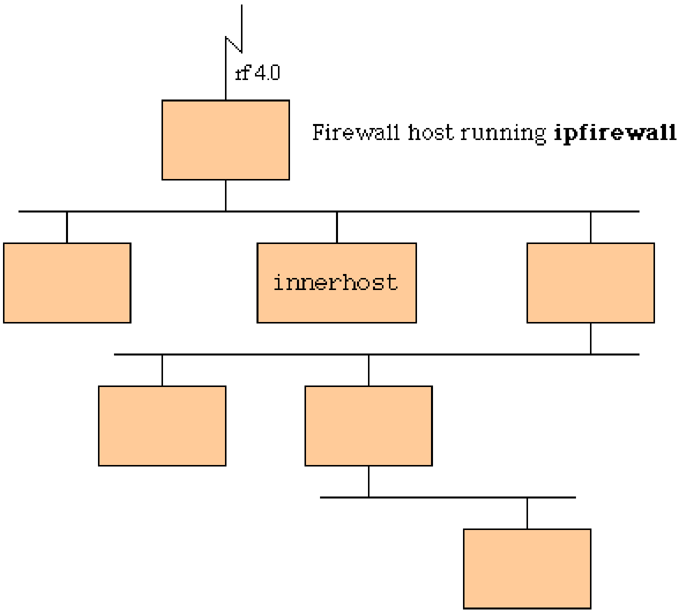
IP spoofing has been widely used in the Internet as an aggression method. For additional information, see CERT summary CS-95:01, and also summaries on the CERT WWW site.

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 rf5.0 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 rf5.0 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 rf5.0 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 rf5.0 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 **rf5.0** interface):

```
ipfw add rf5.0 accept tcp from 0/0 to 0/0 900:5000
ipfw add rf5.0 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 rf5.0 reject tcp_connection from 0/0 to 0/0 900:5000
ipfw add rf5.0 accept tcp from 0/0 to 0/0 900:5000
ipfw add rf5.0 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 rf5.0 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 rf5.0 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 rf5.0 reject tcp from 0/0 to 0/0 135:139
ipfirewall add rf5.0 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).*

10. Loadm command (load meter)

This is a tool to perform the channel load monitoring

Syntax:

```
loadm [-b] [-m] [-l] [-p] [-w delay] 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;
- **-m:** display results in Megabits or Megabytes (with enabled "-b" option) per second;
- **-l:** display information line by line;
- **-p:** calculates average packet size;
- **-w delay:** specifies time interval between updates.

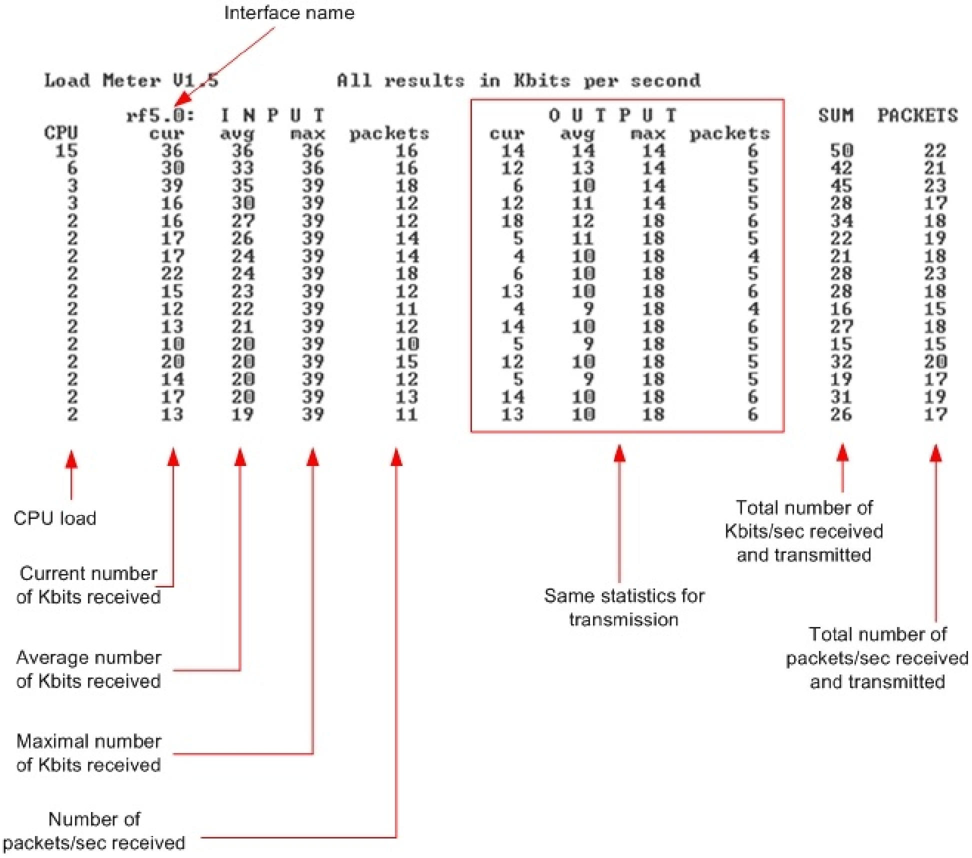
⚠ *For the rf5.0 radio interface the loadm command shows the total channel load, and not only that part of it which corresponds to the given device. On a client unit, for example, not only that unit's own traffic appears on its rf5.0 interface and is taken into account by the loadm command, but also 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).*

Therefore, the real load of the unit is better to see on its ***eth0*** *interface.*

Example:

```
loadm -l rf5.0
```

The output example is shown below.



```
Load Meter V1.5             All results in Kbits per second
        rf5.0:  I N P U T                     O U T P U T               SUM   PACKETS
CPU        cur   avg   max  packets      cur   avg   max  packets
 15         36    36    36       16       14    14    14        6      50        22
  6         30    33    36       16       12    13    14        5      42        21
  3         39    35    39       18        6    10    14        5      45        23
  3         16    30    39       12       12    11    14        5      28        17
  2         16    27    39       12       18    12    18        6      34        18
  2         17    26    39       14        5    11    18        5      22        19
  2         17    24    39       14        4    10    18        4      21        18
  2         22    24    39       18        6    10    18        5      28        23
  2         15    23    39       12       13    10    18        6      28        18
  2         12    22    39       11        4     9    18        4      16        15
  2         13    21    39       12       14    10    18        6      27        18
  2         10    20    39       10        5     9    18        5      15        15
  2         20    20    39       15       12    10    18        5      32        20
  2         14    20    39       12        5     9    18        5      19        17
  2         17    20    39       13       14    10    18        6      31        19
  2         13    19    39       11       13    10    18        6      26        17
```

Interface name
CPU load
Current number of Kbits received
Average number of Kbits received
Maximal number of Kbits received
Number of packets/sec received
Same statistics for transmission
Total number of Kbits/sec received and transmitted
Total number of packets/sec received and transmitted

11. Bpf command (Berkeley Packet Filter)

The command enables packet capturing regime (Berkeley Packet Filter)

Syntax

bpf ifname IP_addr port

bpf ifname –

bpf -f "pcap filter expression"

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
```

bpf –f – allows to set `pcap filter.`

Example:

```
bpf rf5.0 10.11.12.13 8000
```

Enables packet capturing regime, sending all packets from the rf5.0 interface to a workstation at the address **10.11.12.13.**

```
bpf rf5.0 -
```

Disables packet capturing regime at the **rf5.0** interface.

12. RPCAP (Remote Packet Capture)

RPCAP (Remote Packet Capture) system provides the ability to remotely capture the packets being passed over the network allowing the remote control and analysis of the transit data flows.

RPCAP system consists of a server side daemon and a client side application. The client application (for example, packet analyzer) connects to the server daemon, gives instructions which packets should be captured and manages the whole process. The server daemon sniffs the network traffic, captures the requested packets and passes them to the client side that process and analyze the captured packets.

The IW device supports RPCAP protocol and has a built-in RPCAP server daemon. It can be enabled and configured using the “rpcapd” command described below.

Syntax:

```
rpcapd -user=USERNAME -key[=PASSWORD] [add|del|change]
rpcapd [-port[=PORT]] [-maxconn[=MAXCONNECTIONS]] [start|stop]
rpcapd [-buffersize=[SND_BUFFER_SIZE]]
rpcapd {trace|notrace}
rpcapd show [-s=SOURCENAME]
rpcapd clear
```

Description:

To start/stop the RPCAP server daemon use the following command:

```
rpcapd [-port[=PORT]] [-maxconn[=MAXCONNECTIONS]] [start|stop]
```

When started without any arguments (**rpcapd start**) it sets the default RPCAP port value (2002) and unlimited number of allowed client connections. To specify another values “port” and “maxconn” parameters are used.

The following command adds/deletes/changes a username and password to be used by the client to connect to the RPCAP server:

```
rpcapd -user=USERNAME -key[=PASSWORD] [add|del|change]
```

When used without specifying the action (i.e. without “add”, “del” and “change”) the command adds a new user or changes the existing user with the same “username”.

> If no user is configured in the system the RPCAP server daemon will reject any connections.
>
> For using Null Authentication scheme you should add a user with empty “user” and “key” parameters: “rpcapd –user= -key=”.

The following command allows specifying the internal buffer size of the daemon for sending the captured packets to the client application:

```
rpcapd [-buffersize=[SND_BUFFER_SIZE]]
```

The default buffer size is 32Kb.

The following command enables/disables writing daemon debug output to the unit’s system log:

```
rpcapd {trace|notrace}
```

The following command allows viewing all the currently active connections:

```
rpcapd show [-s=SOURCENAME]
```

Parameter “-s” allows viewing the BPF filter of the connection with the specified device’s interface name (SOURCENAME).

To clear the configuration and stop the daemon use the following command:

```
rpcapd clear
```

13.Snmpd command (SNMP daemon)

SNMP protocol version 1 and 3 daemon

Syntax:

```
snmpd user NAME (add|set) [pass PASSWORD] [sec[urity] (noAuthNoPriv|authNoPriv|authPriv)] [acc[essRights] (readOnly|readWrite)] [cla[ss] (guest|admin)] [privpass PRIVPASS] [proto (des|aes128)]

snmpd user NAME del[ete]

snmpd comm[unity] NAME

snmpd (nodebug|debug [prox] [trap] [stat] [mibs] [user] [cryp] [pack] [time] [flow])

snmpd (v1disable|v1enable)
snmpd (start|stop)
snmpd clear
```

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 are 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.

Support of SNMP-V3 with USM (User-based Security Model), MD5 authentication and encryption are also available. For access granting, a user with username, access passwords and access rights (with or without authentication and encryption) is created.

In “snmpd” command **accessRights** can be set to provide access management of the recourses. **ReadOnly|readWrite** parameters allow only reading or also changing some variables. **Class guest/admin** allows providing limited or full access to the variables.

The default SNMP v1 community name for read operations is "public". The “**snmpd community NAME**” command allows changing the default community name.

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

The following SNMP security options can be used by setting the “**sec[urity]**” parameter:

- **noAuthNoPriv** – SNMP messages are sent unauthenticated and unencrypted.
- **authNoPriv** – SNMP messages are sent authenticated and unencrypted.
- **authPriv** – SNMP messages are sent authenticated and encrypted. The passphrase for the encryption is set by the “**privpass PRIVPASS**” parameter. The protocol is set by the “**proto (des|aes128)**” parameter.

The “**nodebug/debug**” options disables/enables printing of SNMP service information into the system log.

The “**snmpd clear**” command deletes SNMP configuration on the unit.

Example:

```
snmpd comm secret
snmpd user john add pass mypassword security authNoPriv
snmpd on
```

14.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 Telnet connection establishment requests from any host in the network. After executing one or several **td enable RemoteHOST** commands, remote Telnet 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
```

15. 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 is possible to identify the sender when IP packet is 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.

As it's known (rfc1918), some part of the 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] [-exclude $DSTACL] [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 are considered as incoming and, if they are not reserved 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.

It is also possible to receive a public IP address from a DHCP server. For example, if the provider haven't provided you with a public IP address, and you have to obtain it automatically. In this case set up DHCP Client on an interface of the router by using the **dhcp IFNAME** option. For example:

```
nat local_acl $NAT dhcp eth0
```

You can also use **-exclude $DSTACL** option to specify a list destination IP-addresses/networks to be excluded from NAT.

In order to **enable/disable/delete** a record for the private network in the configuration, use the corresponding parameters from the command syntax. For example:

```
nat local_acl $NAT delete
```

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 into 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 are 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 flavors.

First type:

```
redirect_port  proto local_addr:local_port_range
               [public_addr:]public_port_range
               [remote_addr[:remote_port_range]]
```

This command redirects 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

local_addr:local_port_range – IP-address and port of the local host which the traffic will be redirected to.

[public_addr:]public_port_range – public IP-address and port of the device.

The port ranges **local_port_range** and **public_port_range** should be of the same size.

If you are using several pairs of public address-private network, it is recommended to specify the exact public address.

Parameters **remote_addr** and **remote_port_range** may be specified for more exact definition of incoming packets (packets only from specified source and port will be allowed). If **remote_port_range** is not specified then its range should coincide with range of **public_port_range**.

```
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 public_port_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 local_addr_1:local_port_range[,
local_addr_2:local_port_range, ...]
[public_addr:]public_port_range
[remote_addr[:remote_port_range]]

Cyclic redirection of incoming packets to several destination addresses for uniform 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_addr [,local_addr,...] public_addr

Redirects all incoming traffic directed to **public_addr** to **local_addr**. If several **local_addr** 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 local_addr [public_addr [remote_addr]]

Redirects all the incoming packets with specified protocol **proto** to the host with address **local_addr**.

nat default_h323 [yes|no]

Includes address modification according to H.323 stack for outgoing connections. Affects all incoming UDP packets 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]...

This command performs redirection of outgoing packets. TCP packets outgoing from LAN to any address with specified port, redirected to specified server and port. Optionally initial destination address may be included into the packet using several ways. Command line consists of word pairs: key parameters 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 specific 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 gatekeeper. Since static subscribers also should work with the gatekeeper, 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.

16. SNMP Traps support (“trapd” command)

SNMP protocol allows a network agent to send asynchronous traps when some specific event occurs on the controlled device (object).

InfiNet Wireless devices has built-in SNMP Traps support module (agent) that performs a centralized information delivery from internal device subsystems to the configured SNMP server. SNMP Traps agent is managed using the “**trapd**” command.

Syntax:

```
trapd dst[addr] x.x.x.x[:PORT] [[GROUPNAME] ...] [[[-]TYPENAME] ...]

trapd –dst[addr] x.x.x.x[:PORT]

trapd map

trapd agent x.x.x.x
trapd -agent
trapd gateway {xxxxxxxxxxxx|auto}
trapd -gateway
trapd type TYPENAME enable|disable
trapd start|stop
  where PORT default value is 162 if omitted
  possible GROUPNAMEs are:
    topoGroup
    mintGroup
    cmxGroup
    ospfGroup
  and possible TYPENAMEs are:
    topoEvent
    newNeighborEvent
    lostNeighborEvent
    mintRetries
    mintBitrate
    mintSignalLevel
    cesDsx1LineStatus
    cesJitterStatus
    ospfNBRState
    ospfVirtNBRState
    ospfIFState
    ospfVirtIFState
    ospfConfigError
    linkEvent
    trapdColdStartEvent
    snmpdAuthenticationFailureEvent
    syslog
```

Description:

To start/stop SNMP Trap agent on the device please use the following command:

```
trapd start|stop
```

The following command adds an SNMP server address to the SNMP configuration:

```
trapd dst[addr] x.x.x.x[:PORT] [[GROUPNAME] ...] [[[-]TYPENAME] ...]
```

Where, “x.x.x.x” is the server’s IP-address. The optional parameter “:PORT” defines the server’s UDP port (port 162 is used by default). The optional parameter “GROUPNAME” defines the groups of traps to be sent to the server. The optional parameter “[-]TYPENAME” allows to define or exclude exact traps to be sent to the server.

Multiple SNMP servers can be set in the configuration.

Example:

```
trapd dst 192.168.1.1
trapd dst 192.168.1.100
```

The above commands will enable sending SNMP traps from the device agent to the SNMP servers with IP-addresses 192.168.1.1 and 192.168.1.100.

To delete an SNMP server IP-address from the configuration use "-" sign as shown in the command syntax.

Example:

```
trapd -dst 192.168.1.1
```

You can view the allocated SNMP servers (their IP-addresses) and the defined traps/groups for each server by using the following command:

```
trapd map
```

Agent's own IP-address, which is set in SNMP-trap packet, is defined by the following command:

```
trapd agent x.x.x.x
```

To delete the agent's IP-address from the configuration use "-" sign as shown in the command syntax. Default agent's own IP-address is 127.0.0.1.

SNMP traps can be sent to the SNMP server via SNMP network agent running on some other device (gateway). To direct SNMP traps to the gateway the following command should be used:

```
 trapd gateway xxxxxxxxxxxx|auto
```

Where: "xxxxxxxxxxxx" is the gateways' MAC-address. When parameter "auto" is specified instead of the gateways' MAC-address, SNMP traps are automatically sent to the MINT SNMP relay, defined by the "mint -snmprelay" command, if it exists in the MINT network. To delete the gateway from the configuration use "-" sign as shown in the command syntax.

All these traps can be allowed or rejected for sending by the following command:

```
trapd type TYPENAME enable|disable
```

Where: "TYPENAME" is the trap's name. By default, all traps are disabled.

Example:

```
trapd dst 192.168.1.1
trapd type newNeighborEvent enable
trapd start
```

The following SNMP traps and trap groups are supported by the system:

ColdStartEvent – trap is sent when the units is powered up

AuthenticationFailureEvent – trap is sent in case of failure in SNMP authentification

Syslog – all new messages in the system log are sent as a trap

TopoGroup

- **topoEvent** – full neighbor list is sent in case of any changes in number of neighbors or their status
- **newNeighborEvent –** trap is sent when a new neighbor appears
- **lostNeighborEvent** - trap is sent when the neighbor is lost

MintGroup

- **mintBitrate** - trap is sent when the bitrate changes
- **mintRetries** - trap is sent Retries has changed by more than 10%
- **mintSignalLevel** - trap is sent Signal Level has changed by more than 10%

cmxGroup

- **cesDsx1LineStatus** - trap is sent when the CES line status changes
- **cesJitterStatus** - trap is sent the CES jitter buffer size changes

ospfGroup

- **ospfNBRState** - trap is sent when there happens a change in the state of a non-virtual OSPF neighbor.
- **ospfVirtNBRState** - trap is sent when there happens a change in the state of a virtual OSPF neighbor.
- **ospfIFState** - trap is sent when there happens a change in the state of a non-virtual OSPF interface.
- **ospfVirtIFState** - trap is sent when there happens a change in the state of a virtual OSPF interface.
- **ospfConfigError** - trap is sent when a packet has been received on an interface from a router whose configuration conflicts with this router's configuration.

linkEvent – this trap is sent when Ethernet state is changing. This trap enables/disables sending two generic SNMP traps: **linkUp** and **linkDown**.

17.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 dscope <SCOPE_NAME> <INTERFACE|*> <START_IP> <END_IP>
add virtual interface <VIFNAME> <GATEWAY> <GWIFNAME|*>
clear
delete option <OPTION_NAME>
delete scope <SCOPE_NAME>
delete virtual interface <VIFNAME>
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 <VIFNAME> add subnet <IP_ADDRESS> <SUBNET_MASK>
virtual interface <VIFNAME> change <GATEWAY> <GWIFNAME|*>
virtual interface <VIFNAME> 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 the 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.

To create dynamic scope you can use the following command:

dhcpd add dscope <SCOPE_NAME> <INTERFACE|> <START_IP> <END_IP>*

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 (and *)
[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 who 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 displayed 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, who 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:

```
add virtual interface <VIFNAME> <GATEWAY> <GWIFNAME|*>
```

Where

- **VIFNAME** – name of the DHCP server's virtual interface
- **GATEWAY** – IP-address of DRA which has a corresponding physical interface.
- **GWIFNAME** – name of DRA's interface that performs retranslation of DHCP packets to DHCP clients. If the "*****" symbol is used as "GWIFNAME" parameter DRA is allowed to use all its interfaces to retranslate DHCP packets.

  After executing this command, one more interface is created in DHCP server's configuration.

Example:

```
#1> dhcpd add virtual interface vvv1  192.168.177.81 *
#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
[vvv1] 192.168.177.81:* UP
```

In server's configuration we now can observe the virtual interface. 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 <VIFNAME> add subnet <IP_ADDRESS> <SUBNET_MASK>
```

Where

- **VIFNAME** – name of the DHCP server's virtual interface
- ***IP_ADDRESS*** – IP-address which DRA has for this subnet
- ***SUBNET_MASK*** – subnet mask

Example:

```
#1> dhcpd virtual interface vvv1
      add subnet 192.168.188.1 255.255.255.0
#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
[vvv1] 192.168.177.81:* UP
 <SUBNET> 192.168.188.1/255.255.255.0
```

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 to.

Example:

```
#1> dhcpd add scope VIRTUAL_TEST
    vvv1 192.168.188.20 192.168.188.50
#1> dhcpd show interface vvv1
>VIRTUAL INTERFACES
[vvv1] 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
#1> dhcpd show scope virtual_test
>SCOPES:
(VIRTUAL_TEST) 192.168.188.20 - 192.168.188.50 [vvv1] ATTACHED [vvv1]
<192.168.188.1>/255.255.255.0 <FREE RANGE> 192.168.188.20 -
192.168.188.50   =31
```

You can delete a subnet from the virtual interface's list using the following command:

Syntax:

```
dhcpd virtual interface <VIFNAME> delete subnet <IP_ADDRESS>
<SUBNET_MASK>
```

Example:

```
#1> dhcpd virtual interface vvv1 delete subnet 192.168.188.1
255.255.255.0
#1> dhcpd show scope virtual_test
>SCOPES:
(VIRTUAL_TEST)       192.168.188.20  - 192.168.188.50  [vvv1]
```

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 the "vvv1" virtual interface.

DHCP server configuration cleanup

In order to clean DHCP server configuration, it first should be stopped by "*dhcpd stop*" command. After that, the configuration can be cleaned:

```
dhcpd clear
```

18. 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. 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
```

Logging

Syntax:

```
dhcpr trace|notrace
```

This command enables/disables wiring DHCP Relay's service messages to the system log.

Resetting

Syntax:

```
dhcpr flush
```

This command clears the DHCP Relay's configurations.

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
```

19.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  rf5.0 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 rf5.0 interface *none* option is set for the number of ARP requests. Thus, rf5.0 will send 16 ARP requests. Also, DHCP client on rf5.0 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  rf5.0  ------------------ ---------------
```

Here, clients are started on eth0 and rf5.0 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 rf5.0 interface has not yet received any parameters.

20.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
vrrp dump IFNAME:VRID
vrrp IFNAME:VRID [start|stop|clean|flush]
vrrp IFNAME:VRID [add]|delete IPADDRESS[/(MASK|MASKLEN)] ...
vrrp IFNAME:VRID [-priority=[PRIO|own]] [-interval=AINT]
                 [-(password|key)=[PASSWORD]]
                 [-preempt=(on|off)] [-owner=[on|off]] [-learn=(on|off)]
                 [-track=(off|default|IPADDRESS/MASKLEN)]
```

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
```

Network Prefix Monitoring mode

Syntax:

```
vrrp IFNAME:VRID -track=(off|default|IPADDRESS/MASKLEN)
```

If Network Prefix Monitoring mode is enabled the VRRP module checks the availability of a route to the specified IP network (IPADDRESS/MASKLEN), or the default route (default). If the routing entries disappear from the system tables the device enters the BACKUP mode.

To disable the Network Prefix Monitoring mode use the “off” option.

Example:

```
vrrp eth0:10 –track=default
```

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:
    - o *MASTER*
    - o *BACKUP*
    - o *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:
    - o *Configuration conflict* – This situation occurs if different VR's with the same interface have the crossovering IP-addresses lists.
    - o *IP Address list is empty* – no IP-addresses are specified.

o *Interface has no primary IP address* – interface does not have primary IP-address or it has been deleted.

o *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.

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 signaling | temperature
```

- **temperature** – manages the executive device automatically according to the temperature sensor data
- **signaling** – 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 value **lo**w value **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:

  o if the temperature is higher or equal with **high**, the executive device turns on

  o 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:

  o if the temperature is less than **low**, the executive device is turned on

  o 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 sign**aling [**en**able | **di**sable | **ma**niac] [**op**ened | **cl**osed] [**lo**g | **nolo**g] [**bt** value] [**fp** | **nof**p] [**tr**aps 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
- **opened –** 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.