BalanceNG® V5

The Software Load Balancer

User and Reference Manual

Command Set BalanceNG V5 (5.016)

SecurITy made in Germany

TeleTrusT Quality Seal
www.teletrust.de/itsmig

Status: **STABLE**
Version: 5.016.0
Date: Nov 8, 2022
Author: Thomas G. Obermair
SW-Date: 2022/11/08

Inlab Networks GmbH
Josef-Würth-Str. 3
82031 Grünwald
Germany

Tel.: +49 89 64911420
Fax: +49 89 64911421
Email: office@inlab.net
WWW: http://www.inlab.net

Table of Contents

Legal Notices

1 Introduction

1.1 What is BalanceNG ?

BalanceNG (“Balance Next Generation”) is a Software Load Balancing Solution utilizing its own network stacks and functionality. In fact, BalanceNG uses the underlying operating system only for accessing the physical interfaces, all TCP/IP and other functionality (like ARP and ICMP) is being processed internally.

BalanceNG runs as a user mode program using the PF_PACKET API on Linux and the BPF API on macOS to access the network interfaces as directly as possible.

With BalanceNG the Network or Data center Administrator is capable to build high availability capable load balancing devices at a very low and very competitive price (compared to dedicated hardware boxes / Load Balancing Appliances).

1.2 BalanceNG Features and Specifications

- Layer 2 (Ethernet) based load balancer.
- Available for Linux-x86 and macOS.
- Linux distribution independent.
- Capable to run in multiple instances on the same host.
- Session persistence based on client address and optional source port.
- Backup targets (hosts) specifiable in case of failure of all primary targets.
- Health checking via: PING, TCP Socket Open and freely customizable UDP Health Check Agent (supplied in Source-Code).
- External target specific health check scripts.
- Alert/Upalert notification scripts (e.g. for sending email or sending a SNMP trap to a network management system).
- Distribution methods: Round Robin, Simple Weighted Round Robin, Random, Weighted Random, Client Address Hashing, First Operational, Least Session, Least Bandwidth and Least Resource based on agent supplied information.
- Unchanged client addresses on IP-level.
- Supports DSR (Direct Server Return) configurations.
- Small, very fast and reliable.
- Simple to implement and administer.
- Simple "init script style" arguments like "start", "stop" and "status" (and "control" for interactive configuration and control).
- Interactive communications mode with command line editing.
- Pcap packet dumping with automated dumpfile rotation (e.g. to implement a "transparent forensic logging bridge").
- Multi-node High Availability capability using standard VRRP (Virtual Router Redundancy Protocol).
- Session table synchronization and connection state replication using a BalanceNG-specific VRRP extension.
- "All service load balancing" based on client IP address enables most protocols to be supported (e.g. active FTP, RTSP/RTP/RTCP streaming protocols etc).

- SNMP Support integrating into Net-SNMP.
- Layer 3 link load balancing to a set of outbound routers or ISP links.
- Supports up to 512 virtual servers and up to 1024 targets (real servers).
- Location-Based Load-Balancing Support.
- In-memory IP-to-Location database.
- DNS-based GSLB-support (Global Server Load-Balancing).

1.3 Hardware and OS Requirements

1.3.1 Linux

BalanceNG runs under most x86 Linux implementations using Kernel revisions 2.6 or 3.0. BalanceNG is distribution independent and should run on any distribution supporting 2.6 / 3.0 kernels.

Known distributions which support BalanceNG are:

- Ubuntu Linux
- Gentoo Linux
- Red Hat Enterprise Linux (RHEL)
- Fedora Linux
- CentOS
- Debian
- Slackware Linux
- SUSE Linux (Novell)

All network adapter cards which are either compiled into the kernel or available as a module are supported.

There is no package manager dependency as BalanceNG comprises a single, statically linked binary that can be installed manually anywhere on the host system

Memory recommendation: 128 Megabytes minimum, 512 Megabytes recommended.

Processor minimum requirement: 1Ghz for 2 x 1000BaseT interfaces and a 100BaseT management network

1.3.2 macOS X

BalanceNG is supported on the following macOS X platforms:

- macOS 10
- macOS 11

1.4 BalanceNG Core Concepts

In order to understand the simplicity of BalanceNG it's important to know about the BalanceNG core concepts, here the basics:

1.4.1 Interfaces

Interfaces are the physical hardware interfaces to one or more networks. They are named like the underlaying Linux kernel names them (like eth0, eth1 and so on).

BalanceNG uses the interfaces that it is allowed to use, this is done by specifying a corresponding interface section.

Interfaces don't have to be "up" or configured, BalanceNG performs all necessary administrative tasks automatically. Also it is neither required or necessary to configure interface addresses in the Linux operating system.

1.4.2 Networks

Networks are IPv4 network definitions the have a network address and a network mask. Additionally one or more interfaces are being referenced by the BalanceNG network definition.

On a UNIX system an interface has one or more associated network definitions and addresses, in the BalanceNG world this relation is reversed: One network definition (and the addresses) are associated to one or many interfaces.

Each network definition additionally has to have two required IPv4 addresses: The "real" address being used for ARP-requests and health checks and the "virt" address being addressable as a routing endpoint for external devices. The "real" network address has to be node specific, the "virt" address has to be shared between multiple BalanceNG nodes in a VRRP HA configuration.

1.4.3 Servers

Servers are the addressable "virtual Servers" in the BalanceNG world. Servers are "virtual" or "artificial" IP addresses represented by BalanceNG. Network requests to those servers are distributed among the targets according to the load balancing definitions.

Servers may be defined in any BalanceNG network referencing targets in any BalanceNG network.

Note: BalanceNG servers would be called "virtual servers" by other load balancing software vendors.

1.4.4 Targets

BalanceNG represents one or more virtual servers and distributes the requests among a set of targets associated with each virtual server.

Note: BalanceNG targets would be called "real servers" by other load balancing software vendors.

1.4.5 Modules

BalanceNG implements several packet handling modules. The functionality of BalanceNG is defined by the module chain, which defines a sequential order of modules. Each packet enters that module chain at the left side and is forwarded until a module gains responsibility for that packet. After some processing, the module in charge may decide to stop processing or may decide to forward a possibly changed packet to the next module in the module chain.

1.4.6 Threads

BalanceNG operated multi-threaded when the multi-threading packet scheduler is active. One BalanceNG interface may be operated by 1 to 8 simultaneous threads.

1.4.7 Configuration and Configuration Files

The behavior and actions of BalanceNG are controlled by its internal configuration. This configuration may be altered in interactive mode by entering configuration commands. An external representation of this configuration may be saved to /etc/bng.conf, BalanceNG loads an existing configuration in /etc/bng.conf automatically at startup (default instance 0).

A BalanceNG configuration consists of the following sections in exactly that order:

1. hostname, remark, license
2. module chain definition
3. parameter settings ("set"-section)
4. interfaces section
5. interfaces register/enable section
6. vrrp-section
7. network definitions
8. network register/enable section
9. IPDB section
10. lgrp (Location Group) section
11. gateway section
12. server definitions
13. server register/enable section
14. target section
15. target register/enable section

BalanceNG makes use of the following configuration files:

/etc/bng.global	Global configuration file for all instances
/etc/bng.conf	Standard configuration file for BalanceNG default instance 0
/etc/bngN.conf	Configuration file for BalanceNG instance N (N: 1 ... 127)
/etc/bng.private	Node specific private data for BalanceNG default instance 0
/etc/bngN.private	Node specific private data for BalanceNG instance N (N: 1 ... 127)

Note: It's safe to copy the main configuration file (/etc/bng.conf) from a master node to the backup if the node private data has been saved on the other side before (“save private”). This allows easy implementation of configuration synchronization scripts between nodes of the same VRRP virtual router.

1.4.8 Instances

BalanceNG may be started independently multiple times on the same host machine (node). Each invocation is called an instance of BalanceNG and has an unique instance number in the range of 0 ... 127. BalanceNG instance 0 is called the default instance.

1.4.9 IPDB, Locations and Location-Groups

BalanceNG supports a very efficient, in-memory IP-to-Location database (IPDB). This database associates ranges in the Ipv4 address space to a set of locations, which are usually 2-Letter codes as “US”, “DE” and “AT”, for example. These location may be logically grouped using the BalanceNG “location groups” (LGRP's). Eventually, a target may be a member of exactly one location. The whole feature set allows easy setup of very powerful “location based server load balancing”.

1.5 BalanceNG Design

BalanceNG is implemented in C based on a multithreading switching engine.

BalanceNG additionally uses multithreading (POSIX threads / pthreads) to operate different helper threads which communicate with the core multithreading switching engine.

These threads are used for the target specific health check scripts and the alert/upalert scripts (see the "target <n> script", target <n> alert" and "target <n> upalert" commands).

2 Initial Implementation

2.1 Hardware selection

BalanceNG runs on Mac OS X and nearly all contemporary Linux x86 systems, hence the choice of hardware is extensive. Please bear in mind that in any case the BalanceNG licensing (Full License) is hardware dependent, therefore it is recommended that the final deployment hardware be purchased prior to license purchase.

For pre-implementation and evaluation testing, you may use the test license implicitly supplied with the software (Basic License, for restrictions see section 2.5).

If you have a free choice of hardware, then the following is recommended:

- Choose 2 identical nodes (If High Availability is required).
- Use of a separate interface for administrative purposes and then as many 100/1000BaseT interfaces for traffic as required. We have found that Intel Gigabit Adapter Cards (or chip sets) work very well.
- If gigabit performance is required select a machine with two (or more) on board gigabit interfaces avoiding PCI bandwidth problems. Use the fastest processor you can get.
- Install the Linux distribution of choice. You may utilize one of the special embedded Linux distributions which do not require a hard disk.
- If you wish to use a separate or multiple networks, an additional Ethernet switch (or VLAN) will probably be required.

There's a benchmark functionality, see the “benchmark” command and take a look at the benchmark results page at http://www.inlab.de/balanceng/bmresults.html .

2.2 Network Setup

A common setup of BalanceNG comprises the following topology (pictured below): A local IPv4 network connected via a router to an external network (e.g. the Internet) and via one or more BalanceNG nodes to the target network.

We refer to the the network connected to the Router the "Access Network", and the network to which the Targets are connected the "Target Network".

Running BalanceNG "single legged" allows integration of BalanceNG into an already existing network installation.

Note: The real physical servers are called "Targets" in the BalanceNG world.

Note: Talking about "Servers" within BalanceNG refers to "Virtual Servers" being presented by BalanceNG.

BalanceNG Servers perform load balancing over one or more BalanceNG Targets.

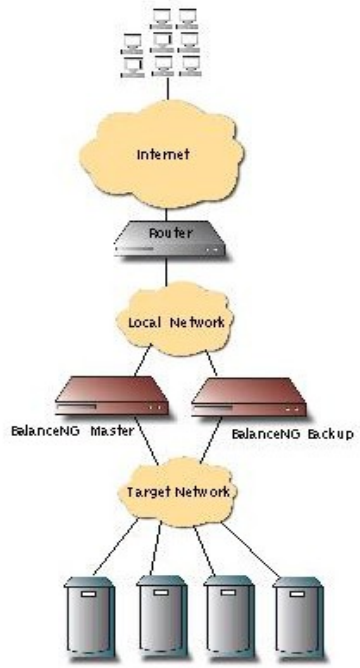


Figure 1: Common BalanceNG Network Setup

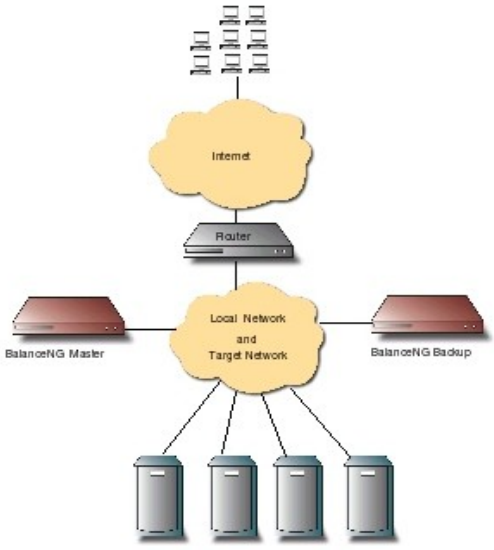


Figure 2: "Single Legged" Network setup

2.3 Installation

2.3.1 Installation on Linux (Debian/Ubuntu Package)

The installation on Debian and Ubuntu i386 systems using the ".deb" package distribution is very easy. A typical command line installation looks like this:

```
# ls BalanceNG*
BalanceNG_5.016_i386.deb
# dpkg -i BalanceNG_5.016_i386.deb
Selecting previously deselected package balanceng.
(Reading database ... 20093 files and directories currently installed.)
Unpacking balanceng (from BalanceNG_5.016_i386.deb) ...
Setting up balanceng (5.016) ...
```

The BalanceNG binary installs in /sbin/bng in that case. Init scripts are also setup starting all active instances automatically on startup (if a config file exists of that instance). The .deb package does not include any configuration files (like /etc/bng.conf or /etc/bng.global).

Removing BalanceNG works like this:

```
# dpkg -r BalanceNG
(Reading database ... 20095 files and directories currently installed.)
Removing balanceng ...
BalanceNG: not running
#
```

2.3.2 Installation on macOS 10

BalanceNG for macOS is installed by a simple tarball-based procedure. Please refer to the online instructions at https://load-balancer.inlab.net.

2.3.3 Installation on Linux (tarball)

The Linux Distribution of BalanceNG comes as a tar archive as well, containing the BalanceNG binary.

The download file is a "gzipped" tar archive (`BalanceNG-<version>-<OS>-<architecture>.tar.gz`) located wherever you downloaded it.

Please extract this tar archive using the "`tar`" command as follows (example, assuming BalanceNG 5.016 for Linux downloaded into /tmp):

```
# cd /tmp
# ls BalanceNG*
BalanceNG-5.016-Linux-x86.tar.gz
# tar xvfz BalanceNG-5.016-Linux-x86.tar.gz
./BalanceNG-5.016-Linux-x86/
./BalanceNG-5.016-Linux-x86/README
./BalanceNG-5.016-Linux-x86/LICENSE
./BalanceNG-5.016-Linux-x86/bng
./BalanceNG-5.016-Linux-x86/bngagent
./BalanceNG-5.016-Linux-x86/bngagent.c
./BalanceNG-5.016-Linux-x86/bngagent-binaries/
./BalanceNG-5.016-Linux-x86/bngagent-binaries/bngagent-1.42-Linux-x86
./BalanceNG-5.016-Linux-x86/bngfilt-binaries/
./BalanceNG-5.016-Linux-x86/bngfilt-binaries/bngfilt32-SPARC
./BalanceNG-5.016-Linux-x86/bngfilt-binaries/bngfilt32-X86
./BalanceNG-5.016-Linux-x86/bngfilt-binaries/bngfilt64-SPARC
./BalanceNG-5.016-Linux-x86/bngfilt-binaries/bngfilt64-X86
./BalanceNG-5.016-Linux-x86/bngfilt-binaries/README
./BalanceNG-5.016-Linux-x86/contrib/
./BalanceNG-5.016-Linux-x86/contrib/BngAgentService.zip
./BalanceNG-5.016-Linux-x86/contrib/README
#
```

The files extracted are:

README	Describes where to obtain BalanceNG
LICENSE	Is a copy of the End User License Agreement
bng	Is the BalanceNG application executable binary
bngagent[.c]	Are the executable and source for the BalanceNG agent

Additional directories of the distribution are:

bngagent-binaries	Precompiled bngagent binaries for different platforms
bngfilt-binaries	bngfilt STREAMS module binaries
contrib	Customer contributions without warranty and support

Note: The distribution may contain additional directories and files which are not mentioned here.

The executable “bng” should be copied to a suitable location. Since BalanceNG is init-script compatible we recommend to copy it to `/etc/init.d`, the default location for init scripts, as follows:

```
# cp ./BalanceNG-5.016/bng /etc/init.d
```

Finally you could verify the modes and ownership which should be as follows:

```
# ls -l /etc/init.d/bng
-rwxr-xr-x  1 root root 214268 Apr  2 22:07 /etc/init.d/bng
#
```

NOTE: Some Linux distributions (notably Gentoo Linux) do not permit executable binaries in /etc/init.d, in this case bng should be installed in /usr/sbin.

2.4 Making BalanceNG Turnkey

2.4.1 Making BalanceNG Turnkey on macOS 10

Please check the online manual at https://load-balancer.inlab.net.

2.4.2 Making BalanceNG Turnkey on Linux

If you wish to have BalanceNG started automatically at system boot (Turnkey), you must configure your system to execute “bng start” on boot.

Note: This step is only needed with the tarball distribution of BalanceNG, the Debian/Ubuntu package already contains the necessary init.d scripts.

The standard method is to provide a link from the binary “bng” wherever it is located to the required run-level directory. This is normally achieved as follows:

- Linking /etc/init.d/bng to a run-level directory and name, for example on a Fedora 3 release "ln -s /etc/init.d/bng /etc/rc5.d/S11bng".

Some Linux distributions do not support this method, please see table below, or refer to your distribution's documentation.

In the Linux realm we recommend the following:

Linux Distribution	***Notes***
Gentoo Linux	installing binary in /usr/sbin

Linux Distribution	*Notes*
	adding the line "/usr/sbin/bng start" to /etc/init.d/local.start adding the line "/usr/sbin/bng stop" to /etc/init.d/local.stop
Fedora Linux	standard method
Red Hat Linux	Package (.rpm)
CentOS Linux	Package (.rpm)
Debian Linux	Package (.deb)
Ubuntu Linux	Package (.deb)
SUSE SLES	Package (.rpm)

Table 1: Turnkey Installation Notes

2.4.3 Linux network offloading disable instructions

On Linux operating systems any offloading optimization needs to be disabled, the following command disables this for eth0, for example:

```
ethtool --offload eth0 rx off tx off gso off gro off lro off tso off
```

In order to make this permanent (reboot-safe), this instruction needs to be placed in a suitable “pre-up” script for any interface being accessed by BalanceNG. Latest BalanceNG V4 releases apply this setting automatically.

2.4.4 Multi-Instance init.d Script

Using the “bng -I” command line option it's possible to implement a init.d script which starts and stops all instances on a host machine with just one init script. Such a script could look like the following (just edit the DAEMON variable accordingly):

```
#!/bin/sh
PATH=/sbin:/usr/sbin:/bin:/usr/bin
NAME=bng
DAEMON=/usr/bin/$NAME

case "$1" in
  start)
        for i in `$DAEMON -I | /usr/bin/awk '{print $1}'`
        do $DAEMON start $i
        done
        exit 0
        ;;
  stop)
        for i in `$DAEMON -I | /usr/bin/awk '{print $1}'`
        do $DAEMON stop $i
        done
        exit 0
        ;;
  status)
        for i in `$DAEMON -I | /usr/bin/awk '{print $1}'`
        do $DAEMON status $i
```

```
        done
        exit 0
        ;;
  restart|force-reload)
        for i in `$DAEMON -I | /usr/bin/awk '{print $1}'`
        do $DAEMON restart $i
        done
        exit 0
        ;;
  *)
        echo "Usage: bng {start|stop|restart|force-reload}" >&2
        exit 3
        ;;
esac
```

2.5 Determining the Nodeid and Licensing

To obtain a full license for BalanceNG you require the nodeid of your hardware. The nodeid is a 6 byte hexadecimal number similar to an Ethernet address (the actual nodeid is derived from several system parameters: The MAC address of Ethernet interface 0 (eth0), CPU and PCI-bus parameters).

Alternatively, you may use the “virtual nodeid” (vnodeid) for licensing. The vnodeid is based on the Ipv4 address of the primary interface only and is therefore hardware independent.

The yearly subscription license is technically the same as the “full node license”.

To determine the nodeid of your installation:

Start BalanceNG:

```
# bng start
BalanceNG: starting up ...
#
```

Open the interactive mode of BalanceNG using:

```
# bng control
BalanceNG: connected to PID 8692
bng#
```

Enter the command to display the nodeid of the current machine:

```
bng# show nodeid
  ba:e9:75:26:36:ab
bng#
```

Alternatively you may enter the command to display the vnodeid of the current machine:

```
bng# show vnodeid
  21:06:5e:24:7a:f3 (interface eth0)
bng#
```

Note that the nodeid and the vnodeid of all instances are the same. Retrieving the nodeid of instance 2 looks like this:

```
# bng start 2
```

```
BalanceNG: starting up instance 2 ...
#
```

Open the interactive mode of BalanceNG using:

```
# bng control 2
BalanceNG: connected to instance 2 PID 27816
bng#
```

Enter the command to display the nodeid of the current BalanceNG instance 2:

```
bng# show nodeid
  ba:e9:75:26:36:ab
bng#
```

This allows you to use the same licensing key for all instances, currently there's a theoretical limit of 128 concurrent BalanceNG instances on the same host machine.

When you receive the license key(s) for your node(s), use the “license” command on each node to apply the relevant license. Use the “save” command to save the license information to /etc/bng.conf. An example license session is shown below:

```
# bng control
BalanceNG: connected to PID 8692
bng#
bng# show license
  status: valid full license
  serial: betaB048
  nodeid: 2a:e2:9f:38:da:20
  type "show version" for version and Copyright information
bng# save
ok
bng# ... bye
```

Now your BalanceNG installation is fully licensed and ready for operation.

2.6 Licensing of all Instances in /etc/bng.global

BalanceNG may be started multiple times at the same time (with many instances). The nodeid of all instances is exactly the same. In order to license all instances on the machine just put the “license” instruction as explained above into the /etc/bng.global file.

This file is evaluated by each instance of BalanceNG right at the beginning, so it may also be used for other global settings.

Example:

```
# cat > /etc/bng.global <<EOF
license betaB048 45f296fc9b387f7576b49d0ceab3f92e
EOF
# bng start 12
BalanceNG: starting up instance 12 ...
# bng ctl 12
BalanceNG: connected to instance 12 PID 13653
bng# show license
  status: valid full license
  serial: betaB048
```

```
  nodeid: 2a:e2:9f:38:da:20
  type "show version" for version and Copyright information
bng#
```

2.7 Trial license restrictions

BalanceNG comes with an "automatic" 30 day full functionality trial license.

The trial license restrictions for BalanceNG are as follows:

- Valid for 30 days and within a 3 month window after day of build of (whichever expires first).
- Instances need to be restarted after 7 days of continuous operation (e.g. "bng restart").

3 Command Reference

3.1 Command Line Interface

The Command Line Interface is designed in such a way, that BalanceNG may be directly linked as an init-Script e.g. from `/etc/init.d/bng -> /etc/rc5.d/S90bng.` There is no need to implement additional init scripts.

The BalanceNG binary is called "**bng**" and should be installed in `/etc/init.d` (another possibility is installing it to `/usr/sbin` linked to `/etc/init.d).`

Invoking BalanceNG requires root permissions (`euid == 0`) since BalanceNG has to be able to directly control the network interfaces.

3.1.1 bng

Invoking BalanceNG with no option displays the usage information together with some Copyright information.

Example:

```
# /etc/init.d/bng

          This is BalanceNG 5.016 (created 2022/11/08)

  Copyright (C) 2005-2017,2018 by Inlab Networks GmbH, Germany.
       All rights reserved / Alle Rechte vorbehalten.
    Visit https://www.inlab.net for further information.

  usage: bng [-df] start|stop|restart|status|control [instance]
#
```

The option `-f` requests that BalanceNG should stay in foreground, `-d` is being used for generating debugging information and implicitly sets `-f.`

BalanceNG may be started in multiple instances representing as many independent software load balancers on the same machine as desired. Instance 0 is the default instance (if no instance is specified).

3.1.2 bng -c *config-file* start | restart [*instance*]

The “-c” option allows to specify a specific configuration file which is being loaded on start or restart (instead of loading bng.conf, bng.private and bng.global).

In case that the configuration file cannot be opened for reading (or is non-existing), the BalanceNG instance starts up with a empty default configuration with no further error message.

The path specification of the configuration file needs to be absolute since the current working directory of any BalanceNG instance is always “/”.

Example:

```
# bng -c /tmp/test001.conf start
BalanceNG: starting up ...
#
```

3.1.3 bng start [instance]

This starts BalanceNG in the background as a daemon. BalanceNG reads as a first step the configuration from /etc/bng.conf (if it exists) and commences operation.

BalanceNG may be started in multiple instances supplying an optional integer parameter ranging from 0 up to and including 128. If the instance parameter is omitted the default instance 0 is assumed.

Example:

```
# /etc/init.d/bng start
BalanceNG: starting up ...
#

# /etc/init.d/bng start 41
BalanceNG: starting up instance 41 ...
#
```

If there's already a BalanceNG running on this machine the output may look like as follows:

Example:

```
# /etc/init.d/bng start
BalanceNG: already running with PID 7914
#
```

3.1.4 bng stop [instance]

This stops the current running BalanceNG process (or the specified instance), the output may look like this:

Example:

```
# /etc/init.d/bng stop
BalanceNG: shutdown of PID 7914 complete
#

# ./bng stop 41
BalanceNG: shutdown of instance 41 PID 27231 complete
#
```

3.1.5 bng reload [instance]

This command issues a CLI “reload” command (see “reload”), the requested instance needs to be running. A third command channel is used to avoid any race conditions (see “bng cmdctl”).

3.1.6 bng restart [instance]

This restarts BalanceNG (or the specified instance) and is equivalent to execute "`bng stop`" and "`bng start`".

Example:

```
# /etc/init.d/bng restart
BalanceNG: shutdown of PID 7919 complete
```

```
BalanceNG: starting up ...
#

# /etc/init.d/bng restart 41
BalanceNG: shutdown of instance 41 PID 27419 complete
BalanceNG: starting up instance 41 ...
#
```

3.1.7 bng status [instance]

This provides information about the current status of BalanceNG (or the specified instance).

Example:

```
# /etc/init.d/bng status
BalanceNG: running with PID 7921
# /etc/init.d/bng stop
BalanceNG: shutdown of PID 7921 complete
# /etc/init.d/bng status
BalanceNG: not running
#

# /etc/init.d/bng status 41
BalanceNG: instance 41 running with PID 27423
# /etc/init.d/bng stop 41
BalanceNG: shutdown of instance 41 PID 27423 complete
# /etc/init.d/bng status 41
BalanceNG: instance 41 not running
#
```

3.1.8 bng [-e] cmdctl [instance]

This connects to a third, additional control interface of the instance (if specified). This particular interface is reserved for internal purposes and used by the “bng reload” command.

If the option “-e” is specified, this command exits with EX_TEMPFAIL if the same command frontend is already running.

3.1.9 bng [-e] imsctl [instance]

This connects to a fourth control interface of the instance (if specified). This control interface may be used for other automatic purposes that must not interfere with “bng cmdctl” or “bng reload”.

If the option “-e” is specified, this command exits with EX_TEMPFAIL if the imsctl command frontend is already running.

3.1.10 bng [-e] control [instance]

This invokes the interactive configuration mode of BalanceNG (or the instance if specified). The shell invocation “bng control” may be abbreviated as “bng ctl”.

If the option “-e” is specified, this command exits with EX_TEMPFAIL if the same command

frontend is already running.

This is indicated by a prompt, the default is: "bng#".

Example:

```
# /etc/init.d/bng control
BalanceNG: connected to PID 7928
bng#

# /etc/init.d/bng restart 41
BalanceNG: not yet running
BalanceNG: starting up instance 41 ...
# /etc/init.d/bng control 41
BalanceNG: connected to instance 41 PID 27479
bng#
```

Typing EOF (Ctrl-D) exits the interactive configuration mode.

The interactive configuration mode allows simple command line editing using the arrow-keys. This is only active is an interactive terminal is detected on stdin, otherwise the command line editing option is switches off to allow automated programs to operate on the command line.

The following command line editing capabilities are currently supported:

arrow up	move up to previous command line
arrow down	move down to next command line in history
arrow left	move cursor left
arrow right	move cursor right
backspace, ^H or DEL	delete character before cursor
^D	exit to operating system shell
^U	erase all characters left of cursor
^W	erase word left of cursor

If the lexical scanning process finds the token "//" this token and the rest of the line is ignored (this is being used for adding timestamps and version stamps to the configuration file).

It's also possible to pipe a command into BalanceNG invoked with "bng control" like this:

```
# echo "show targets" | bng control
bng# show targets
    # ipaddr           port prt net srv sessions status      name
  ------------------------------------------------------------------------
    1 172.17.2.90       any any   1   1        0 operational
bng#
#
```

In general commands may be abbreviated interactively as long as there are no ambiguities.

3.1.11 bng [-e] auxctl [instance]

This connects to a second, auxiliary control interface of the instance if specified. This interface is intended to be reserved for external programs and user interfaces offering exactly the same functionality as “bng control”.

If the option “-e” is specified, this command exits with EX_TEMPFAIL if the same command frontend is already running.

3.1.12 bng purge [instance]

This command removes the associated configuration file of the supplied instance without any further warnings. The instance needs to be down (off) for that purpose. If no instance is specified, the configuration file /etc/bng.conf of the default instance is deleted.

A private configuration data file (e.g. /etc/bng.private) is not deleted by this command.

Example:

```
# bng -I
 0 off
 3 off
# bng purge 3
BalanceNG: /etc/bng3.conf successfully deleted
#
```

3.1.13 bng -I

This command displays information about the state of all instances of BalanceNG on the system. “running” indicates a running instance whereas “off” indicates an available configuration file.

Example:

```
# bng -I
 0 off
# bng start 3
BalanceNG: starting up instance 3 ...
# bng -I
 0 off
 3 running
# bng ctl 3
BalanceNG: connected to instance 3 PID 14277
bng# save
ok
bng# ... bye
# bng stop 3
BalanceNG: shutdown of instance 3 PID 14277 complete
# bng -I
 0 off
 3 off
#
```

3.1.14 bng -L

This commands allows to check the validity of a serial number and a license key for the current node. It requires two arguments, the serial number and the license key. If the license information is valid the invocation of bng returns the Linux return code 0 (77 otherwise).

Example:

```
# bng -L TEST 3bef9fa6b31acec6f64abc162b3dcbda
```

```
# echo $?
0
```

3.1.15 bng -M

This command displays the MAC address of a BalanceNG instance, which is always allocated out of the MA-L (MAC address large) block of Inlab Networks (34-38-AF). An instance number may be specified additionally with the **-i** option.

Example:

```
# bng -M
34:38:af:46:44:eb
# bng -i 0 -M
34:38:af:46:44:eb
# bng -i 1 -M
34:38:af:47:44:eb
# bng -i 120 -M
34:38:af:3e:44:eb
# bng -i 129 -M
BalanceNG: invalid instance number
#
```

3.1.16 bng -N

This command displays the BalanceNG nodeid of the BalanceNG host machine without the need for starting a BalanceNG instance. The nodeid is needed for licensing purposes.

Example:

```
# bng -N
2a:e2:2f:38:4a:20
#
```

3.1.17 bng -V

This command displays the BalanceNG vnodeid (virtual nodeid) of the BalanceNG host machine without the need for starting a BalanceNG instance. The vnodeid is useful for licensing in virtual environments and is bound to the IPv4 address of the BalanceNG primary interface only.

Example:

```
# bng -V
c7:f9:dd:6d:a2:74
#
```

3.1.18 bng -W

This command option expects two arguments, a serial number and a license key. If the license is valid a “license” line with those parameters is written to the file /etc/bng.global without further warning.

3.2 Administrative and Informational Commands

3.2.1 arp-insert

Synopsis: `arp-insert <ip-address> <mac-address>`

This command allows to insert an ARP/ND6 entry for testing purposes only. This makes it possible to simulate the presence of a target.

Example:

```
bng# arp-insert ::ffff:172.16.30.26 00:50:56:ab:6e:97
bng#
```

3.2.2 benchmark

Synopsis: `benchmark <send-if> <rcv-if>`

This command executes a hardware, OS and network benchmark in the background controlled by the parameters “bmduration”, “bmpsize” and “bmwsize” (see “set” command below). The results and/or status can be shown with “show benchmark”.

This command requires the “benchmark” module being loaded (“module benchmark”).

Please note that the benchmark functionality is less efficient than the operation of BalanceNG in multithreading mode, for measuring real packet processing performance of a production configuration this needs to be performed using other external testing tools.

Example:

```
bng# module benchmark
bng# interface eth0
bng# interface eth1
bng# benchmark eth0 eth1
bng# show benchmark
  benchmark active and running:
    duration (seconds) : 300
    packetsize         : 1514
    window size        : 16
    packets sent       : 212921
    packets received   : 212905
    seconds remaining  : 294
bng# show benchmark
  benchmark finished with the following results:
    duration (seconds) : 300
    packetsize         : 1514
    window size        : 16
    packets sent       : 10797158
    packets received   : 10797142
    lost packets       : 0
    packets per second : 35990
    bytes per second   : 54489577
bng#
```

3.2.3 bngsynctest

Synopsis: `bngsynctest <subtype>`

This command allows to test the generation of bngsync protocol packets. Only the subtype needs to be specified, ranging from 4 to 8 (including) as specified. The IPv4/IPv6 protocol choice is determined by the type of the network <N> syncpeer setting of the network being referenced by the VRRP section.

The generated packets contain the subtype 0x01 (1) instead of 0x00 (0) in order to faciliate

automatic testing.

The source and destination port is taken from the “bngsyncport” parameter (default 10439).

The destination address is the “network <n> syncpeer” address and the source address is either the “network <n> real” or “network <n> real6” address (for IPv4 or IPv6, respectively).

3.2.4 bngsyncudptest

Synopsis: `bngsyncudptest <network> <string>`

This command allows to test the basic generation of UDP packets for the bngsync protocol. It uses UDP over IPv4 or IPv6 depending of the syncpeer address type of the specified network index.

The source and destination port is taken from the “bngsyncport” parameter (default 10439) and the specified ASCII data is the variable length data portion of the UDP packet (and thus they are not bngsync protocol messages).

The destination address is the “network <n> syncpeer” address and the source address is either the “network <n> real” or “network <n> real6” address (for IPv4 or IPv6, respectively).

This command is for automatic and manual testing only and should not be used on a live BalanceNG installation.

3.2.5 check

Synopsis: `check`

A check of the current active configuration is performed. This is especially targeted towards the configuration of Servers and Targets. A warning is being issued at the following conditions:

- A Target is references by multiple Servers
- A Target is enabled, but not referenced at all

This function is implicitly called when Servers or Targets are entering the enabled state.

Example (no Warning):

```
bng1# check
bng1#
```

Example (with Warnings):

```
bng1# check
WARNING: target 1 already referenced from server 1
WARNING: target 1 reference from server 8 ignored
WARNING: target 2 already referenced from server 1
WARNING: target 2 reference from server 8 ignored
bng1#
```

3.2.6 clear

Synopsis: `clear <item>`

This command clears counters which are maintained for informational purposes.

“clear ?” shows a list of supported items like in this Example:

```
bng# clear ?
  available items:
```

```
  locationcounters        IPDB location counters
bng#
```

“clear locationcounters” clears the counters which are shown by the “show locations” command. A typical dialog may look like this:

```
bng# clear locationcounters
bng# show locations
  key  counter description
  --- -------- -----------------------------------
   AD          ANDORRA
   AE          UNITED ARAB EMIRATES
   AF          AFGHANISTAN

   ...

   YT          MAYOTTE
   ZA          SOUTH AFRICA
   ZM          ZAMBIA
   ZW          ZIMBABWE
   -           *** NOT FOUND PSEUDO ENTRY ***
  --- -------- -----------------------------------
  235        0 total
bng#
```

3.2.7 coredump

Synopsis: `coredump`

This command deferences a NULL pointer in the BalanceNG main thread and initiates a core dump to be written (if the OS settings allow this). This command if for debugging purposes only.

3.2.8 event <event-id>

Synopsis: `event <event-id>`

This command records the occurrence of an event with the specified event-id in the session table. An event-id is an arbitrary text with a maximum length of 60 characters. The session-id has internally a leading '#' character in order to distinguish the event type session table entries in the session table.

The number of occurrences of an event is internally kept in the “server” column of the session table entry, the point of time since this particular number of occurrences were recorded is kept in the “target” column.

This command is useful for testing the event handling.

Example:

```
bng# event test123
bng# event test123
bng# show sessions
  1 session
    srv   tgt   age timeout ftimeout SYNC session-id
  ----- ----- ----- ------- -------- ---- ------------------
```

```
      2  3480     4     600        0      #test123
bng#
```

3.2.9 evcount <event-id>

Synopsis: `evcount <event-id>`

This command displays the currently kept number of occurrences of the specified event.

Example:

```
bng# evcount test123
8
bng#
```

3.2.10 evfix <event-id>

Synopsis: `evfix <event-id>`

This command fixes the specified event in the session table with all its parameters (thus it will be never reclaimed by the session table garbage collection).

This command is intended for testing the event handling system and functionality.

Example:

```
bng# event test123
bng# evfix test123
bng# show sessions
  1 session
    srv   tgt   age timeout ftimeout SYNC session-id
  ----- ----- ----- ------- -------- ---- ------------------
      1    12 fixed     600        0      #test123
bng#
```

3.2.11 evrate <event-id>

Synopsis: `evrate <event-id>`

This command displays the current rate per second of the occurrences of the specified event.

The internal count and time interval values are recalculated for this event if necessary.

If the specified event has never occurred (or is no longer kept in the session table) the shown value is 0 (as expected).

Example:

```
bng# evrate test123
4
#bng
```

3.2.12 evtime <event-id>

Synopsis: `evtime <event-id>`

This command displays the currently kept time interval for the counting of the specified event (in seconds from the past until now).

If the specified event has never occurred the value 1 is shown (as returned by the internal functionality).

Example:

```
bng# evtime test123
392
bng#
```

3.2.13 evunfix <event-id>

Synopsis: `evunfix <event-id>`

This command unfixes the specified event in the session table by setting the internal age (or time of birth) to the current uptime.

This command is intended for testing the event handling system and functionality.

Example:

```
bng# evunfix test123
bng#
```

3.2.14 help

Synopsis: `help`

Displays the main help information about available commands.

Example:

```
bng# help
available commands, type EOF (^D) to exit:
  arp <ip4>                       declare IP address to be ARP-resolved
  benchmark <ifsend> <ifrcv>      perform BalanceNG loopback benchmark
  check                           perform configuration check
  clear <item>                    clear counter (type "clear ?" for help)
  commit <item> <number(s)>       register and enable networks/servers/targets
  coredump                        dump core immediately
  disable <item> <number(s)>      disable networks/servers/targets
  dump <if> <dir>                 pcap dump traffic on <if> to <dir>
  edit <item> <numbers>           disable+unregister networks/servers/targets
  enable <item> <number(s)>       enable networks/servers/targets
  gateway ...                     gateway commands (see "help gateway")
  help                            display this information
  help <topic>                    display topic specific information, ? for list
  hostname <name>                 specify name of this BalanceNG instance
  inject <if> <pcap> <s> [<e>]    inject packets from pcap file
  interface ...                   interface commands (see "help interfaces")
  ipallow <IP-address>            insert IP address into list (module ipallow)
  ipdeny <IP-Address>             insert IP address into list (module ipdeny)
  ipdb [<file.csv>]               load IPDB from .csv file
  license <serno> <key>           specify serno and license key
  locate <addr>                   lookup IP address in current IPDB
  log <message>                   log a message to the BalanceNG log
  macallow <mac-address>          insert MAC address into list (module macallow)
  macdeny <mac-address>           insert MAC address into list (module macdeny)
  macrouter <mac-address>         declare MAC address as routing device
  modules <ma>,<mb>,...           define packet processing module chain
  network <idx> <cmd> <value>     modify network <idx>, see "help network"
  no <command>                    revert command
  purge <item> <number(s)>        re-initialize servers/targets
  register <item> <number(s)>     register networks/servers/targets
  reload                          reload server and target configuration
  remark <remark>                 specify configuration remarks
  resync                          schedule a complete session table resync
  rms <id>                        remove session table entry (exact lookup)
  rmsession <id>                  remove session table entry
  rmt <target>                    remove all session table entries of specific target
```

```
  save                              shorthand for "save conf"
  save conf                         save current configuration
  save private                      save private configuration data
  save all                          save configuration and private data
  server <idx> <cmd> <value>        modify server <idx>, see "help server"
  sessiondump <file>                dump all sessions to file
  sessionload <file>                load all session information from file
  set <parameter> <value>           set parameter to specific value
  show <item>                       show item, show ? for item list
  shutdown                          alias for stop
  snapshot <file>                   collect all relevant service data in file
  snapshot-light <file>             collect snapshot with limited sessiontable dump
  snatrange <from> <to>             specify SNAT IPv4 address range
  softdisable target <number(s)>    don't create new sessions for target(s)
  stop                              stop background process and exit
  target <idx> <cmd> <value>        modify target <idx>, see "help target"
  tnat <ipa> <ipb> <pr> <prt>       target NAT for outbound communications
  unregister <item> <number(s)>     unregister networks/servers/targets
  vip <ip4>                         represent VIP using ARP
  vrrp <sc> <value>                 vrrp settings, see "help vrrp"
bng#
```

3.2.15 help <subtopic>

Synopsis: `help <subtopic>`

Displays subtopic based help information.

Example:

```
bng# help ?
available help topics:
  ?                              show this list of available help topics
  cledit                         command line editing functionality
  gateway                        default gateway commands
  interface                      interface commands
  network                        network commands
  server                         virtual server commands
  show                           information about items to show
  target                         target (real server) commands
  vrrp                           vrrp commands
bng# help vrrp
  vrrp bscript <script>          specify BACKUP state notify script
  vrrp mscript <script>          specify MASTER state notify script
  vrrp network <number>          specify network for advertisements
  ...
bng# help network
  network <n> addr <addr>        specify network address
  network <n> interface <if>[,]  specify one or more interfaces
  network <n> interface none     remove all interface declarations
  ...
bng# help gateway
  gateway alert <script>         specify gateway alert notification script
  gateway arp <iv>,<to>          perform arp healthcheck (ival,tout)
  gateway arp off                disable arp healthcheck
  ...
bng# help interface
  interface <name>               compatibility syntax
  interface <n> name <name>      specify OS interface name
  interface <n> trackval <val>   specify interface tracking value (default=0)
  ...
bng# help server
  server <n> backup <t>          specify one single backup target
  server <n> backups <t1>,...    specify multiple backup targets
  server <n> backup none         remove all backup target declarations
  ...
bng# help show
  show arphash                   show arp hashtable
  show benchmark                 show benchmark status and results
```

```
  show conf                       show current configuration
  ...
bng# help target
  target <n> agent <p>,<iv>,<to> perform agent operation (port,ival,tout)
  target <n> agent off            remove/disable agent operation
  target <n> alert <script>       specify external alert script
  ...
bng# help cledit
  arrow up                        move up to previous command line
  arrow down                      move down to next command line in history
  arrow left                      move cursor left
  arrow right                     move cursor right
  backspace, ^H or DEL            delete character before cursor
  ^D                              exit to operating system shell
  ^U                              erase all characters left of cursor
  ^W                              erase word left of cursor
```

3.2.16 inject

Synopsis: inject `<interface> <pcap-file> <from> [<to>]`

This command injects one or more packets provided in a file in pcap format into the BalanceNG interface with the specified index or interval. If <to> is omitted only one packet is injected (index <from>).

This command is intended for debugging and QA purposes only and must not be used in a productive environment.

Example:

```
bng# help ?

# bng ctl
BalanceNG: connected to PID 27055
test# inject 1 /tmp/icmp.pcap 2
no:      2 len:     70
00 00 5E 00 44 04 04 1C 7F 43 48 F0 08 00 45 00
00 38 C9 11 34 04 F7 01 2B 72 0A EE F0 88 0A EE
C8 DC 03 04 34 F8 00 00 05 48 43 68 05 50 30 32
40 00 76 06 E4 65 0A EE C8 DC 0A 43 02 6E 43 BB
F2 43 2C E4 A4 C1
      vrrp ... continue
       arp ... continue
      ping ... continue
        hc ... continue
    master ... continue
       slb ... done.
test#
```

3.2.17 locate

Synopsis: `locate <IP-address>`

This command initiates a lookup in the in-memory IPDB or IPDB6 databases (IP to location databases) with the given IP address (IPv4 or IPv6) as the key. The location counters (as show by “show locations”) are not being incremented by this lookup.

Example:

```
# bng start 2
BalanceNG: starting up instance 2 ...
# bng control 2
BalanceNG: connected to instance 2 PID 12423
bng# ipdb
bng# ipdb6
```

```
bng# sh ipdb
  IPDB loaded from /opt/BalanceNG/ip-to-country.csv
  110100 valid 5-column lines
  110100 total IPDB entries available
  no consecutive area overlaps
  241 different IPDB locations referenced
  IPDB6 loaded from /opt/BalanceNG/IpToCountry.6R.csv
  5636 valid 5-column lines
  5636 total IPDB6 entries available
  no consecutive area overlaps
bng# locate 82.135.110.2
  address 82.135.110.2 is in DE (GERMANY)
bng# locate 2a01:198:200:76c::2
  address 2a01:198:200:76c::2 is in DE
bng#
```

3.2.18 pcap-search

Synopsis: `pcap-search <pcap-file> <search-string>`

This command searches for the specified string in every packet contained in the specified file in pcap format and prints out the numbers of packets containing that string.

Example:

```
# pcap-search /tmp/lan.pcap 4e01ddb1a498
639
640
641
642
798 packets processed
```

3.2.19 purge

Synopsis:
```
purge interface <interface no.>
purge network <network no.>
purge server <server no.>
purge target <target no.>
```

This command resets the given interface, network, server or target data structures to an initial uninitialized state. The object must be in unregistered state, multiple interface, network, server or target numbers may be specified.

Example:

```
bng# unregister target 1
bng# purge target 1
```

3.2.20 rms

Synopsis: `rms <session-id>`

This command removes a single session table entry by performing one single session table lookup. If there's no session table entry found, this command has no side-effect.

3.2.21 rmsession

Synopsis: `rmsession <session-id substring>`

This command removes all session table entries with a matching session-id substring (the same set as shown by “show session <session-id substring>”). Please note that a complete linear traversal of the session table is performed every time this command is executed.

3.2.22 rmt

Synopsis: `rmt <target>`

This command removes all session table entries which are associated to a specific target. Please note that a complete linear traversal of the session table is performed every time this command is executed.

3.2.23 resync

Synopsis: `resync`

This command starts a session table resync on the current VRRP master. The parameter sessionsyncack needs to be set to 1 (enabled).

Example:

```
bng# resync
ok, 1000000 entries scheduled for resync
bng# sh log
  2012/12/20 16:33:27 6 RESYNC OF 1000000 ENTRIES SCHEDULED
  2012/12/20 16:41:31 6 RESYNC COMPLETE
bng#
```

3.2.24 save

Synopsis: `save`

This saves the current configuration (as shown by "`show conf`") to the configuration file of BalanceNG in `/etc/bng.conf` (or `/etc/bngN.conf` for other instances).

Example:

```
bng# save
  saved /etc/bng.conf
bng#
```

3.2.25 save conf

Synopsis: `save conf`

This command saves the current configuration (as shown by “`show conf`”) to the configuration file just like “save”.

3.2.26 save private

Synopsis: `save private`

This command saves the node specific private data in the associated private data configuration file (`/etc/bng.private` or `/etc/bngN.private`).

The following data is considered to be “node specific private”:

– The “hostname” setting

– The VRRP priority

– All “network <n> real” addresses

The private data may be displayed with “show private” without any saving.

Example:

```
bng# save private
  saved /etc/bng.private
bng#
```

3.2.27 save all

Synopsis: `save all`

This command save both the configuration and the private configuration data of the current instance.

Example:

```
bng# save all
  saved /etc/bng.conf
  saved /etc/bng.private
bng#
```

3.2.28 sessiondump

Synopsis: `sessiondump <filename>`

The command dumps the complete session table information into the specified file in ascii readable text format (as displayed by the “show sessions” command).

Note: During the dumping process the internal packet forwarding mechanism is paused, which may cause a noticeable delay of packet processing at very large session tables.

Example:

```
bng@testnode# sessiondump /tmp/test.txt
dumping 17 sessions to /tmp/test.txt ...... done.
bng@testnode# ... bye
testnode# cat /tmp/test.txt
      hash ip-address       port  srv tgt  age   stout
  -------- --------------- -----  --- --- ---- -------
   7479774 172.17.2.4      32823    4   1   33  120000
   7479773 172.17.2.4      32822    4   1   34  120000
   7479772 172.17.2.4      32821    4   1   34  120000
   7479771 172.17.2.4      32820    4   1   34  120000
   7479770 172.17.2.4      32819    4   1   34  120000
   7479769 172.17.2.4      32818    4   1   35  120000
   7479768 172.17.2.4      32817    4   1   35  120000
   7479767 172.17.2.4      32816    4   1   35  120000
   7479766 172.17.2.4      32815    4   1   36  120000
   7479765 172.17.2.4      32814    4   1   36  120000
   7479764 172.17.2.4      32813    4   1   36  120000
   7479763 172.17.2.4      32812    4   1   37  120000
   7479762 172.17.2.4      32811    4   1   37  120000
   7479761 172.17.2.4      32810    4   1   38  120000
   7479760 172.17.2.4      32809    4   1   38  120000
   7479759 172.17.2.4      32808    4   1   39  120000
```

```
   7475751 172.17.2.4      32901    4   1   60  120000
testnode#
```

3.2.29 sessionload

Synopsis: `sessionload <filename>`

This command allows to load the internal session-table from a file, that has been previously exported by a “sessiondump” command invocation.

3.2.30 show

Synopsis: `show <item>`

The command show displays various informations about the current running BalanceNG process.

3.2.30.1 show ?

Synopsis: `show ?`

This informational command displays which items may be specified using the `show` command (replacing the ?).

Example:

```
bng# show ?
 show arphash                  show arp hashtable
  show benchmark                show benchmark status and results
  show break                    show current process break
  show conf                     show current configuration
  show debugscopes              show available debugscopes
  show gateway                  show gateway info and status
  show ifstat                   show interface statistics
  show instance                 show current instance number
  show interfaces               same as ifstat
  show ipdb                     ship IPDB (IP location database) info
  show lgrp                     show location group information
  show lgrp <A-Z>               show specific location group status
  show license                  show license information
  show locations                show available IPDB locations
  show log                      show recent log messages
  show machash                  show learned mac addresses
  show modules                  show available and active modules
  show nat                      show network address translations
  show networks                 show specified networks
  show network <n>              show network <n> information
  show nodeid                   show licensing nodeid
  show nous                     show number of unsync'ed sessions
  show parameters               show settable parameter data
  show private                  show private configuration data
  show server <n>               show status of server <n>
  show servers                  show server overview
  show sessions                 show session table information
  show snat                     show SNAT information
  show startuplog               show startup log messages
  show stinfo                   show session hashtable information
  show targets                  show target overview
  show target <n>               show status of target <n>
```

```
  show threads                  show threads overview
  show uptime                   show uptime in seconds
  show vips                     show vips and their mac-addresses
  show vnodeid                  show licensing vnodeid
  show version                  display version information
  show vrrp                     display VRRP status
bng#
```

3.2.30.2 show arphash

Synopsis: `show arphash`

Displays the current ARP hash of BalanceNG. The output consists of several columns: The IP-address, the MAC address (00:00:00:00:00:00 if not yet resolved) and several flags.

Static entries are usually self generated and maintained entries (like BalanceNG virtual servers), dynamic entries are usually BalanceNG targets.

Example:

```
bng1# show arphash
  ipaddr          ethaddr            tgt net  cntr   age flags
  --------------- ----------------- ---- --- ----- ----- -----------
  172.17.2.61     06:00:ac:11:02:3d    -   1     -     - vip fix
  172.17.2.64     00:00:5e:00:01:0e    -   1     -     - vip
  172.17.2.91     00:e0:81:5d:2a:65    1   -   203   192
bng1#
```

3.2.30.3 show benchmark

Synopsis: `show benchmark`

Displays the current benchmark status and the results (if finished).

Example:

```
 bng# show benchmark
  benchmark active and running:
    duration (seconds) : 300
    packetsize         : 1514
    window size        : 16
    packets sent       : 212921
    packets received   : 212905
    seconds remaining  : 294
bng# show benchmark
  benchmark finished with the following results:
    duration (seconds) : 300
    packetsize         : 1514
    window size        : 16
    packets sent       : 10797158
    packets received   : 10797142
    lost packets       : 0
    packets per second : 35990
    bytes per second   : 54489577
```

3.2.30.4 show bngsync

Synopsis: `show bngsync`

Displays the current status of the bngsync protocol settings.

Example:

```
bng# show bngsync
  bngsync is active
  we are talking (and expecting) IPv6 UDP port 10439
  the MAC address for syncpeer is known as 06:01:00:40:39:b2
bng#
```

3.2.30.5 show break

Synopsis: `show break`

Displays the current process break (of the BalanceNG main thread) as returned by sbrk() in hexadecimal format.

Example:

```
bng# show break
  current break is 0xb17b000
bng#
```

3.2.30.6 show conf

Synopsis: `show conf`

Displays the current active configuration of BalanceNG as it would be saved to `"/etc/bng.conf"` using the `save` command.

Example:

```
bng# show conf
  //        configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  interface eth0
  network   1 {
            addr 10.2.2.0
            mask 255.255.255.0
            real 10.2.2.20
            virt 10.2.2.30
            interface eth0
  }
  register  network 1
  enable    network 1
  //        end of configuration
bng#
```

3.2.30.7 show conf <section>

Synopsis: `show conf <section>`

This command displays specific sections of the configuration file intended for automatic access by programs layered on top of BalanceNG (like Web-UI's). Currently supported sections by this command are “gateway”, “vrrp”, “ipdb”, “lgrp” and “parameters”.

Example:

```
# bng control
BalanceNG: connected to PID 15873
NodeA# show gateway
       ipaddr: 172.17.2.254
 total status: operational
NodeA# show conf gateway
  gateway   {
            ipaddr 172.17.2.254
  }
NodeA# show conf vrrp
  vrrp      {
            vrid 14
            priority 200
            network 1
            tracking enable
            mscript "/usr/bin/logger -p daemon.notice VRRP MASTER"
            bscript "/usr/bin/logger -p daemon.notice VRRP BACKUP"
  }
NodeA# show conf parameters
  set       {
            vrrppreempt 1
            localdsr 1
  }
NodeA# show conf ipdb
  ipdb      "/opt/BalanceNG/ip-to-country.csv"
NodeA# show conf lgrp
  lgrp      {
            A "DE,AT,CH"
            B "*,!A,!Z"
            D "E"
            E "F"
            F "GB"
            Y "D,F"
            Z "!Y"
  }
NodeA#
```

3.2.30.8 show conf hostname

Synopsis: `show conf hostname`

This command displays just the “hostname” line as it would appear in the output of “show conf”.

3.2.30.9 show conf network

Synopsis: `show conf network <idx>`

`show conf network states`

This command displays specific network sections the same way as in the configuration file.

3.2.30.10 show conf remark

Synopsis: `show conf remark`

This command displays just the “remark” line as it would appear in the output of “show conf”.

3.2.30.11 show conf server

Synopsis: `show conf server <idx>`

`show conf server states`

This command displays specific server sections the same way as in the configuration file.

3.2.30.12 show conf target

Synopsis: `show conf target <idx>`

`show conf target states`

This command displays specific target sections the same way as in the configuration file.

3.2.30.13 show debugscopes

Synopsis: `show debugscopes`

This command displays a list of the available settings for the “debugscope” parameter.

Example:

```
testA# show debugscopes
  0 off
  1 target script debugging
  2 target ascript debugging
  3 server plugin debugging
  4 DNS library debugging
  5 location group (LGRP) debugging
testB#
```

The “debugscope” parameter must not be enabled during productive use of BalanceNG.

3.2.30.14 show gateway

Synopsis: `show gateway`

Displays informations about the gateway and its current state (see "gateway" setup commands).

Example:

```
bng# show gateway
       ipaddr: 172.17.2.254
 total status: operational
  ping status: up
     trackval: 4
bng#
```

3.2.30.15 show ifstat

Synopsis: `show ifstat`

Displays informations about the Ethernet interfaces which are currently under control of BalanceNG.

The following data is shown:

- received packets
- received bytes
- sent packets
- sent bytes
- dumped bytes.

The row 'dumped bytes' refers to the number of bytes being already dumped in `pcap` format to the current active dump file (see the `dump` command). This is being used to trigger automatic dumpfile rotation as soon as this number of bytes exceeds the parameter `dumprotation` (see also the `set` command for more informations about parameters).

Example:

```
bng# show ifstat

index 0 (eth0)
index 1 (eth1)

interface 1 (eth0)
  link detection: TRUE
  address:
    00:01:80:68:28:2f
  received:
    packets 342
    bytes   22949
  sent:
    packets 0
    bytes   0
  dumped:
    bytes   0

interface 2 (eth1)
  link detection: TRUE
  address:
    00:0e:0c:6c:ba:4a
  received:
    packets 257
    bytes   40541
  sent:
    packets 166
    bytes   8276
  dumped:
    bytes   0
bng1#
```

3.2.30.16 show instance

Synopsis: `show instance`

This shows the number of the current BalanceNG instance.

```
# bng start 41
BalanceNG: starting up instance 41 ...
# bng control 41
BalanceNG: connected to instance 41 PID 27231
bng# show instance
  this is BalanceNG instance 41
bng#
```

3.2.30.17 show interfaces

Synopsis: `show interfaces`

This is a synonym and equivalent to "`show ifstat`".

3.2.30.18 show ipdb

Synopsis: `show ipdb`

This shows information about the current status of the internal IPDB in-memory database.

Example:

```
bng# show ipdb
  IPDB loaded from /opt/BalanceNG/ip-to-country.csv
  83429 valid 5-column lines
  83429 total IPDB entries available
  no consecutive area overlaps
  235 different IPDB locations referenced
bng#
```

3.2.30.19 show license

Synopsis: `show license`

This shows the current licensing status of BalanceNG on the current host machine (node). If the current configuration contains a valid license for the nodeid of the machine then the output could look like follows:

```
bng# show license
  status: valid full license
  serial: TEST0611021
  nodeid: 2a:e2:9f:38:da:20
  type "show version" for version and Copyright information
bng#
```

An unspecified license information or an invalid license key shows up as follows:

```
bng# show license
  status: no or invalid license, trial restrictions apply
  nodeid: 2a:e2:9f:38:da:20
  type "show version" for version and Copyright information
bng#
```

A automatic OEM licensing on specific OEM hardware looks like this:

```
bng# show license
  status: valid full OEM license
  nodeid: 2a:e2:9f:38:da:20
  type "show version" for version and Copyright information
```

```
bng#
```

There's the possibility to obtain promotional full licenses. This license allows testing of VRRP and allows unrestricted number of virtual servers. This license could show up as follows:

```
test# show license
  status: valid testing license (will terminate in 93 minutes)
  serial: TEST123/1.770
  nodeid: 2a:e2:9f:38:da:20
  type "show version" for version and Copyright information
test#
```

The license configuration is at the top in the configuration output or file (after "hostname" and "remark" entries if present). See the `license` and `nodeid` configuration command for further reference.

3.2.30.20 show lgrp

Synopsis: `show lgrp`

This command displays all currently configured location groups and their logical state. An “ok” in column 2 means that all group interdependencies have been solved for that group whereas a question mark indicates that the group specification references yet unknown information.

Example:

```
bng# show lgrp
  A ok "DE,AT,CH"
  B ok "*,!A,!Z"
  D ok "E"
  E ok "F"
  F ok "GB"
  Y ok "D,F"
  Z ok "!Y"
bng#
```

3.2.30.21 show lgrp <g>

Synopsis: `show lgrp <group>`

This command displays more detailed information about the specified location group (referenced as a single capital letter A-Z) including all locations that are member of this group.

Example:

```
bng# show lgrp
  A ok "DE,AT,CH"
  B ok "*,!A,!Z"
  D ok "E"
  E ok "F"
  F ok "GB"
  Y ok "D,F"
  Z ok "!Y"
bng# show lgrp A
  grp A (solved)
  txt DE,AT,CH
  key description
  --- -----------------------------------
```

```
   AT AUSTRIA
   CH SWITZERLAND
   DE GERMANY
  --- -----------------------------------
    3 total entries
bng# show lgrp B
  grp B (solved)
  txt *,!A,!Z
  key description
  --- -----------------------------------
   AD ANDORRA
   AE UNITED ARAB EMIRATES
   AF AFGHANISTAN
   AG ANTIGUA AND BARBUDA
   AI ANGUILLA
   AL ALBANIA
   ...
   ZM ZAMBIA
   ZW ZIMBABWE
   -  *** NOT FOUND PSEUDO ENTRY ***
  --- -----------------------------------
  233 total entries
bng#
```

3.2.30.22 show locations

Synopsis: `show locations`

This command displays the current referenced locations in the internal IPDB (if loaded).

Example:

```
bng# show locations
  key  counter description
  --- -------- -----------------------------------
   AD        1 ANDORRA
   AE       16 UNITED ARAB EMIRATES
   AF          AFGHANISTAN
   AG          ANTIGUA AND BARBUDA
   AI          ANGUILLA
   AL          ALBANIA
   AM        1 ARMENIA
   AN          NETHERLANDS ANTILLES
   AO          ANGOLA
   AQ          ANTARCTICA
   AR       79 ARGENTINA
   AS          AMERICAN SAMOA
   AT      149 AUSTRIA
   AU      279 AUSTRALIA
   AW          ARUBA
   AX
   AZ        4 AZERBAIJAN
   BA        4 BOSNIA AND HERZEGOVINA
   BB          BARBADOS
   BD        3 BANGLADESH
   BE       49 BELGIUM
   ...
   VC          SAINT VINCENT AND THE GRENADINES
   VE       18 VENEZUELA
```

```
   VG          VIRGIN ISLANDS, BRITISH
   VI          VIRGIN ISLANDS, U.S.
   VN       16 VIET NAM
   VU          VANUATU
   WF          WALLIS AND FUTUNA
   WS          SAMOA
   YE          YEMEN
   YT          MAYOTTE
   ZA          SOUTH AFRICA
   ZM          ZAMBIA
   ZW          ZIMBABWE
   -        98 *** NOT FOUND PSEUDO ENTRY ***
  --- -------- -----------------------------------
  235    14805 total
bng#
```

Note: The location counters may be reset with the “clear locationcounters” command.

3.2.30.23 show log

Synopsis: `show log`

Displays the most recent log messages as they have been sent to the syslog. Up to 40 log messages are stored in a cyclic buffer.

Errors found in the initial startup file (/etc/bng.conf) are also reported to the syslog and may be displayed here for further analysis.

Example:

```
bng1# show log
  2008/10/11 21:25:18 6 main interface is bge0
  2008/10/11 21:25:18 6 BalanceNG 1.672: starting background operation
  2008/10/11 21:25:18 6 loading /etc/bng.conf
  2008/10/11 21:25:18 6 configuration taken Sun Oct  1 15:58:39 2008
  2008/10/11 21:25:18 6 configuration saved by BalanceNG 1.643 (created 2008/10/01)
  2008/10/11 21:25:18 5 this virtual router is now BACKUP
  2008/10/11 21:25:18 6 /etc/bng.conf successfully loaded
  2008/10/11 21:25:22 5 this virtual router is now MASTER
  2008/10/11 21:25:24 5 target 1 operational
bng1#
```

3.2.30.24 show machash

Synopsis: `show machash`

Displays the learned MAC addresses and their associated interfaces. For security reasons BalanceNG restricts the amount of different MAC addresses to 10000 (and will stop any further learning in that case).

If the MAC address belongs to a target the target number is show in the “tg” column of the output.

Example:

```
bng# show machash
  ethaddr            tg ifc
  ------------------ -- ----
  00:04:13:25:06:97     bge0
  00:0a:8a:f8:cb:01     bge0
  00:14:4f:48:82:50     bge0
  00:04:13:22:1b:03     bge0
  00:14:bf:66:70:36     bge0
```

```
  00:00:5e:00:01:09    bge0
  00:0c:f1:9c:90:e8    bge0
  00:14:38:95:7a:04    bge0
  06:00:ac:11:02:54    bge0
  00:0b:82:00:a8:fe    bge0
  00:20:e0:69:ad:4c    bge0
  00:11:50:c3:54:ce    bge0
  00:e0:81:5d:2a:65    bge0
  00:02:2d:15:ac:53    bge0
  00:04:13:25:06:96    bge0
  00:40:63:c9:f5:b5    bge0
  00:0e:0c:6c:ba:59  1 bge0
bng#
```

3.2.30.25 show maxbucket

Synopsis: `show maxbucket`

This command displays the length of the largest bucket list of the internal session table management data structure. This length should be minimal and is an indicator for the quality of the session-id hash function. This command blocks all packet processing threads for some short time and should therefore not be executed during production or stress testing.

Example:

```
bng# sh maxbucket
  length of longest session table bucket list: 2
bng#
```

3.2.30.26 show modules

Synopsis: `show modules`

Displays the current active module chain and a list of available modules.

Examples:

```
bng# show modules
current module chain:
    benchmark
  supported module chains:
    benchmark
    vrrp,arp,ping,hc,master,slb,tnat,nat,rt
    vrrp,arp,ping,hc,master,llb
  available modules:
    arp         - ARP request/reply processing
    benchmark   - HW loopback benchmark
    crossover   - virtual crossover cable
    hc          - health-check processing
    ipallow     - simple IP address allow filter
    ipdeny      - simple IP address denial filter
    llb         - link load-balancing
    macallow    - simple MAC address allow filter
    macdeny     - simple MAC address denial filter
    master      - proceed only if current master
```

```
    nat         - NAT processing
    out         - packet output processing (EXPERIMENTAL)
    ping        - ICMP echo request processing
    rt          - IPv4 and IPv6 routing module
    slb         - server load balancing
    strict      - strict IP packet acceptance filter (EXPERIMENTAL)
    tnat        - target NAT module
    vrrp        - initial VRRP handling
    ip4debug    - IPv4 debugging module (EXPERIMENTAL)
    sctpdebug   - SCTP debugging module (EXPERIMENTAL)
```

3.2.30.27 show module <module>

Synopsis: `show module <module>`

This command shows module specific information for the specified module (if available).

3.2.30.28 show nat

Synopsis: `show nat`

Displays the current active NAT table contents (Network Address Translation). The NAT table contains separate entries for TCP and UDP network address translations.

Examples:

```
bng# show nat
  TCP: 0 entries
  UDP: 0 entries
bng#

bng# show nat
  TCP: 1 entry
    ip-address      iport  oport  age
    --------------- -----  ----- ----
    172.17.2.4      32853  20012    3
  UDP: 2 entries
    ip-address      iport  oport  age
    --------------- -----  ----- ----
    172.17.2.4      32768  20573    3
    172.17.2.4        123  20572  159
bng#
```

3.2.30.29 show network <n>

Synopsis: `show network <n>`

Displays an overview of the current parameters of network <n>.

3.2.30.30 show networks

Synopsis: `show networks`

Displays current parameters of all networks.

3.2.30.31 show nodeid

Synopsis: `show nodeid`

Display the nodeid of the current BalanceNG instance, which is a 6 byte identification being represented in Ethernet address format. This nodeid is being used to identify the BalanceNG instance for licensing purposes.

The nodeid is derived from the Ethernet address of the eth0 interface, the instance number and other system parameters.

Example:

```
bng# show nodeid
  13:e3:ac:12:9d:47
bng#
```

The nodeid can change due to hardware replacement or extensions (e.g. due Ethernet interface "reordering"). Please goto http://www.inlab.de for further licensing information and relicensing procedures.

Note: All BalanceNG instances have the same nodeid.

See also "`license`" and "`show license`".

3.2.30.32 show nous

Synopsis: `show nous`

Displays the number of unsynchronized sessions. If this command displays “0” on the current VRRP master and the parameter sessionsyncack is set to 1 (enabled), the session table of the VRRP backup is in sync with the session table of the VRRP master.

3.2.30.33 show parameters

Synopsis: `show parameters`

This shows the settable parameter of BalanceNG. Please see the set command for detailed parameter explanations.

Example:

```
bng# show parameters
  name                min     max default current
  ------------------ ---- ------- ------- -------
  arplookup             5      60      10      10
  arprefresh           60    3600     300     300
  arptimeout            0   86400       0       0
  backupalerts          0       1       1       1
  bmduration           10   86400     300     300
  bmpsize              20    1514    1514    1514
  bmwsize               1   10000     128     128
  bngfilter             0       1       1       1
  debugscope            0       9       0       0
  dumprotation          1 1048576    1024    1024
  gnatdlimit           10  100000      10      10
  gratarpremind         0     120       0       0
  hashbytes4            1       4       4       4
  hashbytes6            1      16      16      16
```

```
  hcportoffset      1024   65535   30000   30000
  ipforwarding         0       1       0       0
  localdsr             0       1       0       0
  localvirt            0       1       0       0
  maxsyncps            0   10000       0       0
  multithreading       0       1       0       1
  natdlimit           10     500      10      10
  natscan              1      20      10      10
  natsync              0       1       1       1
  natsynciv           10     120      10      10
  nattimeout          10  172800     600     600
  outmtu               0    1514       0       0
  psvrelearn           0       1       0       0
  sendprobes           0       1       0       0
  sessionautoresync    0       1       0       0
  sessionarrtimeout    0    3600      60      60
  sessiongclimit    1000  500000  100000  100000
  sessiondlimit       10    1000      10      10
  sessionscan          1      20      10      10
  sessionscanbup       1    1000     100     100
  sessionsync          0       1       1       1
  sessionsyncack       0       1       0       0
  sessionsyncetype     0       1       0       0
  sessionsynciv       10     120      10      10
  sessiontimeout      10  172800     600     600
  snattimeout         10  172800    1800    1800
  syncackbdelay        1      60      10      10
  syncackmaxps         1   10000    2000    2000
  syncackresend        1      60       5       5
  syncackwsize         1   10000     100     100
  stickytarget         0       1       1       1
  strictrouting        0       1       1       1
  vrrpmasterdown       3      10       3       3
  vrrppreempt          0       1       0       0
  vrrppreemptts        0     100       0       0
  vrrpstateplugin      0       1       0       0
```

Each parameter has a predefined minimum, maximum and default value. The NAT-specific parameters are currently not used and for future releases of BalanceNG.

3.2.30.34 show private

Synopsis: `show private`

This command shows the “private” node specific data of the current instance as it would be saved by “save private” to the primate configuration data file.

Example:

```
NodeA# show private
  //        private configuration data Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  hostname  NodeA
  vrrp      priority 209
```

```
  network   1 real 172.17.2.61
  //        end of private configuration data
NodeA#
```

3.2.30.35 show server <n>

Synopsis: `show server <n>`

Shows information about one particular server and its current state.

Example:

```
bng# show server 1
server 1
  ipaddr    10.55.55.222
  network   1
  port      any
  protocol  any
  status    enabled
  method    rr
  portrel   on
  targets   1,(5)
bng#
```

3.2.30.36 show servers

Synopsis: `show servers`

This shows an overview of configured servers and their current state including the states of the associated targets.

A target or backup target as a number is operational according to it's health checks, a target or backup target in round parentheses is not available due to health check failures, a target or backup target in square brackets is disabled by purpose.

Only enabled and previously registered server definitions are shown.

Example:

```
bng# show servers
    # ipaddr           port prt net S targets {backups}
  ---------------------------------------------------------
    1 10.55.55.222      any any   1 e 1,2,3,4,(5),[6] {}
    2 10.55.55.226      any any   1 e 8,9,(10),(11) {20}
bng#
```

In this example targets 1,2,3 and 4 of server 1 are operational. The health checks of target 5 are failing and target 6 is disabled.

Target 8 and 9 of server 2 are operational, 10 and 11 have currently failing health checks. There's an operational backup target 20 available which would be addressed as soon as there's no ordinary target available.

3.2.30.37 show sessiongroups

Synopsis: `show sessiongroups`

Information about the current target sessiongroups is displayed (see “target <n> sessiongroup” and “server <n> maxgrpsessions”).

Example:

```
bng# show sessiongroups
  grp  sessions targets
  --- --------- ----------------------------------------
    0         0 2
    7         0 1 3 4 5
```

3.2.30.38 show sessions

Synopsis: `show sessions [<session-id substring>]`

Information about current active sessions is displayed.

The number of active sessions is displayed together with the first ten active sessions (which are usually the latest ones).

It is normal for the session tables to become very huge. The session table entry timeout is stored with each session table entry (column “stout”).

An optional second argument restricts the output to the specified session-id substring. The number of displayed session table entries is restricted by the “sessiondlimit” parameter.

```
bng# show sessions
 4 sessions
  srv tgt  age timeout ftimeout SYNC session-id
  --- --- ---- ------- -------- ---- ------------------
    1   2   65    7320        0 SYNC 10.255.107.175:0
    1   5    1    7320        0 SYNC 10.255.33.129:0
    1   5 1148    7320        0 SYNC 10.255.129.138:0
    1   4  776    7320        0 SYNC 10.255.130.136:0
bng#

bng@testnode# show sessions 10.255.107.175
 4 sessions
  srv tgt  age timeout ftimeout SYNC session-id
  --- --- ---- ------- -------- ---- ------------------
    1   2   65    7320        0 SYNC 10.255.107.175:0
bng@testnode#
```

3.2.30.39 show snat

Synopsis: `show snat`

Displays information current SNAT status.

3.2.30.40 show startuplog

Synopsis: `show startuplog`

Displays the first 40 log messages that appeared after the initial startup of the instance. The output of “show startuplog” is also included in the snapshot file (see “snapshot” command).

3.2.30.41 show stinfo

Synopsis: `show stinfo`

Displays information about the internal session hashtable. Session entries are pre-allocated in chunks, the number of current allocated chunks is displayed.

Example:

```
bng# show stinfo
  bytes/session: 144
  allocated 16 chunks of 65536 entries
  current number of TOTAL sessions ......... : 1000000
  current number of NEW (unsynched) sessions :  989995
  current number of sessions waiting for ACK :       0
  current number of ACK'ed sessions ........ :   10005
```

3.2.30.42 show targets

Synopsis: `show targets`

Displays information about current registered and enabled targets and their current health check status.

Example:

```
bng# show targets
  no ipaddr         ipaddr6                  si port prt net net6 srv sessions status      info                       name
  -- -------------- ------------------------ -- ---- --- --- ---- --- -------- ----------- -------------------------- ----
   1 10.100.206.85  2001:db8:5678:aaaa::85    3  any any   3    3   1        0 operational arp:up,nd6:up,agent:1
   2 10.100.206.116 2001:db8:5678:aaaa::116   3  any any   3    3   1        1 operational arp:up,nd6:up,agent:1
   3 10.100.206.117 2001:db8:5678:aaaa::117   3  any any   3    3   1        0 down        arp:down,nd6:up,agent:down
   4 10.100.206.118 2001:db8:5678:aaaa::118   3  any any   3    3   1        1 operational arp:up,nd6:up,agent:1
   5 10.100.206.119 2001:db8:5678:aaaa::119   3  any any   3    3   1        2 operational arp:up,nd6:up,agent:1
  14 10.100.206.123 2001:db8:5678:aaaa::123   3  any any   3    3   1        0 down        arp:down,nd6:up,agent:down
  16 10.100.206.124 2001:db8:5678:aaaa::124   3  any any   3    3   1        0 down        arp:up,nd6:up,agent:down
bng#
```

3.2.30.43 show target <n>

Synopsis: `show target <n>`

Displays more detailed information about one specific target.

Example:

```
bng# show target 1
target 1
  status           operational (arp:up,script:up)
  ipaddr           172.17.2.91
  port             any
  network          1
  protocol         any
  sessions         0
  maxsessions      0
  sessiongroup     7
  grpsessions      0
  maxgrpsessions   5000
  trackval         20
  psent            0
  bsent            0
  prcvd            17
  brcvd            4862
  bwin             0
  bwout            0
  bw               0
bng#
```

3.2.30.44 show targetregistry

Synopsis: `show targetregistry`

Displays more detailed information about current registered targets and their current health check status. This is a view from the internal target registry indented for debugging purposes. This command is usually not needed and therefor not included in the output of the "help" command.

Example:

```
bng# show targetregistry
target 2
  ipaddr    172.17.2.5
  port      any
  network   1
  protocol  any
  status    down
  arp       up
  agent     down
target 1
  ipaddr    172.17.2.4
  port      any
  network   1
  protocol  any
  status    down
  arp       up
  agent     down
bng#
```

3.2.30.45 show threads

Synopsis: `show threads`

Displays informations about all active packet processing threads if multithreading is active (parameter multithreading is set to 1).

Example:

```
bng# show threads
 thread no 1
   interface 1 (eth0)
   packets processed 1332
 thread no 2
   interface 1 (eth0)
   packets processed 1342
 thread no 3
   interface 1 (eth0)
   packets processed 1303
```

3.2.30.46 show uptime

Synopsis: `show uptime`

Display the uptime of the currently running BalanceNG process in seconds.

Example:

```
bng# show uptime
  current uptime is 10812 seconds
bng#
```

3.2.30.47 show vips

Synopsis: `show vips`

Displays current configured virtual IP addresses, their associated Ethernet addresses and the associated network.

Example:

```
bng1# show vips
  ipaddr          ethaddr           n
  ----------------------------------
  10.1.1.100      06:00:0a:01:01:64 2
  10.55.55.222    00:00:5e:00:01:01 1
  10.55.55.221    00:00:5e:00:01:01 1
  10.55.55.220    06:00:52:81:05:dc 1
  10.55.55.230    00:00:5e:00:01:01 1
  10.55.55.234    00:00:5e:00:01:01 1
  10.1.1.254      00:00:5e:00:01:01 2
bng1#
```

3.2.30.48 show version

Synopsis: `show version`

Display the current version / release of this BalanceNG process.

Example:

```
bng# show version

          This is BalanceNG 5.016 (created 2022/11/08)

  Copyright (C) 2005-2017,2018 by Inlab Networks GmbH, Germany.
       All rights reserved / Alle Rechte vorbehalten.
    Visit https://www.inlab.net for further information.

bng#
```

3.2.30.49 show vnodeid

Synopsis: `show vnodeid`

Display the vnodeid of the current BalanceNG instance, which is a 6 byte identification being represented in Ethernet address format.

The vnodeid is derived from the IPv4 address of the primary interface only.

Example:

```
bng# show vnodeid
  21:06:5e:24:7a:f3 (interface eth0)
bng#
```

Since the vnodeid is based on IPv4 address information only it is suitable for virtual machines which may be moved from one physical hardware to another.

See also "`license`" and "`show license`".

3.2.30.50 show vrrp

Synopsis: `show vrrp`

Displays the current VRRP configuration and the current state of the VRRP node (as defined in rfc3768 with an additional "OFF"-State when deactivated).

Example 1 (node is master for 2 virtual addresses):

```
bng# show vrrp
  state     MASTER
  vrid      1
  priority  200
  ip00      10.2.2.3
  ip01      10.2.2.4
```

Example 2 (vrrp is not active):

```
bng# show vrrp
  state     OFF
```

Example 3 (node is backup for 2 virtual addresses):

```
bng# show vrrp
  state     BACKUP
  vrid      1
  priority  200
  ip00      10.2.2.3
  ip01      10.2.2.4
```

3.2.31 shutdown

Synopsis: `shutdown`

This is an alias for "stop.

3.2.32 snapshot

Synopsis: `snapshot <file>`

This command collects various important system data in text format into the supplied file. This file is intended to be sent to support staff for in detail analysis in case of problems.

Only a partial session table dump is generated according to the sessiondlimit parameter (see also “snapshot-full” below).

Example:

```
bng# snapshot /tmp/snapshot.txt
collecting OS data    ...... done.
dumping configuration ...... done.
collecting BNG data   ...... done.
dumping sessiontable  ...... done.
bng#
```

3.2.33 snapshot-full

Synopsis: `snapshot-full <file>`

This command collects a snapshot with a full sessiontable dump.

3.2.34 snapshot-light

Synopsis: `snapshot-light <file>`

This command is a synonym for “snapshot”.

3.2.35 stfill

Synopsis: `stfill <number-of-sessions>`

This command fills the session table with the specified number of dummy session table entries. This is intended for testing and debugging purposes only.

Example:

```
bng#
bng# stfill 1000000
bng# sh sessions
  1000000 sessions
  srv tgt  age timeout ftimeout SYNC session-id
  --- --- ---- ------- -------- ---- ------------------
    1   1    3    6000        0      TEST-DATA-0000999999
    1   1    3    6000        0      TEST-DATA-0000999998
    1   1    3    6000        0      TEST-DATA-0000999997
    1   1    3    6000        0      TEST-DATA-0000999996
    1   1    3    6000        0      TEST-DATA-0000999995
    1   1    3    6000        0      TEST-DATA-0000999994
    1   1    3    6000        0      TEST-DATA-0000999993
    1   1    3    6000        0      TEST-DATA-0000999992
    1   1    3    6000        0      TEST-DATA-0000999991
    1   1    3    6000        0      TEST-DATA-0000999990
    1   1    3    6000        0      TEST-DATA-0000999989
  ... remaining sessions not shown
bng#
```

3.2.36 stop

Synopsis: `stop`

Immediately stops the BalanceNG program. This has the same effect as "`bng stop`" on the command line. There's no additional confirmation required.

Example:

```
bng# stop
ok
BalanceNG: no peer available
#
```

3.3 Configuration Commands

The command described in this chapter are actually changing the current configuration of

BalanceNG immediately.

Some commands are revertable using the "no" special command.

3.3.1 ! <command>

Synopsis: `! <command>`

This special command executes the supplied command as expected, but suppresses the output of the interactive prompt afterwards. This is useful for interfacing programs to BalanceNG (like Web User Interfaces).

Example:

```
root# bng auxctl << EOF
> ! show targets
> EOF
    # ipaddr           port prt net srv sessions status      name
  --- ---------------- ---- --- --- --- -------- ----------- ---------------
    1 172.17.2.90       any any   1   0        0 operational
root#
```

3.3.2 arp

Synopsis: `[no] arp <ip4-address>`

This command adds an IPv4 address to the list of addresses that are being "arp resolved". The mac (Ethernet) address of this IP address will be retrieved using the ARP protocol.

The operation can be reverted on the command line with the "`no`" prefix.

If the mac address is unknown, the retrieval will be tried all "`arplookup`" seconds (default 10 seconds). See also parameter summary at the `set` command.

If the mac address is already known, then it will be refreshed after "`arprefresh`" seconds (default 120).

As soon as a target is being activated (registered and enabled) an implicit (and invisible) arp command line is inserted for the target ip address into the running configuration.

A target always has an implicit "base" ARP health check, which times out after "`arptimeout`" seconds (default 0, which disables this permanent healthcheck).

ARP requests are being issued on all associated interfaces with an anonymous IP-Source address per default.

Note: You usually don't have to enter arp command lines into the configuration, the required administrative ARP resolving for targets and Servers is done automatically by BalanceNG.

Example:

```
NodeA# show arphash
  ipaddr          ethaddr            tgt net  cntr   age flags
  --------------- ----------------- ---- --- ----- ----- -----------
  172.17.2.61     06:00:ac:11:02:3d    -   1     -     - vip fix
  172.17.2.64     00:00:5e:00:01:0e    -   1     -     - vip
  172.17.2.91     00:e0:81:5d:2a:65    1   -   119   108
NodeA# arp 172.17.2.74
NodeA# show arphash
  ipaddr          ethaddr            tgt net  cntr   age flags
  --------------- ----------------- ---- --- ----- ----- -----------
  172.17.2.61     06:00:ac:11:02:3d    -   1     -     - vip fix
  172.17.2.64     00:00:5e:00:01:0e    -   1     -     - vip
```

```
  172.17.2.74     00:00:00:00:00:00    -   -     2     - vis
  172.17.2.91     00:e0:81:5d:2a:65    1   -   158   147
NodeA#
```

3.3.3 commit

Synopsis: `commit [network(s)|server(s)|target(s)] <number(s)>`

This command registers and enables one or more networks, servers or targets at once. "`commit <item> <list>`" is equivalent to "`register <item> <list>`" and then executing "`enable <item> <list>`".

The network, server or target numbers in the list are separated by commas. This command is meant to be an abbreviation for interactive use.

See also the Server and Target state description at the "`register`" command.

Example:

```
test# /etc/init.d/bng control
BalanceNG: connected to PID 7957
bng# interface eth0
interface eth0 successfully attached
bng# network 1 addr 10.2.2.0
bng# network 1 mask 255.255.255.0
bng# network 1 real 10.2.2.20
bng# network 1 virt 10.2.2.21
bng# network 1 interface eth0
bng# commit network 1
bng# target 1 ipaddr 10.2.2.10
bng# target 2 ipaddr 10.2.2.20
bng# commit targets 1,2
WARNING: target 1 in enabled state but not referenced
WARNING: target 2 in enabled state but not referenced
bng# show conf
  //        configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  interface eth0
  network   1 {
            addr 10.2.2.0
            mask 255.255.255.0
            real 10.2.2.20
            virt 10.2.2.21
            interface eth0
  }
  register  network 1
  enable    network 1
  target    1 {
            ipaddr 10.2.2.10
  }
  target    2 {
            ipaddr 10.2.2.20
  }
  register  targets 1,2
  enable    targets 1,2
```

```
  //        end of configuration
bng#
```

3.3.4 disable

Synopsis: `disable [network(s)|server(s)|target(s)] <list>`

This command immediately disables the specified networks, servers or targets. The server's virtual IP addresses are immediately unreachable, targets are immediately taken out of any load balancing distribution, networks are completely taken out of BalanceNG processing. The corresponding "`enable`"-entries are removed from the running configuration.

Note: The networks, servers or targets remain "registered", for changing and editing parameters they first have to be taken to the "unregistered" state using the "`unregister`" command (not necessary for networks, they may be edited and changed in the registered state).

See also the Server and Target state description at the "`register`" command.

Example:

```
bng# show conf
  //        configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  interface eth0
  network   1 {
            addr 10.2.2.0
            mask 255.255.255.0
            real 10.2.2.20
            virt 10.2.2.21
            interface eth0
  }
  register  network 1
  enable    network 1
  target    1 {
            ipaddr 10.2.2.10
  }
  target    2 {
            ipaddr 10.2.2.20
  }
  register  targets 1,2
  enable    targets 1,2
  //        end of configuration
bng# disable targets 1,2
bng# show conf
  //        configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  interface eth0
  network   1 {
            addr 10.2.2.0
            mask 255.255.255.0
            real 10.2.2.20
            virt 10.2.2.21
            interface eth0
  }
```

```
  register  network 1
  enable    network 1
  target    1 {
            ipaddr 10.2.2.10
  }
  target    2 {
            ipaddr 10.2.2.20
  }
  register  targets 1,2
  //        end of configuration
bng# show targets
    # ipaddr           port prt net srv status
  ------------------------------------------------------------
    1 10.2.2.10         any any   1   0 disabled
    2 10.2.2.20         any any   1   0 disabled
bng#
```

3.3.5 dump

`Synopsis: dump <interface> <directory>`

This immediately starts dumping of all traffic going through `<interface>` to dumpfiles in the given directory. Both inbound and outbound packets are being recorded.

The “dump” functionality is not available if the multithreading scheduler is being used (with “set multithreading 1”).

BalanceNG stores the packets in "pcap" format, the data can be immediately investigated using usual tools e.g. like *snoop*, *tcpdump*, *netgrep* and/or *wireshark*.

The filename is build up as follows (containing the time of creation):

`"eth"<interface number>.<year><month><day><hour><minute><second>`

E.g. it could look like:

`eth0.20080318140703`

There's an automatic logfile rotation build in which closes the file and opens a new one with a new timestamp. When this happens is controlled by the parameter "`dumprotation`" which specifies a byte threshold (see "`set`" and "`show parameters`").

Example:

```
bng# dump eth0 /bigscratch
```

3.3.6 edit

Synopsis: `edit [network(s)|server(s)|target(s)] <list>`

This command disables and unregisters one or more networks, servers or targets at once. "edit `<item> <list>`" is equivalent to "`disable <item> <list>`" and then executing "`unregister <item> <list>`".

The item numbers in the list are separated by commas.

See also the Server and Target state description at the "`register`" command.

Example:

```
bng# show conf
```

```
  //        configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  interface eth0
  network   1 {
            addr 10.2.2.0
            mask 255.255.255.0
            real 10.2.2.20
            virt 10.2.2.21
            interface eth0
  }
  register  network 1
  enable    network 1
  target    1 {
            ipaddr 10.2.2.10
  }
  target    2 {
            ipaddr 10.2.2.20
  }
  register  targets 1,2
  enable    targets 1,2
  //        end of configuration
bng# edit targets 1,2
bng# show conf
  //        configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  interface eth0
  network   1 {
            addr 10.2.2.0
            mask 255.255.255.0
            real 10.2.2.20
            virt 10.2.2.21
            interface eth0
  }
  register  network 1
  enable    network 1
  //        end of configuration
bng# show targets
    # ipaddr           port prt net srv status
  ------------------------------------------------------------
bng#
```

3.3.7 enable

Synopsis:

```
  enable [interface(s)|network(s)|server(s)|target(s)] <list>
```

This command enables the specified interfaces, networks, servers or targets. Targets in "enabled" mode (or state) immediately participate in load balancing distributions. Servers in "enabled" mode start answering ARP requests and ICP-ECHO requests ("pings"). Both servers and targets have to be members of enabled networks.

If a virtual network address is locally occupied and used by the operating system an error message is generated and the enabling is refused. This applies to “server ipaddr”, “network real” and “network virt” addresses (which must not be used in parallel by the underlying

operating system).

This command is revertable by the appropriate "`disable`" command.

See the Server and Target state description at the "`register`" command.

Example:

```
bng# show conf
  //        configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  interface eth0
  network   1 {
            addr 10.2.2.0
            mask 255.255.255.0
            real 10.2.2.20
            virt 10.2.2.21
            interface eth0
  }
  register  network 1
  enable    network 1
  target    1 {
            ipaddr 10.2.2.10
  }
  target    2 {
            ipaddr 10.2.2.20
  }
  register  targets 1,2
  //        end of configuration
bng# show targets
    # ipaddr           port prt net srv status
  ------------------------------------------------------------
    1 10.2.2.10         any any   1   0 disabled
    2 10.2.2.20         any any   1   0 disabled
bng# enable targets 1,2
WARNING: target 1 in enabled state but not referenced
WARNING: target 2 in enabled state but not referenced
bng# show targets
    # ipaddr           port prt net srv status
  ------------------------------------------------------------
    1 10.2.2.10         any any   1   0 down
    2 10.2.2.20         any any   1   0 down
bng# show conf
  //        configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  interface eth0
  network   1 {
            addr 10.2.2.0
            mask 255.255.255.0
            real 10.2.2.20
            virt 10.2.2.21
            interface eth0
  }
  register  network 1
```

```
  enable    network 1
  target    1 {
            ipaddr 10.2.2.10
  }
  target    2 {
            ipaddr 10.2.2.20
  }
  register  targets 1,2
  enable    targets 1,2
  //        end of configuration
bng#
```

3.3.8 gateway

BalanceNG may use a default gateway, which may be specified using the gateway configuration command family. A DSR configuration does usually not need a gateway so it doesn't need to be specified in that case.

3.3.8.1 gateway <ip4addr>

Synopsis: `gateway <ip4addr>`

This command is a synonym for "`gateway ipaddr <ip4-address>`" and is implemented to maintain backwards compatibility to older releases of BalanceNG.

3.3.8.2 gateway alert <script>

Synopsis: `gateway alert <script>`

This command specifies an external notification script which is called as soon as the specified gateway becomes down (not operational). For removing the script the empty string (“”) needs to be specified.

Example:

```
bng# gateway alert "/usr/bin/logger -p daemon.notice GATEWAY IS DOWN"
bng#
```

3.3.8.3 gateway arp <interval>,<timeout>

Synopsis:
```
gateway arp <interval>,<timeout>
gateway arp off
gateway arp disable
```

This command establishes an "arp" healthcheck of the gateway. An arp request is sent out every <interval> seconds, the gateway state changes to "down" if an arp reply has not been received for <timeout> seconds. The special form "gateway arp off" (or "gateway arp disable") removes the arp healthcheck from the configuration.

The arp gateway health check may exist in parallel to the arp ping healthcheck.

Example:

```
bng# gateway arp 2,5
bng# show gateway
       ipaddr: 172.17.2.254
 total status: operational
  ping status: up
   arp status: up
```

3.3.8.4 gateway ipaddr <ip4addr>

Synopsis: `gateway ipaddr <ip4addr>`

Specifies the default gateway towards the external world (e.g. "Internet"). If an IP4-address is not locally connected (not part of a local network definition) the packet is being sent to this gateway address for routing.

ARP-resolution of the gateway address is being processed automatically by BalanceNG.

The command "`no gateway`" removes the gateway declaration from the current configuration.

Example:

```
bng# gateway ipaddr 10.2.2.254
bng# show conf
  //        configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  interface eth0
  network   1 {
            addr 10.2.2.0
            mask 255.255.255.0
            real 10.2.2.20
            virt 10.2.2.21
            interface eth0
  }
  register  network 1
  enable    network 1
  gateway   {
            ipaddr 10.2.2.254
  }
  target    1 {
            ipaddr 10.2.2.10
  }
  target    2 {
            ipaddr 10.2.2.20
  }
  register  targets 1,2
  enable    targets 1,2
  //        end of configuration
bng#
```

3.3.8.5 gateway ipaddr6 <ip6addr>

Synopsis: `gateway ipaddr6 <ip6addr>`

Specifies the default IPv6 gateway towards the external world. If an IP6-address is not locally connected (not part of a local network definition) an IPv6 packet is being sent to this gateway address for routing.

ND-resolution (Neighbor Discovery) of the gateway IPv6 address is being processed automatically by BalanceNG.

The command "`no gateway`" removes all gateway declarations from the current configuration (ipaddr and ipaddr6).

Example:

```
bng# gateway ipaddr6 fe80::230:48ff:fe93:4302
```

3.3.8.6 gateway nd6 <interval>,<timeout>

Synopsis:
```
gateway nd6 <interval>,<timeout>
gateway nd6 off
gateway nd6 disable
```

This command establishes an IPv6 Neighbor Discovery healthcheck towards the gateway IPv6 address (ipaddr6).

3.3.8.7 gateway ping <interval>,<timeout>

Synopsis:
```
gateway ping <interval>,<timeout>
gateway ping off
gateway ping disable
```

This command establishes a "ping" healthcheck of the gateway. An ICMP echo request is sent out every <interval> seconds, the gateway state changes to "down" if an ICMP echo reply has not been received for <timeout> seconds. The special form "gateway ping off" (or "gateway ping disable") removes the ping healthcheck from the configuration.

The ping gateway health check may exist in parallel to the arp healthcheck.

Example:

```
bng# gateway ping 1,3
bng# show gateway
       ipaddr: 172.17.2.254
 total status: operational
  ping status: up
 arp status: up
```

3.3.8.8 gateway ping6 <interval>,<timeout>

Synopsis:
```
gateway ping6 <interval>,<timeout>
gateway ping6 off
gateway ping6 disable
```

This command establishes a ping IPv6 healthcheck, the ipaddr6 IPv6 address must be present.

3.3.8.9 gateway trackval <value>

Synopsis:
```
gateway trackval <value>
gateway trackval default
```

This command associates a tracking value to the gateway. If the gateway state changes to "down" (according to the current active health checks) and VRRP is active and the current priority is not equal to 255 then the current priority is degraded by the tracking value of the gateway. The default value is 0, the special form "gateway trackval default" resets the gateway tracking value to 0.

Example:

```
bng# show gateway
       ipaddr: 172.17.2.254
 total status: operational
```

```
  ping status: up
bng# gateway trackval 4
bng# show gateway
       ipaddr: 172.17.2.254
 total status: operational
  ping status: up
     trackval: 4
bng#
```

3.3.8.10 gateway upalert <script>

Synopsis: `gateway upalert <script>`

This command specifies an external notification script which is called as soon as the specified gateway becomes operational. For removing the script the empty string ("") needs to be specified.

Example:

```
bng# gateway upalert "/usr/bin/logger -p daemon.notice GATEWAY IS UP"
bng#
```

3.3.9 hostname

Synopsis: `[no] hostname <name>`

This sets the current hostname of the BalanceNG instance to the specified name and changes the interactive command line prompt to that name followed by a "#". This is just for informational purposes and does not change functionality.

This command may be reverted by the "no" keyword, changing the name to "undefined" and changing the interactive command line prompt back to the default (which is "bng#").

The hostname may contain spaces, in that case the hostname must be embedded in double quotes. Specifying an empty string in double quotes also sets the hostname to the initial "undefined" state.

Example:

```
# bng control
  BalanceNG: connected to PID 8660
bng# show conf
  //        configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  //        end of configuration
bng# hostname test1
test1# show conf
  //        configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  hostname  test1
  //        end of configuration
test1# no hostname test1
bng# show conf
  //        configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  //        end of configuration
bng# hostname "LoadBalancer Side A"
```

```
LoadBalancer Side A# show conf
  //        configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  hostname  "LoadBalancer Side A"
  //        end of configuration
LoadBalancer Side A# hostname ""
bng# show conf
  //        configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  //        end of configuration
bng#
```

3.3.10 interface <name>

Synopsis: `interface <interface specifier>`

This command is implemented for compatibility reasons with older BalanceNG versions, where “interface <name>” is an abbreviation for the following three commands (using the next free numerical index):

```
interface <n> name <name>
register interface <n>
enable interface <n>
```

3.3.11 interface <n>

3.3.11.1 interface <n> access

Synopsis: `interface <n> access raw`

`interface <n> access tap`

`interface <n> access vpn`

This directive defines the access method for the specified interface. Access method “raw” is the default and enables direct “raw” access to the Ethernet Layer-2 network. Access method “tap” creates a virtual Layer-2 NIC and connects BalanceNG to the outer side. Access method “vpn” connects BalanceNG to a remote interface using the RBridge UDP protocol.

Note: Access method “tap” is not available on macOS platforms.

3.3.11.2 interface <n> alert

Synopsis: `interface <n> alert “<script>”`

`interface <n> alert none`

This directive allows to specify a script, which is called in the event of link loss on this particular interface. The link detection runs once per second in the background. Specifying “none” removes and disables the external alert script (if in place). The special variable “$name$” is replaced by the interface name.

```
bng# interface 1 alert "/usr/bin/logger -p daemon.notice LINK LOST ON INTERFACE $name$"
bng#
```

3.3.11.3 interface <n> init

Synopsis: `interface <n> init <commands>`

This configurational command allows to specify initialization commands which are executed by the operating system shell at “enable interface”.

Example:

This configuration file excerpt declares an bng0 “TAP”-Device on Linux and initializes the interface bng0 with a specific IP address.

```
interface 1 {
          name eth0
}
interface 2 {
          name bng0
          init "ip addr add 172.17.2.60/24 dev bng0; ip link set bng0 up"
}
register  interfaces 1,2
enable    interfaces 1,2
```

3.3.11.4 interface <n> modules

Synopsis: `interface <n> module <module>`

`interface <n> modules <module1>, …`

This command defines an interface-specific module chain which is executed on that particular interface. This module chain is executed by the “imc” module if the “imc” module is included in the main module chain.

The “imc” module itself may not be part of the interface specific module chain.

3.3.11.5 interface <n> name

Synopsis: `interface <n> name <name>`

This configurational command specifies the OS interface to be used by the BalanceNG interface with the specified number.

On Linux the special names “bng0” up to “bng9” define a “TAP” device which allows BalanceNG to communicate directly with the Linux TCP/IP Stack.

The HW interface is brought into promiscuous mode as soon as the numerical BalanceNG interface is both registered and enabled.

Example:

```
# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:0E:0C:6C:BA:59
          inet addr:10.55.55.12  Bcast:10.55.55.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:5045 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3111 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:740063 (722.7 Kb)  TX bytes:1215602 (1.1 Mb)
          Base address:0xc000 Memory:df020000-df040000
# /etc/init.d/bng start
BalanceNG: starting up ...
# /etc/init.d/bng control
BalanceNG: connected to PID 8012
bng# interface 1 name eth0
bng# commit interface 1
bng#
```

3.3.11.6 interface <n> scope

Synopsis: `interface <n> scope <value>`

This command declares the scope or orientation of the interface which may be used by a specific module.

“interface <n> scope external” declares the interface to be oriented towards the external world (or a router), “interface <n> scope internal” does the opposite.

The “internal” setting is the default which is not shown in the configuration if active.

Please see the module descriptions referencing this declaration (e.g. Ifilter and haswitch).

3.3.11.7 interface <n> threads

Synopsis: `interface <n> threads <value>`

This command specifies the number of packet processing threads per interface. The default value is 1 (one packet processing thread per interface). This command directive is effective only when multithreading is activated (“set multithreading 1”). The current configurable maximum of threads per interface is 8.

3.3.11.8 interface <n> switching

Synopsis: `interface <n> switching <value>`

This command controls if the specified interface is being considered for processing by a switching module. The parameter “enable” enables it, “disable” does the contrary and is the default setting.

If the default setting is active, the setting does not appear in the configuration (as usual).

3.3.11.9 interface <n> trackval

Synopsis: `interface <n> trackval <value>`

This command associates a VRRP tracking value to an interface, which is deducted if an enabled interface loses its link (determined by the automatic link detection).

3.3.11.10 interface <n> upalert

Synopsis: `interface <n> upalert “<script>”`

`interface <n> upalert none`

This directive allows to specify a script, which is called in the event of link availability on this particular interface. The link detection runs once per second in the background. Specifying “none” removes and disables the external upalert script (if in place). The special variable “$name$” is replaced by the interface name.

```
bng# interface 1 alert "/usr/bin/logger -p daemon.notice LINK DETECTED ON INTERFACE $name$"
bng#
```

3.3.12 ipallow

Synopsis: `ipallow <ipaddr>`

`no ipallow`

This directive allows to add IP addresses to the ipallow list. IPv4 addresses have to be specified in IPv6 notation (e.g. ::ffff:10.1.2.3). This list is used by the ipallow module.

3.3.13 ipdb

Synopsis: `[no] ipdb [filename.csv]`

This command loads a IP-to-location database (IPDB) from a file in .csv format into memory. The default location “/opt/BalanceNG/ip-to-country.csv” is used when there's no filename specified. This file is in place automatically if BalanceNG has been installed using the Solaris or Ubuntu/Debian packages.

The command “no ipdb” unloads the IPDB in memory database. Both, “ipdb <file>” and “no ipdb” may be entered interactively even during operation as an active VRRP master. During the time the database is being loaded packet processing is suspended for a short time.

The command “ipdb” understands the following two .csv formats:

A) The 5-column format as found in the file “ip-to-country.csv” as available for free from webhosting.info (URL: http://ip-to-country.webhosting.info/). A probably outdated version of this file is being distributed with the Ubuntu/Debian package distribution of BalanceNG.

B) The 7-column format as found in the file “IpToCountry.csv” as available for free from “software77.net” (URL: http://software77.net/cgi-bin/ip-country/geo-ip.pl).

Please note that Inlab is not responsible for the quality of the available database files. We recommend performing some prior analysis and testing before deploying a live location based load balancing installation. Also, it's up to the administrator to schedule updates of the chosen database file.

There's currently a hard coded maximum of IPDB entries and a current maximum of 512 locations being referenced by the IPDB (these values will be increased as necessary). Please make sure to upgrade to a current release if you have any doubts.

The IPDB reference is automatically inserted into the running configuration and will be loaded on startup or restart if made permanent (with “save”).

Example:

```
bng# ipdb
bng# show ipdb
  IPDB loaded from /opt/BalanceNG/ip-to-country.csv
  83429 valid 5-column lines
  83429 total IPDB entries available
  no consecutive area overlaps
  235 different IPDB locations referenced
bng# no ipdb
bng# show ipdb
  no IPDB entries available
bng# ipdb /tmp/IpToCountry.csv
bng# show ipdb
  IPDB loaded from /tmp/IpToCountry.csv
  86569 valid 7-column lines
  86569 total IPDB entries available
  2 consecutive area overlaps
  222 different IPDB locations referenced
bng# show conf ipdb
  ipdb      "/tmp/IpToCountry.csv"
bng#
```

3.3.14 ipdeny

Synopsis: `ipdeny <ipaddr>`

`no ipdeny`

This directive allows to add IP addresses to the ipdeny list. IPv4 addresses have to be specified in IPv6 notation (e.g. ::ffff:10.1.2.3). This list is used by the ipdeny module.

3.3.15 ipdb6

Synopsis: `[no] ipdb6 [filename.csv]`

This command loads a IPv6-to-location database (IPDB6) from a file in .csv format into memory. The default location "/opt/BalanceNG/IpToCountry.6R.csv" is used when there's no filename specified. This file is in place automatically if BalanceNG has been installed using the Ubuntu/Debian packages.

The command "no ipdb6" unloads the IPDB6 in memory database. Both, "ipdb6 <file>" and "no ipdb6" may be entered interactively even during operation as an active VRRP master.

During the time the database is being loaded packet processing is suspended for a short time.

The command "ipdb6" accepts the 5-column format as found in the file "IpToCountry.6R.csv" as available for free from software77.net (URL: http://software77.net/geo-ip/).

3.3.16 lgrp

Synopsis:
```
lgrp <A-Z> <specification>
no lgrp <A-Z>
no lgrp
```

This command specifies a location group based on the set of locations as found in the IP-to-location database (IPDB). A location group is being referenced by a single, capital letter such allowing 26 location groups per BalanceNG instance. A target may be associated with exactly one location group (see "target <n> lgrp").

The specification of a location group may contain one of the following tokens, separated by commas:

1) A dual character location reference which declares to include that location in the location group (allowed characters are [A-Z] and [0-9]).
2) A single group letter ([A-Z]) referencing a different location group to be included to the specified group.
3) The special character "*" declaring "all possible locations" including locations not found in the database.
4) The special character "-" referencing the "not found" pseudo location.

All four token types may be preceded by a "!", which specifies that the location set is **not** part of the location group. Recursion and self-referencing of location groups is not allowed.

Examples:

`lgrp A "DE,AT,CH"`	Location group A should include locations DE,AT and CH.
`lgrp B "*,!A"`	Location group B should include "everything" but not lgrp A.
`lgrp C "-"`	Location group C should include only "not found" entries.
`lgrp D "A,B,!C"`	Location group B should include group A, group B, but not location group C.

3.3.17 license

Synopsis: `license <serial-number> <license-key>`

This command fully activates BalanceNG using the purchased license for exactly this node. The serial number and license key is being provided by Inlab Networks GmbH. The license-key is only valid for exactly one specific node identified by its nodeid (see also "`show`

nodeid").

BalanceNG with no or without valid licensing information runs for 30 days with a full functionality trial license.

The effect of the "license" command can be examined with "show license".

Example:

```
bng# show license
  no license specified
bng# show nodeid
  fa:e9:55:26:66:a8
bng# license beta123 e0bf3035c3c673ee725eae1bbb30c31b
bng# show license
  status: valid full license
  serial: TEST0611021
  nodeid: 2a:e2:9f:38:da:20
  type "show version" for version and Copyright information
bng#
```

3.3.18 log

Synopsis: log <message>

This command logs a specific message to the BalanceNG log.

Example:

```
bng# log “now starting with upgrade”
bng#
```

3.3.19 macallow

Synopsis: macallow <MAC address>

no macallow

This command enters the MAC address into the set of “macallow” addresses. If the module “macallow” is in the module chain, all packets with a source MAC address out of this set will be forwarded and ***all other packets will be discarded***. “no macallow” empties the complete list.

3.3.20 macdeny

Synopsis: macdeny <MAC address>

no macdeny

This command enters the MAC address into the set of “macdeny” addresses. If the module “macdeny” is in the module chain, all packets with a source MAC address out of this set will be discarded. “no macdeny” empties the complete list.

3.3.21 macrouter

Synopsis: macrouter <MAC address>

no macrouter

This command allows the declaration of routing devices by specifying their MAC address. If a routing device is specified this way, updates of the internal ARP table are disabled in the special case that the source IPv4 address is already known to be reachable directly.

Example:

```
bng# macrouter 11:22:33:44:55:66
bng# macrouter 11:22:33:44:55:67
bng# show conf
...
  macrouter {
    11:22:33:44:55:66
    11:22:33:44:55:67
  }
...
bng#
```

3.3.22 modules

Synopsis: `module <module>`

`modules <module1>,<module2>,...`

This command defines the module chain which controls the core functionality of BalanceNG. The command “show modules” displays a list of supported module chains and a short description of the available modules.

BalanceNG encapsulates packet processing functionalities in modules. The module chain defines a sequential order of modules and thus controls the core functionality of the BalanceNG packet handling engine.

There are the following 5 supported standard module chains:

benchmark

This single module module chain needs to be in place for running the BalanceNG benchmark. Please note that this works only for the single threading scheduler (with “set multithreading 0”).

vrrp,arp,ping,hc,master,slb,tnat,nat,rt

This is the standard module chain for SLB (server load balancing) processing of IPv4 and IPv6 traffic.

vrrp,arp,ping,hc,master,llb

This is the standard module chain for LLB (link load balancing) of IPv4 and IPv6 traffic.

The following modules are available:

3.3.22.1 arp

The arp module answers IPv4 ARP requests and IPv6 ND (Neighbor Discovery) requests for IP addresses represented by BalanceNG. Additionally, it processes IPv4 ARP and IPv6 ND replies.

3.3.22.2 benchmark

The benchmark module has to be in place for the built-in hardware benchmark (see “benchmark” command).

3.3.22.3 crossover

This module implements a virtual crossover cable which may connect two virtual TAP interfaces. The following example dialog shows a simple configuration of a BNG instance 1,

which implements two virtual TAP interfaces on the Linux host (veth0 and veth1) and connects them with a virtual crossover cable implemented by the “crossover” module:

```
c_1_crossover# sh conf
  //        configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  hostname  c_1_crossover
  license   INLAB 7dde7c5755b64cc0bbc0767edee1d6fb
  module    crossover
  interface 1 {
            name veth0
            access tap
            threads 4
            init "ip addr add 10.0.0.1/24 dev veth0; ip link set veth0 up"
  }
  interface 2 {
            name veth1
            access tap
            threads 4
            init "ip addr add 10.0.0.2/24 dev veth1; ip link set veth1 up"
  }
  register  interfaces 1,2
  enable    interfaces 1,2
  //        end of configuration
c_1_crossover#sh inst
  this is BalanceNG instance 1
c_1_crossover#
```

3.3.22.4 haswitch

This module implements an high available (HA) layer 2 switch where the ethernet forwarding table is kept in the session table. Thus the current VRRP backup node is constantly updated with the forwarding table information.

The session key of a haswitch session table entry always starts with “$H” followed my the MAC address in readable format. The server property is always 1 (per definition), and the target value contains the interface index on which this address has been seen (or received) the last time.

The forwarding table entry timeout is 1 hour minutes (3600 seconds) and can be controlled by the haswitch:timeout parameter.

If VRRP is active, and a failover is necessary or initiated, the former backup node becomes master and informs all surrounding switches by sending out special forwarding table update packets (actually these packets are VRRP V3 advertisement packets with a variating source mac address, IPv4 or IPv6 is selected as configured for VRRP).

This module considers the “interface <n> switching” parameter and switches only among interfaces where switching has been set to “enabled”.

If parameter haswitch:isolate is set to 1 (active), LANs on interfaces with “scope internal” may only communicate with LANs on interfaces with “scope external” (but not among each other).

A typical module chain with the “haswitch” module would be the following:

```
  modules   vrrp,arp,master,haswitch
```

3.3.22.5 hc

This module encapsulates all health-check processing (IPv4 and IPv6).

3.3.22.6 imc

This module calls (if included) the interface specific module chain of an interface (if present).

The imc module itsepf may not be part of an interface specific module chain (see also the “interface <n> modules” configuration command).

This mechanism allows to express interface-specific packet processing.

3.3.22.7 ipallow

The ipallow module allows only packets to pass which source IP address is in the ipallow list of IP addresses (see ipallow command). All other packets are dropped.

3.3.22.8 ipdeny

The ipdeny module drops packets, whose source address is contained in the ipdeny list of IP addresses (see ipdeny command). All other packets are passed unchanged.

3.3.22.9 isolate

The isolate module drops all packets which originate from the MA-L address block 34:38:AF. Since all normal BalanceNG MAC addresses are within that block, the isolate module effectively blocks the ingress of packets from all possible BalanceNG instances in the network.

3.3.22.10 lfilter

The lfilter module requires loaded IPv4 and IPv6 location databases (see “ipdb” and “ipdb6”) and – obviously – needs to be inserted in a module chain to be effective.

As soon as an IPv4 or IPv6 packet is passed to the lfilter module, the lfilter module performs some checks and either drops or forwards the packet upwards. A packet that is not an IPv4 or an IPv6 packet is always forwarded unchanged.

If the receiving interface is in “scope external”, the lfilter module checks if the IP packet source address is in the location group “X” (which is hard coded) and if yes, the packet is passed upwards for further processing. If the receiving interface is in “scope internal” (the default) the destination address is checked instead.

If the IP packet is not in location group “X”, the packet is dropped.

Together with other common processing modules like “slb”, “llb” or “haswitch” this module provides outstanding control in a location based manner.

“show module lfilter” shows further module specific information (general communication statistics, packets dropped and location group settings).

The lfilter internal information and counters are not synchronized to the backup node because of efficiency reasons.

A typical module chain referencing the “lfilter” module would be the following (thus using it together with “haswitch”:

```
modules   vrrp,arp,master,lfilter,haswitch
```

3.3.22.11 llb

The “llb” module implements IPv4 and IPv6 Link Load Balancing (LLB) with automatic IPv4 and IPv6 NAT (Network Address Translation).

The parameter “ipforwarding” needs to be set to 1 (“set ipforwarding 1”) and the “llb” modules needs to be part of the module chain. Additionally, VRRP needs to be active.

All traffic received on a special “ipaddr any” virtual server (which may be the default route) is distributed among the targets, representing outgoing routers (e.g. DSL lines) in that case.

3.3.22.12 macallow

This is a positive MAC address filter.

3.3.22.13 macdeny

This is a negative MAC address filter.

3.3.22.14 master

This module passes packets to the remaining part of the chain if the instance is VRRP master.

3.3.22.15 nat

The “nat” module implements both IPv4 and IPv6 network address translation. The parameter “ipforwarding” needs to be set to 1 (“set ipforwarding 1”). Network translation state is kept in the “Generic NAT” table (GNAT) and is synchronized with a special, non-standard VRRP packet (type 5) to the backup if parameter “natsync” is set to “1” (which is the default).

VRRP needs to be active for this module and the “network virt” addresses need to be used as routing endpoints. All traffic received at “network virt” routing addresses on a network with “nat inside” are translated to the “network virt” address of the network with the “nat outside” property.

Packets of the following types are processed: IPv4 (UDP, TCP, ICMP ECHO) and IPv6 (UDP, TCP, ICMP ECHO). You may “ping” to the outside from a host which is located in a “nat inside” network either over IPv4 or IPv6.

3.3.22.16 ping

This answers IPv4 and IPv6 echo requests sent to virtual IP addresses represented by BalanceNG.

3.3.22.17 rt

This module implements IPv4 and IPv6 routing and should be placed at the very end of the module chain. The parameter “ipforwarding” needs to be set to 1 in order to enable routing.

3.3.22.18 slb

This is the IPv4/IPv6 server load balancing module.

3.3.22.19 tarpit

The tarpit module implements a generic IPv4 / IPv6 tarpit on all networks that have tarpit processing enabled (see “network <n> tarpit”). This module performs the following actions:

1. ARP and ND6 requests are answered if the address is not represented by BalanceNG itself and if it can be proven that there is no other machine representing this address at the time the ARP or ND6 request is being received.
2. ICMP4 and ICMP6 ECHO REQUESTS are answered if received on such a virtual represented address.
3. TCP open requests on any port on such a virtual represented address are processed without any further consumption of internal memory for state information by answering with a corresponding SYN-ACK TCP packet.
4. UDP packets received on any port are logged without any further action.

The position for the tarpit module is between the “master” module and the “slb” module like this:

```
modules   vrrp,arp,ping,hc,master,tarpit,slb,tnat,nat,rt
```

The following list shows the possible messages logged to the BalanceNG log and the syslog with LOG_WARNING level (4 for each protocol):

```
TARPIT IPv4 ARP_REPLY for <IPv4 addr> sent to <IPv4 addr> [<MAC addr>]
TARPIT IPv4 ECHO_REPLY for <IPv4 addr> sent to <IPv4 addr> [<MAC addr>]
TARPIT IPv4 TCP_SYNACK for <IPv4 addr>/<port> sent to <IPv4 addr>/<port> [<MAC addr>]
TARPIT IPv4 UDP_PACKET for <IPv4 addr>/<port> received from <IPv4 addr>/<port> [<MAC addr>]

TARPIT IPv6 ND6_REPLY for <IPv6 addr> sent to <IPv6 addr> [<MAC addr>]
TARPIT IPv6 ECHO_REPLY for <IPv6 addr> sent to <IPv6 addr> [<MAC addr>]
TARPIT IPv6 TCP_SYNACK for <IPv6 addr>/<port> sent to <IPv6 addr>/<port> [<MAC addr>]
TARPIT IPv6 UDP_PACKET for <Ipv6addr>/<port> received from <IPv6 addr>/<port> [<MAC addr>]
```

The address information of existing and simulated addresses is kept in the session table, thus with a valid master/backup configuration and session table synchronization the tarpit functionality becomes high available automatically.

An IP address which is known to be real, is remembered with a session table entry of the following format:

```
tarpit-real-<IP address>
```

An IP address which is currently simulated is remembered with a session table entry of the following format:

```
tarpit-TRAP-<IP address>
```

See also the parameter “tarpitrealto” and “tarpittrapto” which control the default session table expiry time for these two types of entries.

3.3.22.20 tnat

This modules implements IPv4 and IPv6 “target” NAT allowing a 1:1 network address translation optionally selectable by protocol and port (see “tnat” command).

3.3.22.21 vrrp

This module processes VRRP replies and control the VRRP status (master/backup) of the BalanceNG instance.

3.3.23 network <n>

Synopsis: `network <n> <subcommand> <value>`

This command family is used to specify network definitions in the BalanceNG configuration.

The special subcommand "{" opens a network definition block interactively, so that the first two arguments can be omitted until the block is closed with a corresponding "}". A currently open block is indicated by a "+"-sign at the end of the command line prompt (instead of a "#").

The network index may range from 1 to 10, such allowing a total of 10 network section per BalanceNG instance.

A network ca only be enable, if "addr", "mask", "real" and "virt" parameters are all specified correctly.

Example:

```
bng# network 1 {
bng+ addr 10.3.3.0
bng+ mask 255.255.255.0
```

```
bng+ real 10.3.3.10
bng+ virt 10.3.3.20
bng+ }
bng#
```

3.3.23.1 network <n> addr

Synopsis: `network <n> addr <address>`

This command specifies the network address of the specified network. The host part of the network must be all "0" bits (determined by the "mask" specification).

This address is not a virtual IP address (not "pingable") and has to be specified equally on all participating VRRP nodes.

Example:

```
bng# network 2 addr 10.20.0.0
bng# commit network 2
mask of network 2 not specified
bng#
```

3.3.23.2 network <n> interface[s]

Synopsis: `network <n> interface[s] <list>`

This command associates one or more interfaces to the network with the given number. All interfaces in the list have to be activated using the "interface" command before. Multiple networks may share the same interfaces (or interface).

In general network activities will be restricted on the specified interfaces, e.g.:

- ARP-requests will be sent out only to the specified interfaces for IP addresses in the network
- ARP-requests are only accepted (and answered) if received on a suitable interface.
- ICMP-echo requests ("ping's") will be not answered on not associated interfaces.
- VRRP multicast packets will only be sent out on interfaces associated to the VRRP network.

Example:

```
bng# interface eth0
interface eth0 successfully attached
bng# interface eth1
interface eth1 successfully attached
bng# network 2 interface eth0
bng# show networks
  #     name S netaddr        netmask        real addr      virt addr      interfaces
  ------------------------------------------------------------------------------------
  2          D 10.20.0.0      0.0.0.0        0.0.0.0        0.0.0.0        eth0
bng# network 2 interface eth0,eth1
bng# show networks
  #     name S netaddr        netmask        real addr      virt addr      interfaces
  ------------------------------------------------------------------------------------
  2          D 10.20.0.0      0.0.0.0        0.0.0.0        0.0.0.0        eth0,eth1
bng#
```

3.3.23.3 network <n> mask

Synopsis: `network <n> mask <netmask>`

This specifies the IP4 netmask for the given network.

Example:

```
bng# interface eth1
interface eth1 successfully attached
bng# network 3 addr 10.20.0.0
bng# network 3 mask 255.255.0.0
bng# commit network 3
virt of network 3 not specified
bng#
```

3.3.23.4 network <n> mask6

Synopsis: `network <n> mask6 <IPv6 netmask>`

`network <n> mask6 none`

This specifies the IPv6 netmask prefix for the given network. An IPv6 netmask needs to be specified as a network prefix in the range 1 to 128. The command “network <n> mask6 none” sets the internal mask6 value of the network to 0.

Example:

```
bng# interface eth1
interface eth1 successfully attached
bng# network 3 real6 2001:DB8::4
bng# network 3 mask6 32
bng# commit network 3
```

3.3.23.5 network <n> name

Synopsis: `network <n> name <name>`

This command specifies an optional name for the given network. Specifying "none" removes the current name definition. The network name may be embedded in double quotes to specify a name containing spaces. Specifying an empty string in double quotes also removes the current name definition.

Example:

```
bng# network 3 name test123
bng# show networks
  #     name S netaddr        netmask        real addr      virt addr      interfaces
  ------------------------------------------------------------------------------------
  3  test123 D 10.20.0.0      255.255.0.0    0.0.0.0        0.0.0.0        -
bng# network 3 name none
bng# show networks
  #     name S netaddr        netmask        real addr      virt addr      interfaces
  ------------------------------------------------------------------------------------
  3          D 10.20.0.0      255.255.0.0    0.0.0.0        0.0.0.0        -
bng#
```

3.3.23.6 network <n> nat

Synopsis: `network <n> nat inside|outside|off`

This network parameter controls the participation of the network in NAT (network address translation). If “network nat” is set to “inside”, all IP addresses in that network are being translated to the “network virt” address of the network with “network nat” set to “outside”.

Only one network may have “nat outside” set, NAT is active as soon as at least one network with “nat inside” and one network with “nat outside” exist and are enabled.

NAT table entries are synced per default between master and backup nodes (if VRRP is active). See the NAT parameters for more information.

Example:

```
nat-test# show conf
  //        configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  hostname  nat-test
  license   test123 660a24a35167cc8e69acfe223cc2d360
  interface bge0
  interface nge0
  network   1 {
            addr 10.135.110.0
            mask 255.255.255.0
            real 10.135.110.61
            virt 10.135.110.62
            nat outside
            interface bge0
  }
  network   2 {
            addr 10.17.2.0
            mask 255.255.255.0
            real 10.17.2.61
            virt 10.17.2.62
            nat inside
            interface nge0
  }
  register  networks 1,2
  enable    networks 1,2
  gateway   {
            ipaddr 10.135.110.254
            ping 5,12
  }
  //        end of configuration
net-test#
```

3.3.23.7 network <n> real

Synopsis: `network <n> real <real ip address>`

This specifies a "real" IP address for the given network. The IP address has to be "inside" the network (according to the address and netmask settings). This address is a "pingable" virtual IP address (active on the networks interfaces). It has to be unique for each BalanceNG node in a VRRP setup (it will not hop at a VRRP failover).

The specification of a "real" (node specific) address is recommended, since it is needed to make ARP-lookups and health checks work (they both need an originating address).

It is being used for the following purposes:

- ARP-requests are using it as the source address resolving IP addresses in the network
- Health-checks will use it as the source address checking targets in this particular network.

Example:

```
bng# network 3 real 10.20.3.3
bng# show networks
  #     name S netaddr        netmask        real addr      virt addr      interfaces
  ------------------------------------------------------------------------------------
  3          D 10.20.0.0      255.255.0.0    10.20.3.3      0.0.0.0        -
bng#
```

3.3.23.8 network <n> real6

Synopsis: `network <n> real6 <real IPv6 address>`

This specifies a "real" IPv6 address for the given network.

3.3.23.9 network <n> syncface

Synopsis: `network <n> syncface <interface>`

This specifies a dedicated interface, where session synchronization traffic and NAT synchronization traffic should be directed.

Using this, a dedicated crossover cable may be put between two BalanceNG VRRP nodes in order to separate this traffic from everything else. This command is only effective in a network section which is referenced by the VRRP section.

Example:

```
interface eth1
interface eth2
vrrp {
  ...
  network 1
  ...
}

network 1 {
  ...
  interface eth1
  syncface eth2
  ...
}
```

3.3.23.10 network <n> syncpeer

Synopsis: `network <n> syncpeer <address>`

This specifies a IPv4 or IPv6 address which is being used for the bngsync protocol. If a IPv6 address is provided, the bngsync protocol uses IPv6 transport, otherwise IPv4 is being used.

Specifying an “syncpeer” address activates the BalanceNG synchronization over bngsync (thus disabling the use if the non-standard VRRP extensions).

This command is only effective in a network section which is referenced by the VRRP section.

Example:

```
interface eth1

vrrp {
  ...
  network 1
  ...
}

network 1 {
  ...
  interface eth1
```

```
  syncpeer 10.1.1.1
  ...
}
```

3.3.23.11 network <n> tarpit

Synopsis: `network <n> tarpit enable|disable`

This enables tarpit processing on the specified network and is functional if the tarpit module is part of the active module chain.

3.3.23.12 network <n> virt

Synopsis: `network <n> virt <address>`

This specifies the virtual address in the given network. This is a "pingable" address that **is being shared between all VRRP nodes**.

The specification of a network "virt" address is not mandatory (using 0.0.0.0 as "unspecified").

This should be used as a routing endpoint (gateway) for all hosts in tat network, e.g.:

- Default gateway for targets in that network
- Gateway for targets in that network using a specific routing rule.

Example:

```
bng# network 3 virt 10.20.4.4
bng# enable network 3
bng# show networks
  #     name S netaddr        netmask        real addr      virt addr      interfaces
  ------------------------------------------------------------------------------------
  3          E 10.20.0.0      255.255.0.0    10.20.3.3      10.20.4.4      -
bng#
```

3.3.23.13 network <n> virt6

Synopsis: `network <n> virt6 <IPv6 address>`

This specifies the virtual IPv6 address for the given network. This address is represented by all VRRP nodes and may be used as a routing endpoint.

3.3.24 no

Synopsis: `no <command>`

This special command reverts another command and eventually removes the configuration command out of the current configuration. This applies to a subset of commands, the following commands are revertable by "no" (in alphabetical order):

`arp`, `dump`, `gateway`, `hostname`, `license, tnat` and `vip`.

The arguments of the original command are only checked if necessary when reverting it with "no".

Example:

```
bng# show conf
  //        configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  //        end of configuration
```

```
bng#
bng# vip 192.168.1.100
bng# show conf
  //        configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  vip       192.168.1.100
  //        end of configuration
bng# no vip 192.168.1.100
bng# show conf
  //        configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  //        end of configuration
bng#
```

3.3.25 register

Synopsis: `register <networks[s]|target[s}|server[s]> <list>`

The specified list of networks, targets or servers (or just one of them) is taken from the "unregistered" state to the "registered state".

Target and Server: Definitions and States

A "Network" in BalanceNG associates the network parameters to a set of physical interfaces.

A "Server" in BalanceNG is a "virtual Server" and an addressable virtual "Host" that is capable to forward and load balance requests to the so called "real servers", the Targets.

A "Target" in BalanceNG is associated with a real existing address in the Target Network.

One Server is associated to one or more Targets and performs load balancing between them according to the specified load balancing methods.

Both Servers and Targets are associated automatically to a network according to the addressing.

Each Network, Server or Target is inside in BalanceNG in one of three states. See the state diagram below:

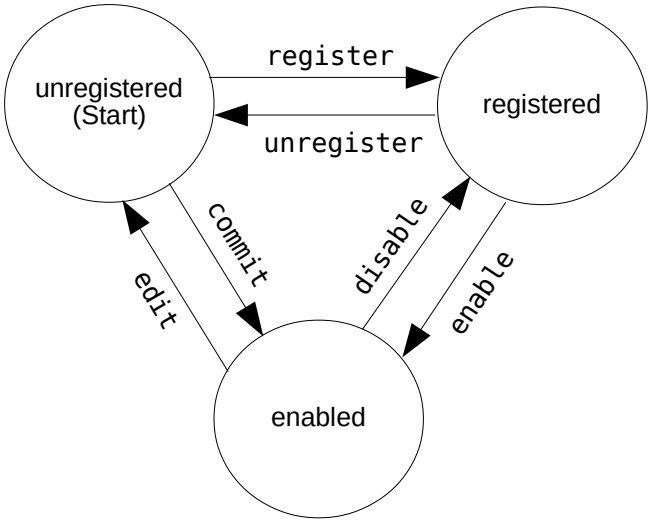


Figure 4: Server and Target States

The states are defined as follows:

"unregistered"-State

Each Target or Server is initially in that state. The Target or Server is unknown for BalanceNG and there's no associated functionality. The Entity (Target or Server) does not appear in the configuration. This state allows parameters and settings of the Target or Server to be edited and changed.

The command "`register`" registers the Target or Server and it's internal state changes to registered.

Using the abbreviation command "`commit`" the Target or Server may also be immediately taken to the "enabled"-State.

"registered"-State

In this state the entities (Target or Servers) are registered at the internal administrative data structures of BalanceNG. They appear in the configuration (e.g. at "`show conf`") but are not functional. The can not be edited or changed in that state. The further properties in this state are as follows:

Networks:

- Network is visible in "show networks"

Targets:

- Health checks are not being executed
- Targets are not being addressed by their associated Servers (they appear in square brackets at "`show servers`").
- ARP resolving is not being performed

Servers:

- ARP request are not answered
- ICMP echo requests are not answered ("pings")
- Requests to Services are ignored
- No VRRP sharing of Server address

The command "`enable`" puts the entity into "enabled" state, the command "`unregister`" puts it back into "unregistered" state.

"enabled"-state

In that state the entities are functional, that means the following:

Networks:

- "real" and "virt" addresses are pingable and usable by all other processing.
- VRRP is usable on the VRRP network

Targets:

- ARP resolving of the targets IP-address is performed
- Health checks are being executed according to the definitions
- Targets participate in load balancing is their health checks succeed

Servers:

- ARP requests are being answered

- ICP-ECHO requests ("Pings") are being answered
- Requests to Services are "load balanced" to the associated targets
- The session tables are active
- The Servers address is being shared using VRRP (if activated)

Examples:

```
BalanceNG: connected to PID 8158
bng# show conf
  //        configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  //        end of configuration
bng#
bng# network 1 addr 10.3.3.0
bng# network 1 mask 255.255.255.0
bng# network 1 real 10.3.3.1
bng# network 1 virt 10.3.3.2
bng# register network 1
bng# show conf
  //        configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  network   1 {
            addr 10.3.3.0
            mask 255.255.255.0
            real 10.3.3.1
            virt 10.3.3.2
            interface none
  }
  register  network 1
  //        end of configuration
bng#
bng# target 1 ipaddr 10.3.3.10
bng# register target 1
bng# show conf
  //        configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  network   1 {
            addr 10.3.3.0
            mask 255.255.255.0
            real 10.3.3.1
            virt 10.3.3.2
            interface none
  }
  register  network 1
  target    1 {
            ipaddr 10.3.3.10
  }
  register  target 1
  //        end of configuration
bng#
```

3.3.26 reload

Synopsis: `reload`

This command allows to reload the configuration file and potentially updated server / target relationships on the current VRRP master where not affected session-table entries are maintained.

The command reload fails, if there are any changes to the network and vrrp sections or to the “set” parameter section (and a “bng restart” is required immediately afterwards). In that case the error message “ERROR: was unable to reload, restart required.” is displayed and reported to the log.

Example:

```
# /etc/init.d/bng control
BalanceNG: connected to PID 8673
bng# reload
```

3.3.27 remark

Synopsis: `remark "<arbitrary remarks>"`

This command is available to allow adding custom remarks to the configuration file. This might be helpful e.g. for version tracking, configuration management and more.

The following example shows how the remark configuration command could be used to transport the Id of RCS (Revision Control System).

Example:

```
# /etc/init.d/bng restart
BalanceNG: not yet running
BalanceNG: starting up ...
# /etc/init.d/bng control
BalanceNG: connected to PID 8673
bng# show conf
  //        configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  //        end of configuration
bng# remark "$Id$"
bng# save
ok
bng# ... bye
# ci -l /etc/bng.conf
/etc/bng.conf,v  <--  /etc/bng.conf
enter description, terminated with single '.' or end of file:
NOTE: This is NOT the log message!
>> BalanceNG configuration Side A
>>
initial revision: 1.1
done
# /etc/init.d/bng restart
BalanceNG: shutdown of PID 8673 complete
BalanceNG: starting up ...
# /etc/init.d/bng control
BalanceNG: connected to PID 8683
bng# show conf
```

```
  //        configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  remark    "$Id: bng.conf,v 1.1 03/19/2006 11:17 root Exp root $"
  //        end of configuration
bng#
```

3.3.28 server <n>

Synopsis: `server <n> <subcommand> <values>`

This command is used to configure the parameters of a BalanceNG server. In general this is only possible if the Server is in the "unregistered" state (see the explanations at the "`register`" command).

The server index may range from 1 to 512, such allowing a total of 512 server sections per BalanceNG instance.

BalanceNG is capable of handling 512 independent Servers. Each server has one or multiple associated Targets where the load is being distributed to.

3.3.28.1 server <n> backup[s]

Synopsis: `server <number> backup[s] <list>|"none"`

This command assigns one or more backup targets to the Server with the specified number. Backup targets are being addressed if either there's no available addressable operational target available or if the "first choice" target fails and the Server's failover mode is "`backup`" (see "`server <n> failover`").

The list of backup targets contains of the backup target numbers separated by commas.

The command "`server <n> backup none`" deletes the list of backup targets completely. If there are multiple backup targets available BalanceNG always uses a Round Robin distribution among them.

The specification of a backup target together with "`failover backup`" allows the configuration of "N+1" high availability, where the overall load capacity of the target cluster remains the same even if one target fails.

Example:

```
BalanceNG: connected to PID 8169
bng# server 1 targets 1,2
bng# server 1 ipaddr 10.11.40.12
bng# server 1 backup 3
bng# commit server 1
WARNING: server 1 has no matching network
bng# show conf
  //        configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  server    1 {
            ipaddr 10.11.40.12
            targets 1,2
            backup 3
  }
  register  server 1
  enable    server 1
  //        end of configuration
bng#
```

3.3.28.2 server <n> failover

Synopsis: `server <n> failover "backup"|"normal"`

This command allows to switch a virtual server from normal failover mode to "backup" failover mode (and backwards). This has only an effect for the round-robin (default) method.

If this first target is not operational (either because disabled or a health check is failing) then BalanceNG selects the “next” alternate target among the group of defined targets (failover mode "normal").

If the failover mode for that server is "backup" then BalanceNG immediately selects a target from the set of backup targets.

Example:

```
bng# edit server 1
bng# server 1 failover backup
bng# commit server 1
WARNING: server 1 has no matching network
bng# show conf
  //        configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  server    1 {
            ipaddr 10.11.40.12
            failover backup
            targets 1,2
            backup 3
  }
  register  server 1
  enable    server 1
  //        end of configuration
bng#
```

3.3.28.3 server <n> ftimeout <value>|default

Synopsis: `server <n> ftimeout <value>|default`

This command sets a server specific TCP FIN/RST session timeout in seconds. A value of 0 or the string “default” disables this server specific TCP FIN/RST timeout.

As soon as a TCP FIN or RST packet is seen for such a session (in each direction), the session specific timeout will be degraded to that value. This session table entry event will be synchronized to the VRRP backup.

This command directive is implemented in the “slb” module for TCP IPv4 and TCP IPv6.

The garbage collection mechanism will then remove (or reclaim) that entry when this new timeout expires (if there's no more associated traffic).

The minimum value is 5 seconds, the maximum value is 172880 seconds (48 hours).

3.3.28.4 server <n> gslb dispatch

Synopsis: `server <n> gslb dispatch`

This server setting instructs BalanceNG to intercept DNS traffic which is handled by this server and instructs BalanceNG to directly return special A-record replies for a specific set of requested names.

The virtual server containing this instruction typically needs to do load balancing of DNS traffic on port 53 for both UDP and TCP for this kind of operation.

Only UDP traffic is handled by this setting. If an A record is requested for a name which is set as “name” of a different virtual server with “gslb enable” set, then the IPv4 address of the next available target is returned applying the usual load balancing and health check rules.

Example:

In this example “server 1” is used to load-balance DNS traffic to targets 1 and 2. If an A record is requested for “example.balanceng.net” (the “name” of server 2), then the IP address of either target 10 or 11 is returned instead with a TTL of only 10 seconds. The target is selected with the “session” method using the usual session management and health check rules.

```
server    1 {
          gslb dispatch
          ipaddr 172.17.2.70
          port 53
          ipdb enable
          method session
          targets 1,2
}
server    2 {
          name example.balanceng.net
          gslb enable
          gslbttl 10
          method session
          targets 10,11
}
register  server 1,2
enable    servers 1,2
```

3.3.28.5 server <n> gslb enable

Synopsis: `server <n> gslb enable`

This command enables GSLB (global server load balancing) operation for the server. If an DNS A-record request received on a virtual server with “gslb dispatch” set and if the name of this server matches the “name” of the server with “gslb enable” set, then BalanceNG returns the Ipv4 address of the target according to the usual load balancing and session handling rules.

Example:

```
server    1 {
          gslb dispatch
          ipaddr 172.17.2.70
          port 53
          ipdb enable
          method session
          targets 1,2
}
server    2 {
          name example.balanceng.net
          gslb enable
          gslbttl 10
          method session
          targets 10,11
}
```

```
  register  servers 1,2
  enable    servers 1,2
  ...
  target 10 {
            ipaddr 10.11.12.1
            ...
  }
  target 11 {
            ipaddr 10.11.12.2
            ...
  }
```

3.3.28.6 server <n> gslbttl

Synopsis: `server <n> gslbttl <seconds>`

This command specifies the DNS TTL which should be returned in the A-records generated with GSLB processing.

Example:

```
  ...
  server    2 {
            name example.balanceng.net
            gslb enable
            gslbttl 10
            method session
            targets 10,11
  }
  register  servers 1,2
  enable    servers 1,2
```

3.3.28.7 server <n> ipaddr

Synopsis: `server <n> ipaddr <ip4-address>`

`server <n> ipaddr none`

This command specifies the IPv4 address of the Server. As soon as the Server is in "enabled" State, BalanceNG responds to IPv4 ARP and ICMP ECHO (ping) requests and represents that address that way.

If VRRP is enabled then this IP address is being shared between all BalanceNG nodes of the VRRP vrid (virtual router id).

Example:

```
BalanceNG: connected to PID 8172
bng# server 1 ipaddr 10.11.40.12
bng# server 1 target 1
bng# register server 1
bng# enable server 1
WARNING: server 1 has no matching network
bng# show server 1
server 1
  ipaddr    10.11.40.12
  network   0
```

```
  port      any
  protocol  any
  status    enabled
  method    rr
  portrel   off
  target    [1]
bng#
```

If a special keyword "any" is supplied as the address, BalanceNG enters Link Load Balancing mode and performs routing using the targets as Layer 3 routing endpoints.

The following example shows a configuration which offers a virtual router with the private address 10.10.10.71 to a network and distributes the traffic evenly to two different outbound routers.

Example:

```
bng# show conf
  //        configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  license   INLAB01 d94e1a73b75478f4e1751b29253bf2de
  set       ipforwarding 1
  interface eth0
  interface eth1
  vrrp      {
            vrid 10
            priority 255
            network 2
            tracking enable
  }
  network   1 {
            name "Router Network"
            addr 172.17.2.0
            mask 255.255.255.0
            real 172.17.2.70
            virt 172.17.2.71
            interface eth0
  }
  network   2 {
            name "Internal Network"
            addr 10.10.10.0
            mask 255.255.255.0
            real 10.10.10.70
            virt 10.10.10.71
            interface eth1
  }
  register  networks 1,2
  enable    networks 1,2
  server    1 {
            name "BalanceNG virtual router"
            ipaddr any
            method session
            targets 1,2
  }
```

```
  register  server 1
  enable    server 1
  target    1 {
            name "Outbound Router 1"
            ipaddr 172.17.2.254
            ping 2,5
  }
  target    2 {
            name "Outbound Router 2"
            ipaddr 172.17.2.253
            ping 2,5
  }
  register  targets 1,2
  enable    targets 1,2
  //        end of configuration
```

3.3.28.8 server <n> ipaddr6

Synopsis: `server <n> ipaddr6 <ip6-address>`

`server <n> ipaddr6 none`

This command specifies the IPv6 address of the Server.

Example:

```
bng# server 1 ipaddr6 fe80::20e:cff:fe6c:1
bng# commit server 1
```

3.3.28.9 server <n> ipdb

Synopsis: `server <n> ipdb enable`

`server <n> ipdb disable`

This command enables IPDB processing for the specified server. All source Ipv4 addresses are being looked up in the database updating the location counters. Targets that are associated with a location group (see “target <n> lgrp”) will receive traffic if the location of the client source IP address is part of the target's location group.

The whole IPDB location based server load balancing is working with the load balancing methods “agent”, “bw”, “bwin”, “bwout”, “random”, “rndagent” and “session” (but not with “hash”, “rr” and server plugins).

It's valid to specify “server <n> ipdb enable” only, which allows to keep track of just the client locations (as shown by the command “show locations”).

Example:

```
bng# show conf
...
  server    1 {
            ipaddr 172.17.2.70
            port 53
            ipdb enable
            method session
            targets 1,2
  }
...
```

```
  target    1 {
            ipaddr 172.17.2.91
            port 53
            lgrp A
            ping 5,12
            tcpopen 53,5,12
            dsr enable
  }
...
bng#
```

3.3.28.10 server <n> method

Synopsis: `server <n> method "rr"|"hash"|"random"|"agent"|...`

This command specifies which load balancing method should be active for the specified Server. The load balancing method determines which target to choose for new sessions and for sessions, where the associated target has become nonoperational (down).

The following methods are available:

3.3.28.10.1 rr

This is the default "Round Robin" distribution method. Targets are chosen cyclically. A simple weighting can be implemented by adding the same target twice or multiple times to the servers target list. If the method "rr" is active there's no output in the configuration file since this is the default method.

3.3.28.10.2 hash

Using a hash function with the client source IP address as the key each possible client source address is being associated with the same target in a target set. This method may be used to achieve a persistence which noes not depend on the current session table state (as long as all target servers are operational and enabled).

The hash function consists of XOR'ing the four octets of the source IP address modulo the total number of associated (enabled) targets of the server.

The behavior of this method may be modified using the hashbytes4 and hashbytes6 parameters.

3.3.28.10.3 random

One target out of the set is chosen randomly. A weighting is possible using the “target <n> weight” keyword (see there). The default target weight is 1.

3.3.28.10.4 agent

All targets of a Servers target set have to be specified with "agent" as one of the health checks. This method chooses the target with the lowest agent return value (starting with 1). This allows "least resource" load balancing using the bngagent program on the target.

Please take also a look at method “rndagent” which avoids overloading a target in some situations.

Example:

```
BalanceNG: connected to PID 8175
bng# server 1 ipaddr 10.2.2.1
bng# server 1 method agent
```

```
bng# server 1 targets 1,1,2
bng# commit server 1
WARNING: server 1 has no matching network
bng# show servers
    # ipaddr           port prt net S targets {backups}
  ------------------------------------------------------------
    1 10.2.2.1          any any   0 e [1],[1],[2] {}
bng# show server 1
server 1
  ipaddr    10.2.2.1
  network   0
  port      any
  protocol  any
  status    enabled
  method    agent
  portrel   off
  targets   [1],[1],[2]
bng#
```

3.3.28.10.5 bw

This method chooses the target which consumes the least current total bandwidth among all targets of the server. The current total bandwidth value may be modified virtually by the “offset” and “scale” parameters of the target allowing arbitrary weighting and preference settings.

Example:

```
bng# edit server 1
bng# server 1 method bw
bng# commit server 1
bng# show server 1
server 1
  ipaddr    172.17.2.189
  network   1
  port      22
  protocol  tcp
  status    enabled
  method    bw
  portrel   off
  target    1
bng#
```

3.3.28.10.6 bwin

This method operates like method “bw”, but only the incoming bandwidth is taken into account.

3.3.28.10.7 bwout

This method operates like method “bw”, but only the outgoing bandwidth is taken into account.

3.3.28.10.8 first

This method selects the first operational target in the server target list (examined from left to

right). If there's no operational target the selection process continues with the backup target list (if there is one).

The the IP databases are loaded and the location groups are activated, this method selects the first operational target which matches the location group of the source IPv4 or IPv6 address of the client.

3.3.28.10.9 rndagent

This method takes the scores collected by the BalanceNG agent (bngagent) into account very similar to the “agent” method. The difference is, that internally a weight is calculated per target and that the next session is chosen by a weighted random algorithm.

The weight for rndagent is calculated by the following formula:

target_weightN = 100 * (1 – agent_scoreN / (agent_score1 + agent_score2 + ...));

The agent_scores above are calculated by the following formula:

agent_scoreN = original_agent_scoreN * target_scaleN + target_offsetN;

Example:

```
bng# edit server 1
bng# server 1 method rndagent
bng# commit server 1
bng# show server 1
server 1
  ipaddr    172.17.2.189
  network   1
  port      22
  protocol  tcp
  status    enabled
  method    rndagent
  portrel   off
  target    1
bng#
```

3.3.28.10.10 session

This method chooses the target with the least number of current sessions as target for the current new session.

Example:

```
bng# edit server 1
bng# server 1 method session
bng# commit server 1
bng# show server 1
server 1
  ipaddr    172.17.2.189
  network   1
  port      22
  protocol  tcp
  status    enabled
  method    session
  portrel   off
  target    1
bng#
```

The session distribution method may be modified by the target “offset” and “scale” parameters allowing weighting by applying a linear function to the current number of sessions.

3.3.28.11 server <n> name

Synopsis: `server <n> name <name>|"none"`

Assigns a server a name for informational purposes. The string "none" as the name arguments deletes the name from the specified server.

The server name may be embedded in double quotes to specify a name containing spaces. Specifying an empty string in double quotes also removes the current name definition.

For GSLB (Global Server Load Balancing) this parameter contains the name for which an A record should be returned by BalanceNG (GSLB needs to be enable with “gslb enable” for that server).

Example:

```
bng# edit server 1
bng# server 1 name test44
bng# commit server 1
WARNING: server 1 has no matching network
bng# show conf
  //        configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  server    1 {
            name test44
            ipaddr 10.2.2.1
            method hash
            targets 1,1,2
  }
  register  server 1
  enable    server 1
  //        end of configuration
bng# edit server 1
bng# server 1 name none
bng# commit server 1
WARNING: server 1 has no matching network
bng# show conf
  //        configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  server    1 {
            ipaddr 10.2.2.1
            method hash
            targets 1,1,2
  }
  register  server 1
  enable    server 1
  //        end of configuration
bng#
```

3.3.28.12 server <n> plugin

Synopsis: `server <n> plugin <script>|"none"|""`

This command connects the session creation mechanism of the virtual server <n> to a plugin program or script. The plugin script path has to be specified as a full path name.

Specifying “none” or two double quotes (“”) sets the plugin parameter of the virtual server to “nothing”, which is equivalent of disabling that feature.

A plugin script or program is connected through stdin and stdout and receives session information as a line in readable ASCII format containing the following five parameters separated by a single blank:

1.	Protocol	“UDP” or “TCP”
2.	Source IP Address	Ipv4 address in numerical dot notation
3.	Source Port	decimal number
4.	Destination IP Address	Ipv4 address (usually that of the virtual server)
5.	Destination Port	decimal number

The result of the plugin is delivered back to BalanceNG by stdout by printing one line consisting of a number in readable ASCII. The semantics are as follows:

Return Value	Meaning
-1	This session request is denied, drop the packet
0	This session request is OK, proceed as usual
>= 1	The session should be directed to the target with this specific number (target has to be among the target set of the virtual server).

WARNING: Server plugins should be as fast as possible since the BalanceNG core switching engine actually waits until the plugin returned it's result value.

BalanceNG assumes an ultra fast, 100% working component at this very critical interface. Plugins are started once at "server enable" and killed at "server disable".

Example:

Here's a simple example of a server plugin written in perl. This example implements a filter which allows session only from source IP addresses originating from 10.10.10.0/24. IMPORTANT: Setting the autoflush (IO::Handle) functionality is required for perl scripts to allow a line-by-line communication with BalanceNG (BalanceNG could stall otherwise).

```
#!/usr/bin/perl
use IO::Handle;
autoflush STDOUT 1;
while ($request = <STDIN>) {
  chomp $request;
  ($proto, $saddr, $sport, $daddr, $dport) = split /\s+/, $request;
  if($saddr=~ m/10\.10\.10\.\d+/) {
    printf("0\n");
  } else {
    printf("-1\n");
  }
}
```

This script is connected using the “plugin” keyword inside a server block:

```
server    1 {
          ipaddr 172.17.2.64
          port 80
          protocol tcp
          plugin /home/bng/filter.pl
          method session
          targets 1,2
}
```

3.3.28.13 server <n> port

Synopsis: `server <n> port <portspec>|"any"`

This command restricts the services being offered by the BalanceNG Server to a specific port. Per default any port (and any protocol) is being load balanced to the associated targets. If a specific port is specified, then a new session is only created if the destination port towards the Server matches this port.

The specification of a port may be removed / reverted to the default by applying the keyword "any".

If the port is specified using this command, but the protocol is "any" then new sessions will be created for both UDP and TCP packets.

A server with a specified port and an associated target with a different port means that a port translation takes place.

Example (round robin load balancing port 8080 to two targets port 80):

```
# /etc/init.d/bng control
BalanceNG: connected to PID 8178
bng# interface eth0
interface eth0 successfully attached
bng# interface eth1
interface eth1 successfully attached
bng# network 1 {
bng+ addr 10.2.2.0
bng+ mask 255.255.255.0
bng+ real 10.2.2.100
bng+ virt 10.2.2.101
bng+ interface eth0
bng+ }
bng# network 2 {
bng+ addr 192.168.1.0
bng+ mask 255.255.255.0
bng+ real 192.168.1.100
bng+ virt 192.168.1.101
bng+ interface eth1
bng+ }
bng# commit networks 1,2
bng# target 1 ipaddr 10.2.2.1
bng# target 1 port 80
bng# target 1 protocol tcp
bng# target 1 tcpopen 80,10,30
bng# commit target 1
```

```
WARNING: target 1 in enabled state but not referenced
bng# target 2 ipaddr 10.2.2.2
bng# target 2 port 80
bng# target 2 protocol tcp
bng# target 2 tcpopen 80,10,30
bng# commit target 2
WARNING: target 1 in enabled state but not referenced
WARNING: target 2 in enabled state but not referenced
bng# server 1 ipaddr 192.168.1.1
bng# server 1 port 8080
bng# server 1 protocol tcp
bng# server 1 targets 1,2
bng# commit server 1
bng# show conf
  //        configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  interface eth0
  interface eth1
  network   1 {
            addr 10.2.2.0
            mask 255.255.255.0
            real 10.2.2.100
            virt 10.2.2.101
            interface eth0
  }
  network   2 {
            addr 192.168.1.0
            mask 255.255.255.0
            real 192.168.1.100
            virt 192.168.1.101
            interface eth1
  }
  register  networks 1,2
  enable    networks 1,2
  server    1 {
            ipaddr 192.168.1.1
            port 8080
            protocol tcp
            targets 1,2
  }
  register  server 1
  enable    server 1
  target    1 {
            ipaddr 10.2.2.1
            port 80
            protocol tcp
            tcpopen 80,10,30
  }
  target    2 {
            ipaddr 10.2.2.2
            port 80
            protocol tcp
```

```
            tcpopen 80,10,30
  }
  register  targets 1,2
  enable    targets 1,2
  //        end of configuration
bng#
```

3.3.28.14 server <n> ports <p1>,<p2>

Synopsis: `server <n> ports <p1>,<p2>`

This command specifies two related ports for a server. BalanceNG directs connections to each of the ports always to the same target and manages sessions accordingly. This is useful to combine e.g. port 80 and 443 in order to maintain the same target during a switchover from HTTP to SSL-HTTP.

Port relevance of the server has to be switched off and the associated targets must have "port any" specified.

Example:

```
bng# edit server 1
bng# server 1 ports 80,443
bng# commit server 1
bng# show server 1
server 1
  ipaddr    172.17.2.189
  network   1
  ports     80,443
  protocol  any
  status    enabled
  method    rr
  portrel   off
  target    1
bng#
```

3.3.28.15 server <n> protocol

Synopsis: `server <n> protocol any|tcp|udp|sctp`

This command either restricts the server load balanced connections to tcp, udp, sctp or to any of these protocols.

SCTP support is experimental and available in DSR mode only.

This command may be used with "`server <n> port`" in any combination.

Example:

```
bng# edit server 1
bng# server 1 ipaddr 10.2.2.4
bng# server 1 protocol any
bng# commit server 1
bng# show server 1
server 1
  ipaddr    10.2.2.4
  network   1
  port      8080
```

```
  protocol  any
  status    enabled
  method    rr
  portrel   on
  targets   (1),(2)
bng#
```

3.3.28.16 server <n> proxy enable

Synopsis: `server <n> proxy enable`

This command enables proxy mode for the specified server and is implemented in the “slb” module. If enabled, connections to the virtual server are forwarded to the selected target by replacing the original client IP address and port by the IP address of the virtual server and a new source port. Per default, “server <n> proxy” is disabled. This mechanism works for IPv4 and IPv6 addresses, state information is replicated from the VRRP server node to the VRRP backup node.

If an associated target has DSR enabled at the same time, the proxy functionality is chosen only for clients connecting over local LAN (as known by BalanceNG).

Example:

```
//        configuration taken Tue Nov 16 15:23:55 2010
//        BalanceNG 3.449 (created ...)
license   INLAB-TEST-01 584e6beefec16c55cbd5ce8fa8d1a74a
modules   vrrp,arp,ping,hc,master,slb
interface 1 {
          name eth0
}
register  interface 1
enable    interface 1
vrrp      {
          vrid 33
          priority 200
          network 1
}
network   1 {
          addr 172.17.2.0
          mask 255.255.255.0
          real 172.17.2.55
          virt 172.17.2.56
          interface 1
}
register  network 1
enable    network 1
server    1 {
          ipaddr 172.17.2.58
          port 80
          protocol tcp
          proxy enable
          targets 1,2
}
register  server 1
enable    server 1
```

```
target    1 {
          ipaddr 172.17.2.30
          port 80
          protocol tcp
          tcpopen 80,3,10
}
target    2 {
          ipaddr 172.17.2.31
          port 80
          protocol tcp
          tcpopen 80,3,10
}
register  targets 1,2
enable    targets 1,2
//        end of configuration
```

3.3.28.17 server <n> prx <IPv4 address>

Synopsis: `server <n> prx <IPv4 address>`

This directive controls the IPv4 address which is being used as the proxy address in conjunction with ***server <n> proxy enable*** and IPv4 protocols. It needs to be an address which is represented by BalanceNG with respect to ARP resolution, so the ***network <n> virt*** address is suitable for this purpose (and recommended).

If this directive is not present in the server section, the address of the virtual server itself is being used.

Note: This extension is available with BalanceNG release 4.110 (and higher).

3.3.28.18 server <n> prx6 <IPv6 address>

Synopsis: `server <n> prx <IPv6 address>`

This directive controls the IPv6 address which is being used as the proxy address in conjunction with ***server <n> proxy enable*** and IPv6 protocols. It needs to be an address which is represented by BalanceNG with respect to ND6 resolution, so the ***network <n> virt6*** address is suitable for this purpose (and recommended).

If this directive is not present in the server section, the address of the virtual server itself is being used.

Note: This extension is available with BalanceNG release 4.110 (and higher).

3.3.28.19 server <n> sessionid <handler>

This command associates a specific session handler to a particular virtual server. The “slb” (Server Load Balancing) Module needs to be part of the current module chain.

3.3.28.19.1 sip

The sessionid is based on SIP/UDP Call-ID only.

3.3.28.19.2 src

The sessionid is based only on the source IP address.

3.3.28.19.3 src+dstport

The sessionid is based on the source IP address and the destination port.

3.3.28.19.4 src+port

The sessionid is based on the source IP address and the source port.

3.3.28.19.5 src+ports

The sessionid is based on the source IP address and both the source and destination ports.

3.3.28.19.6 src+tag

The sessionid is based on the source IP address and the sessiontag (see “server <n> sessiontag” and “target <n> sessiontag”).

3.3.28.19.7 dst

The sessionid is based only on the destination IP address.

3.3.28.19.8 dst+port

The sessionid is based on the destination IP address and the destination port.

3.3.28.19.9 dst+ports

The sessionid is based on the destination IP address and both the source and destination ports.

3.3.28.19.10 dst+srcport

The sessionid is based on the source IP address and the source port.

3.3.28.19.11 dst+tag

The sessionid is based on the destination IP address and the sessiontag (see “server <n> sessiontag” and “target <n> sessiontag”).

3.3.28.20 server <n> snat enable|disable

Synopsis: `server <n> snat enable|disable`

This command enables or disables SNAT processing in the SLB module for a specific virtual server, respectively. See also “snatrange” and “show snat” as related commands.

3.3.28.21 server <n> sessiontag <tag>

Synopsis: `server <n> sessiontag <tag>|default`

This command modifies a server specific session tag. The tag may be used by the sessionid “src+tag” thus allowing to group several server ports to be handled by the same session table entry. A tag is either 0 (default) or any positive integer number.

3.3.28.22 server <n> stimeout <value>|null|default

Synopsis: `server <n> stimeout <value>|default`

This command sets a server specific session timeout in seconds. A value of 0 or the string “default” disables the server specific sessions timeout, in that case the global session timeout is valid for all sessions (see the parameter “sessiontimeout”).

The special value “null” disables session generation completely, which is useful to implement true round robin load-balancing for UDP protocols (like SIP).

The minimum value is 10 seconds, the maximum value is 172880 seconds (48 hours).

Example:

```
bng# show server 4
server 4
  ipaddr    172.17.2.82
  network   1
  port      22
  protocol  tcp
  status    enabled
  method    rr
  portrel   off
  target    1
bng# edit server 4
bng# server 4 stimeout 120
bng# commit server 4
bng#
```

3.3.28.23 server <n> target[s]

Synopsis: `server <n> target[s] <list>|none`

This command associates one or more targets to the specified server. The "virtual" server then distributes the requests to the operational targets in that set according to the specified load balancing method.

One or more target can be specified. The same target may appear multiple times to allow a simple weighting of the distribution.

Either the singular or plural of "`target`" may be used.

Specifying "none" as list parameter completely empties the list of associated targets.

Example:

```
bng# edit server 1
bng# server 1 targets 1,2,3,4,5,6,7,8
bng# commit server 1
bng# show server 1
server 1
  ipaddr    10.2.2.4
  network   1
  port      8080
  protocol  any
  status    enabled
  method    rr
  portrel   on
  targets   (1),(2),[3],[4],[5],[6],[7],[8]
bng# show servers
    # ipaddr           port prt net S targets {backups}
  ------------------------------------------------------------
    1 10.2.2.4         8080 any   1 e (1),(2),[3],[4],[5],[6],[7],[8] {}
bng#
```

3.3.28.24 server <n> tcprefuse

Synopsis: `server <n> tcprefuse enable|disable`

If the tcprefuse flag is enabled, a TCP connection attempt to the virtual server will be immediately refused by sending a RST/ACK packet back to the originating client.

This feature works for both IPv4 and IPv6 and the target list of the virtual server has no further effect (and may be left empty).

The virtual server may have either a specific port or “port any” defined, thus refusing connections to a specific port or to any possible TCP port, respectively.

The server protocol should be set to TCP (using “server <n> protocol tcp”).

The tcprefuse settings can be changed even if the virtual server is in registered or enabled state.

Example:

```
bng# server tcprefuse enable
```

3.3.28.25 server <n> tcpreset

Synopsis: `server <n> tcpreset enable|disable`

If the tcpreset flag is enabled, an already existing TCP connection to the virtual server will be immediately closed by sending a RST packet back to the originating client. This happens in the moment as a valid packet for this virtual server connection is received in the “slb” module.

This feature works for both IPv4 and IPv6.

The virtual server may have either a specific port or “port any” defined, thus resetting connections to a specific port or to any possible TCP port, respectively.

The server protocol should be set to TCP (using “server <n> protocol tcp”).

The tcpreset settings can be changed even if the virtual server is in registered or enabled state.

Example:

```
bng# server tcprefuse enable
```

3.3.28.26 server <n> udpdup

Synopsis: `server <n> udpdup s1,s2,...`

`server <n> udpdup none`

This configuration directive allows to configure additional virtual servers for UDP IPv4/IPv6 packet duplication.

When a virtual server with an udpdup list has sent out an UDP packet, an additional UDP packet is sent to each server in the udpdup list (selecting a an individual target and generating a session table entry for each).

In order to insert the virtual servers into the server registry, the servers addressed by an udpdup list do not specify an IPv4 or IPv6 address but need to have an unique port number configured (which is not used or addressable).

The load balancing methods and server specific session timeouts are configured as usual and may be distinct.

As usual, the target configuration controls whether DSR is used and initiates port and address

rewriting if necessary.

Server section configuration example:

```
server 1 {
  ipaddr 10.1.1.1
  port 514
  targets 1,2,3
  udpdup 2,3,4
}
server 2 {
  port 2
  targets 4,5,6
}
server 3 {
  port 3
  targets 7,8,9
}
server 4 {
  port 4
  targets 10,11,12
}
...
```

3.3.29 set

Synopsis: `set <parameter> <value>|"default"`

BalanceNG uses a set of internal parameters. Using set these parameters may be changed. All parameters are numeric with a minimum, a maximum and a default. If the parameter is currently set to the default value then no "`set`" line appears in the configuration file (see "`show parameters`").

A special set block may be opened by specifying "{" as the first argument to `set`.

The command "`set <parameter> default`" sets the specified parameter back to its default value.

3.3.29.1 set arplookup

Synopsis: `set arplookup <value>|default`

This parameter controls how often (interval in seconds) an still unknown mac address is being requested using the ARP protocol. The minimum of this parameter is 5, the maximum 60 and the default value is 10 seconds.

Example:

```
BalanceNG: connected to PID 8182
bng# set arplookup 40
bng# show conf
  //        configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  set       arplookup 40
  //        end of configuration
bng# show parameters
            name        min        max    default    current
  ------------- ---------- ---------- ---------- ----------
```

arplookup	5	60	10	10
arprefresh	60	3600	300	300
arptimeout	0	86400	0	0
backupalerts	0	1	1	1
bngsyncport	1024	49151	10439	10439
bmduration	10	86400	300	300
bmpsize	20	1514	1514	1514
bmwsize	1	10000	128	128
bngfilter	0	1	1	1
debugscope	0	9	0	0
dumprotation	1	1048576	1024	1024
gnatdlimit	10	100000	10	10
gratarpremind	0	120	0	0
hashbytes4	1	4	4	4
hashbytes6	1	16	16	16
hcportoffset	1024	65535	30000	30000
ipforwarding	0	1	0	0
localdsr	0	1	0	1
localvirt	0	1	0	0
maxsyncps	0	10000	0	0
multithreading	1	1	1	1
natdlimit	10	500	10	10
natscan	1	20	10	10
natsync	0	1	1	1
natsynciv	10	120	10	10
nattimeout	10	172800	600	600
noftupdate	0	1	0	0
outmtu	0	1514	0	0
psvrelearn	0	1	0	0
pthreadstacksize	0	1000000	204800	204800
sendprobes	0	1	0	0
sessionautoresync	0	1	1	1
sessionarrtimeout	0	3600	60	60
sessiongclimit	1000	500000	100000	100000
sessiondlimit	10	1000	10	10
sessionscan	1	20	10	10
sessionscanbup	1	1000	100	100
sessionsync	0	1	1	1
sessionsyncack	0	1	1	1
sessionsyncetype	0	1	0	0
sessionsynciv	10	120	10	10
sessiontimeout	10	172800	600	600
snattimeout	10	172800	1800	1800
sweeponreload	0	1	1	1
syncackbdelay	1	60	10	10
syncackmaxps	1	10000	2000	2000

```
  syncackresend          1      60       5       5
  syncackwsize           1   10000     100     100
  stickytarget           0       1       1       1
  strictrouting          0       1       1       1
  tarpitrealto          60   86400   14400   14400
  tarpittrapto          60   86400     600     600
  vrrpmasterdown         3      10       3       3
  vrrppreempt            0       1       0       0
  vrrppreemptts          0     100       0       0
  vrrpstateplugin        0       1       0       0
  vrrpv3ip               4       6       5       5
  vrrpversion            2       3       3       3
  xstlog                 0       1       0       1
bng# set arprefresh 180
bng# show conf
  //        configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  set       {
            arplookup 40
            arprefresh 180
  }
  //        end of configuration
bng#
```

3.3.29.2 set arprefresh

Synopsis: `set arprefresh <value>|default`

This parameter controls how often an already known mac address is being reexamined using the ARP-Protocol (interval in seconds). The minimum of this parameter is 60, the maximum 300 and the default value is 120 seconds.

This parameter determines how e.g. fast a changed mac address of a target is being recognized by BalanceNG.

Example:

```
bng# set arprefresh 240
bng# show conf
  //        configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  set       {
            arplookup 40
            arprefresh 240
  }
  //        end of configuration
bng#
```

3.3.29.3 set arptimeout

Synopsis: `set arptimeout <value>|default`

All targets in "enabled" state have an associated IP address, for which the mac / Ethernet address has to be determined using the ARP protocol. This action can be regarded as a basic health check, that has to succeed as a prerequisite for all other health checks.

The parameter "arptimeout" controls after how many seconds of a missing ARP reply a target has to become not operational.

The special value of 0 disables this particular implicit healthcheck.

The minimum of this parameter is 0, the maximum 86400 and the default value is 0 seconds (disabled).

```
bng# set arptimeout 60
bng# show conf
  //        configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  set       {
            arplookup 40
            arprefresh 240
            arptimeout 60
  }
  //        end of configuration
bng#
```

3.3.29.4 set backupalerts

Synopsis: `set backupalerts <value>|default`

If this parameter is set to 1 (default) alert and upalert scripts are also execute on a backup VRRP node. If this parameter is set to 0 the execution of these alert/upalert scripts is suppressed on the VRRP backup node.

3.3.29.5 set bngsyncport

Synopsis: `set bngsyncport <value>|default`

This parameter defines the UDP port being used by the bngsync “BalanceNG session table synchronization” protocol. It defaults to the IANA registered port 10439.

It may be set to a different value in the range 1024 to 49151 (including) and should be the same on all participating nodes.

3.3.29.6 set bmduration

Synopsis: `set bmduration <value>|default`

This sets the benchmark duration in seconds (see benchmark command).

3.3.29.7 set bmpsize

Synopsis: `set bmpsize <value>|default`

This sets the packet size in bytes being used for benchmarking (see benchmark command).

3.3.29.8 set bmwsize

Synopsis: `set bmwsize <value>|default`

This parameter controls the number of packets being sent out at the beginning of the benchmark (“window size”).

3.3.29.9 set debugscope

Synopsis: `set debugscope <value>|default`

This parameter sets the scope for additional built in debug messages. The default is 0 (no debug messages). The command “show debugscopes” displays the currently available debugging scopes.

3.3.29.10 set dumprotation

Synopsis: `set dumprotation <value>|default`

This parameter controls the dumpfile rotation of the files generated by the "`dump`" command. The parameter dumprotation specifies the maximum size threshold of one dumpfile in megabytes.

Specifying "`default`" as parameter restores the current value to the default value.

The minimum of this parameter is 1 (one Megabyte), the maximum 1048576 (one Terabyte) and the default value is 1024 (one Gigabyte).

3.3.29.11 set eventinterval

Synopsis: `set eventinterval <value>|default`

This parameter controls the interval for event rate computation (see “event” command). The default and minimum of this parameter is 10 (seconds), the upper maximum is 600 seconds (5 minutes).

3.3.29.12 set gnatdlimit

Synopsis: `set gnatdlimit <value>|default`

This parameter controls the maximum number of GNAT (Generic NAT) entries at the “show nat” command. The Generic NAT Table is used by the “nat” module and the server proxy mode. The default of this parameter is 10, the maximum may be set up to 100000 for testing purposes.

3.3.29.13 set gratarpremind

Synopsis: `set gratarpremind <value>|default`

This parameters specifies an interval in minutes at which additional "reminding" gratuitous ARP requests are being sent out by BalanceNG. A value of "0" disables this feature (default).

If VRRP is active only the current VRRP master will send out additional gratuitous arp requests.

Example:

```
# /etc/init.d/bng control
BalanceNG: connected to PID 8194
bng# set gratarpremind 30
bng# show conf
  //        configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  set       gratarpremind 30
  //        end of configuration
bng#
```

3.3.29.14 set hashbytes4

Synopsis: `set hashbytes4 <value>|default`

This parameter controls the number of bytes which are considered by the “hash” method

(server <n> method hash) of the IPv4 source address. The default value is 4 (all 4 octets).

3.3.29.15 set hashbytes6

Synopsis: `set hashbytes6 <value>|default`

This parameter controls the number of bytes which are considered by the “hash” method (server <n> method hash) of the IPv6 source address. The default value is 16 (all 16 octets).

3.3.29.16 set hcportoffset

Synopsis: `set hcportoffset <value>|default`

This parameter controls the offset of the source port being used for `tcpopen` and `agent` health checks. To calculate the source port the target index is simply added to this offset.

The minimum of this parameter is 1024 (the first usually non privileged port), the maximum 65535 (maximum of unsigned short) and the default value is 30000.

Example:

```
# /etc/init.d/bng control
BalanceNG: connected to PID 8194
bng# set hcportoffset 1024
bng# show conf
  //        configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  set       hcportoffset 1024
  //        end of configuration
bng#
```

3.3.29.17 set ipforwarding

Synopsis: `set ipforwarding <value>|default`

This parameter enables IP forwarding if set to 1 and disables IP forwarding if set to 0 (which is the default). If IP forwarding is enabled, BalanceNG routes IP packets between all configured networks. Packets which are not locally addressable are being forwarded to the default gateway (see "gateway") if specified.

Network virtual ("virt") addresses should be preferably used as routing destinations since those addresses are shared between all nodes of the virtual VRRP router.

Example (on a Sun X2100):

```
bng# show conf
  //        configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  set       ipforwarding 1
  interface bge0
  interface nge0
  network   1 {
            addr 10.100.1.0
            mask 255.255.255.0
            real 10.100.1.71
            virt 10.100.1.70
            interface bge0
  }
  network   2 {
```

```
            addr 10.100.2.0
            mask 255.255.255.0
            real 10.100.2.71
            virt 10.100.2.70
            interface nge0
  }
  register  networks 1,2
  enable    networks 1,2
  //        end of configuration
bng#
```

3.3.29.18 set localdsr

Synopsis: `set localdsr 0|1`

This boolean parameter allows if set to 1 operating BalanceNG on a target machine itself in DSR mode. Usually BalanceNG refuses to represent an IP address which is already present on one of the local interfaces. If localdsr is set to one this check is exempted for virtual server IP addresses which then may be present as a local loopback alias at the same time on the same machine. The localdsr parameter is set to 0 (off) by default.

Example:

```
bng# set localdsr 1
bng# show conf
  //        configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  remark    "$Id: bng.conf,v 1.1 2022/11/08 13:59:31 root Exp
  set       localdsr 1
  //        end of configuration
bng#
```

3.3.29.19 set localvirt

Synopsis: `set localvirt 0|1`

“set localvirt 1” allows to have “network virt” addresses active on the host operating system at the same time together with BalanceNG. This is required for example if BalanceNG operates as a VRRP daemon attracting traffic to a local interface. This localvirt parameter is set to 0 (off) by default.

Example:

```
bng# set localvirt 1
bng# show conf
  //        configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  remark    "$Id: bng.conf,v 1.1 2022/11/08 13:59:31 root Exp
  set       localvirt 1
  //        end of configuration
bng#
```

3.3.29.20 set multithreading

Synopsis: `set multithreading <value>|default`

This boolean parameter activates multithreading mode if set to 1 (active). The default value is 0 (not activated). If this parameter is changed, BalanceNG needs to be restarted (e.g. by “bng

restart") to activate the change. A BalanceNG reload alone (e.g. by "bng reload") has no further effect (the parameter is changed, but the currently running scheduler remains the same).

When the multithreading scheduler is running, dumping packets with the "dump" command is not functional due to efficiency reasons. Adding an interface needs a "save" and "bng restart" to take effect.

3.3.29.21 set natdlimit

Synopsis: `set natdlimit <value>|default`

This parameter controls the maximum number of NAT entries displayed per protocol (TCP, UDP) at the "show nat" command. The default of this parameter is 10.

3.3.29.22 set natscan

Synopsis: `set natscan <value>|default`

This parameter specifies the number of NAT entries being checked and reclaimed per second on a not busy BalanceNG system. The default of this parameter is 10.

3.3.29.23 set natsync

Synopsis: `set natsync 1|0`

This boolean parameter enables syncing of NAT table entries from the master to the backup BalanceNG node if set to 1 (which is the default).

3.3.29.24 set natsynciv

Synopsis: `set natsynciv <value>|default`

This parameter controls the interval at which active NAT entries are being re-synced / refreshed between the master and the backup BalanceNG node (is natsync is enabled and set to 1). The default value of this parameter is 10 seconds.

3.3.29.25 set nattimeout

Synopsis: `set nattimeout <value>|default`

This parameter controls the lifetime of an unused NAT entry in the BalanceNG NAT table in seconds. The default of this parameter is 600 seconds (10 minutes).

3.3.29.26 set noftupdate

Synopsis: `set noftupdate <value>|default`

This boolean parameter allows to disable the update of the BalanceNG internal Layer-2 forwarding table completely (if set to "1", enabled). The default is "0" (disabled). To display the BalanceNG forwarding table see the "show machash" command.

This is an experimental parameter and functionality and should only be enabled on request by the BalanceNG software support.

3.3.29.27 set outmtu

Synopsis: `set outmtu <value>|default`

This parameter limits the maximum packet size to the specified number of bytes. Packets which exceed the specified number of bytes are simply truncated to the maximum cutting off the exceeding trailer. If set to 0 this mechanism is disabled. This parameter may be set to

1514 bytes (the maximum) if an unwanted FCS (Ethernet frame checksum) is passed from the OS to BalanceNG on the reading side.

3.3.29.28 set pthreadstacksize

Synopsis: `set pthreadstacksize <value>|default`

This parameter controls the stack size of any POSIX thread. If this parameter is changed during runtime, a “save” and “bng restart” is needed to take effect. The default is 204800 bytes.

3.3.29.29 set psvrelearn

Synopsis: `set psvrelearn <value>|default`

This parameter enables passive updates to the ARP table if set to 1. The default value is 0 (disabled).

3.3.29.30 set maxsyncps

Synopsis: `set maxsyncps <value>|default`

This parameter controls the maximum number of new session table entries per second being sent from the current VRRP master to the VRRP backup node. The default value is 0, which is interpreted as an unlimited number of synchronization packets. This parameter may be safely set to any value in order to prevent unwanted synchronization traffic in case of a DoS (Denial of Service) or DDoS (Distributed Denial of Service) attack.

Since this parameter is only valid for freshly created session table entries, the usual session synchronization takes place afterwards ensuring a proper state of the VRRP backup session table.

Example:

```
bng# set maxsyncps 1000
bng# show conf
  //        configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  remark    "$Id: bng.conf,v 1.1 2022/11/08 13:59:31 root Exp
  set       maxsyncps 1000
  //        end of configuration
bng#
```

3.3.29.31 set sendprobes

Synopsis: `set sendprobes <value>|default`

This boolean parameter controls whether ARP request probes are sent out periodically to probe for potential IP address conflict. This parameter is off (0) by default.

Example:

```
bng# set sendprobes 1
bng# show conf
  //         configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  set       {
            hcportoffset 1024
            sessionscan 2
            vrrppreempt 1
```

```
            vrrpmasterdown 10
            sendprobes 1
  }
  //        end of configuration
bng#
```

3.3.29.32 set sessionautoresync

Synopsis: `set sessionautoresync <value>|default`

This boolean parameter controls the resynchronization behavior when the parameter sessionsyncack is also active at the same time.

If sessionautoresync is set to 1 (active), then the current VRRP master starts a complete resynchronization of the session table towards a freshly started (or restarted) backup node.

The default value of this parameter is 1 (active).

The resynchronization process is the same as a “resync” command would have been given on the CLI of the VRRP master.

3.3.29.33 set sessionarrtimeout

Synopsis: `set sessionarrtimeout <value>|default`

This boolean parameter controls how long a freshly started backup is attempting to request for resynchronization from the current VRRP master. The parameters sessionautoresync and sessionsyncack must both be also active (1) in order to have an effect.

If sessionarrtimeout is set to 0, the backup will request for resynchronization infinitely (until it receives a session table resync request acknowledge). The default of this parameter is 60 (one minute), the maximum is 3600 seconds (one hour).

3.3.29.34 set sessiongclimit

Synopsis: `set sessiongclimit <value>|default`

This parameter controls how many outdated session table entries per second may be automatically reclaimed by the internal garbage collection mechanism. The default is 100,000, so it may take about 10 seconds to reclaim 1,000,000 outdated entries on an idle BalanceNG system.

3.3.29.35 set sessiondlimit

Synopsis: `set sessiondlimit <value>|default`

This parameter controls the number of session table entries being displayed interactively by the “show session” command. This parameter has a minimum and default of 10, and a maximum of 1000 entries.

Example:

```
bng# set sessiondlimit 20
bng# show conf
  //         configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  set       {
            sessiondlimit 20
  }
  //        end of configuration
bng#
```

3.3.29.36 set sessionscan

Synopsis: `set sessionscan <value>|default`

This parameter controls how many sessions are internally scanned and tested for timeout per second. Additionally BalanceNG performs the same session timeout test every time a session is being looked up in the session table.

The minimum is one per second, the maximum is 20 per second, the default is 10 per second.

Example:

```
bng# set sessionscan 2
bng# show conf
  //        configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  set       {
            hcportoffset 1024
            sessionscan 2
            vrrppreempt 1
            vrrpmasterdown 10
  }
  //        end of configuration
bng#
```

3.3.29.37 set sessionscanbup

Synopsis: `set sessionscanbup <value>|default`

This parameter controls how many sessions are internally scanned and tested for timeout per second if the node is in VRRP “backup” state.

The minimum is one per second, the maximum is 1000 per second, the default is 100 per second.

3.3.29.38 set sessionsync

Synopsis: `set sessionsync <value>|default`

This boolean parameter activated session table synchronization and state replication from the current active master node to the backup node(s). It has to be set to 1 on both the master and the backup to be active.

The default value of this parameter is 1 (active).

Session table synchronization uses either a BalanceNG specific VRRP extension or a bngsync protocol message. Session table state information is thus being propagated in a controlled manner (see parameter sessionsynciv below).

It is recommended to set the parameter vrrppreempt to 0 at the same time sessionsync is set to 1 (both is the default in current BalanceNG releases).

Example:

```
bng# set sessionsync 1
bng# show conf
  //        configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  set       {
            sessionsync 1
```

```
            vrrppreempt 0
  }
  //        end of configuration
bng#
```

3.3.29.39 set sessionsyncack

Synopsis: `set sessionsyncack <value>|default`

This parameter controls the acknowledgement of single session-table entries during the continuous synchronization process. The default value is 1 (active/on). If set to 1 (on), the behavior is as follows:

The current VRRP master node accepts VRRP extension type 6 packets and updates the local session table entry and counters accordingly.

The current VRRP backup node acknowledges each VRRP extension type 4 packet by sending the acknowledgment (type 6) back directly after the local session table update.

Example:

```
bng# set sessionsyncack 1
bng#
```

3.3.29.40 set sessionsyncetype

Synopsis: `set sessionsyncetype <value>|default`

This parameter controls the Ethertype (the two bytes directly following the Ethernet source address) of session synchronization packets and the session synchronization acknowledgement packets.

If this parameter is set to 0 (the default), the standard Ethertype 0x08, 0x00 is used.

If this parameter is set to 1, a non-standard Ethertype of 0x8b, 0x00 is used instead.

Example:

```
bng# set sessionsyncetype 1
```

3.3.29.41 set sessionsynciv

Synopsis: `set sessionsynciv <value>|default`

This parameter controls how often the information of a current actively used session is being notified to listening backup nodes (in seconds). The default value is 10 seconds. It could be increased to lower the number of notification packets in the network.

Example:

```
bng# set sessionsynciv 10
bng# show conf
  //        configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  set       {
            vrrppreempt 0
            sessionsync 1
            sessionsynciv 20
  }
  //        end of configuration
bng#
```

3.3.29.42 set sessiontimeout

Synopsis: `set sessiontimeout <value>|default`

This parameter controls how long a n inactive session is being remembered by BalanceNG. After having reached this timeout threshold the session will be removed from the session table at the next opportunity.

The minimum value of this parameter is 10 seconds, the maximum 172800 seconds (48 hours). The default value is 600 (10 minutes).

Example:

```
bng# set sessiontimeout 60
bng# show conf
  //        configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  set       {
            sessiontimeout 60
            hcportoffset 1024
            sessionscan 2
            vrrppreempt 1
            vrrpmasterdown 10
  }
  //        end of configuration
bng#
```

3.3.29.43 set stickytarget

Synopsis: `set stickytarget <value>|default`

This boolean parameter controls the update of the layer 2 forwarding table when packets from known targets are received. If set to 1 (the default), the forwarding table is not updated if a packet from a known target is received on a different additional interface. If set to 0, the forwarding table will be updated.

3.3.29.44 set strictrouting

Synopsis: `set strictrouting <value>|default`

This boolean parameter controls the processing of incoming packets. If set to “1”, packets are only accepted for forwarding if they are received at the current VRRP Ethernet address. VRRP needs to be enabled for this parameter. The default value of this parameter is “1” (enabled). The parameter setting is necessary if there are multiple BalanceNG instances running on one machine listening on the same IPv4 network and if BalanceNG runs connected to a VMware vswitch in a virtual environment.

The functionality controlled by this parameter is integrated in the following modules:

- The “strict” experimental module
- The “slb” and “llb” modules.

Example:

```
bng# set strictrouting 1
```

3.3.29.45 set sweeponreload

Synopsis: `set sweeponreload 0|1`

This parameter allows to disable session table processing during the second pass of a reload command. The default value is 1 (active). Switching off session table processing during reload may be helpful if the session table is very large and if it can be assured by other means that the session table contents are correct also with the new configuration.

3.3.29.46 set syncackbdelay

Synopsis: `set syncackbdelay <value>|default`

This parameter controls an initial delay, after which a freshly started node in VRRP backup state starts to accept session table sync advertisements. The default value of this parameter is 10 seconds.

This parameter is effective only if sessionsyncack is set to 1 (enabled).

3.3.29.47 set syncackmaxps

Synopsis: `set syncackmaxps <value>|default`

This parameter controls the maximum number of session table sync requests per second issued by the VRRP master. The default value is 2000 session table entries per second. Please consider to establish a separate sync interface for higher values of this parameter.

This parameter is effective only if sessionsyncack is set to 1 (enabled).

3.3.29.48 set syncackresend

Synopsis: `set syncackresend <value>|default`

This parameter controls the number of seconds after which a not yet acknowledged session table sync advertisement is resent again (timeout). A session table sync advertisement may either be sent as a non-standard VRRP message type 4 (deprecated) or as a bngsync message (type 0 subtype 4).

This parameter is effective only on the current VRRP master and only if sessionsyncack is set to 1 (enabled) at the same time.

3.3.29.49 set syncackwsize

Synopsis: `set syncackwsize <value>|default`

This parameter controls the number of session table entry sync requests sent out once per second by the VRRP master.

This parameter is effective only if sessionsyncack is set to 1 (enabled).

3.3.29.50 set tarpitrealto

Synopsis: `set tarpitrealto <value>|default`

This parameter controls how long the tarpit module remembers a real existing and not simulated IP address. The minimum value is 60 seconds (one minute), the maximum is 86400 seconds (1 day) and the default is 14400 seconds (4 hours).

3.3.29.51 set tarpittrapto

Synopsis: `set tarpittrapto <value>|default`

This parameter controls how long the tarpit module simulates or represents a specific requested IP address. The minimum value is 60 seconds (one minute), the maximum is 86400 seconds (1 day) and the default is 600 seconds (10 minutes).

3.3.29.52 set tcphalfopen

Synopsis: `set tcphalfopen <value>|default`

This boolean parameter controls the behavior of the tcpopen and tcpopen6 health checks. If set to 1 (active), both TCP health checks perform a TCP half open health check by immediately sending a RST packet after receiving a SYN-ACK packet from the target. If set to 0 (disabled) the full TCP 3-way handshake is performed followed by a immediate close.

If enabled (active) this has the benefit that the health check activity does usually not appear in any log file of the listening server software.

The value 1 (active) is the default value of this parameter.

3.3.29.53 set vrrpmasterdown

Synopsis: `set vrrpmasterdown <value>|default`

This parameter controls the time interval in seconds after which a VRRP backup not receiving Master advertisements will declare the master to be down. This parameter implements the "Master_Down_Interval" of RFC3768 (see Reference /3/).

The minimum and default of this parameter is 3 seconds, the maximum is10 seconds.

Example:

```
bng# set vrrpmasterdown 10
bng# show conf
  //        configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  remark    "$Id: bng.conf,v 1.1 2022/11/08 13:59:31 root Exp
  set       {
            hcportoffset 1024
            vrrpmasterdown 10
  }
  //        end of configuration
bng#
```

3.3.29.54 set vrrppreempt

Synopsis: `set vrrppreempt <value>|default`

This parameter controls whether a higher priority VRRP Backup node preempts a lower priority Master. This is a boolean parameter with 1 = TRUE and 0 = FALSE. **The default of this parameter is 0 (preemption *not* active).**

This parameter implements the parameter "Preempt_Mode" of RFC3768 (see Reference /3/).

Here a potentially simpler explanation:

If a previously failed VRRP master (with a priority lower than 255) comes back into operation it will stay gently in background as a VRRP backup node if vrrppreempt equals 0. If vrrppreempt equals 1 it will force to be master again as quickly as possible.

Note: If session state replication is being used with setting sessionsync to 1, it is a good idea to set vrrppreempt to 0 at the same time. This allows the new joining node (the previous master) to learn back the currently active session table.

Note vrrppreempt defaults to 0 and sessionsync to 1 for administration convenience (sessions are being synced per default).

Example:

```
bng# set vrrppreempt 0
```

```
bng# show conf
  //        configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  remark    "$Id: bng.conf,v 1.1 2022/11/08 13:59:31 root Exp
  set       {
            hcportoffset 1024
            vrrppreempt 0
            vrrpmasterdown 10
  }
  //        end of configuration
bng#
```

3.3.29.55 set vrrppreemptts

Synopsis: `set vrrppreemptts <value>|default`

This parameter (VRRP Preemption Threshold) allows further control of the preemption behavior of a higher priority backup. If vrrppreempt is set to 1 on a higher priority backup, it immediately tries to become VRRP master of the VR. If this parameter is not equal to 0, it is subtracted from the local priority before that comparison.

Together with the VRRP tracking features, this allows fine tuning of the failover if external resources are failing on the master still being available on the backup.

3.3.29.56 set vrrpstateplugin

Synopsis: `set vrrpstateplugin <value>|default`

If this boolean parameter is set to 1 (active), a VRRP state change is reported to all server plugins (see “server <n> plugin” for more information). The default value is 0 (inactive).

If this parameter is active (1) and a node becomes backup, the word “BACKUP” is written to stdin of all server plugins and a the word “MASTER” if a node becomes VRRP master.

The plugin must acknowledge the receipt of the state change information by sending back a line containing one single character (a “0”) and a line feed **as fast as possible**.

3.3.29.57 set vrrpv3ip

Synopsis: `set vrrpv3ip <value>|default`

This parameter controls the IP protocol which is used to send VRRP v3 advertisements and is only effective if parameter vrrpversion is set to “3” at the same time.

BalanceNG is able to generate and send VRRP v3 over IPv4 packets if the “network real” and a “network virt” addresses are configured for the network being referenced by the vrrp section. The “network virt” IPv4 address is advertised in the “IPvX” field in that case (count 1).

BalanceNG is able to generate and send VRRP v3 over IPv6 packets if the “network real6” and a “network virt6” addresses are configured for the network being referenced by the vrrp section.The “network virt6” IPv6 address is advertised in the “IPvX” field in that case (count 1).

The possible values of this parameter range from 4 to 6 and have the following meaning and functionality:

```
value       functionality

4           VRRP v3 advertisements are sent (only) over IPv4
```

```
5           “automatic mode”, default
            VRRP v3 advertisements are sent over IPv6 if possible,
            otherwise over IPv4

6           VRRP v3 advertisements are sent (only) over IPv6
```

3.3.29.58 set vrrpversion

Synopsis: `set vrrpversion <value>|default`

This parameter defines the version of the VRRP protocol being used. If set to 2, BalanceNG uses the VRRP V2 protocol as defined by RFC 3768 and if set to 3, BalanceNG uses the VRRP V3 protocol as defined by RFC 5798.

The default value of this parameter is 3.

3.3.29.59 set xstlog

Synopsis: `set xstlog 0|1`

This parameter allows to log the creation of a new session table entry on the current VRRP master for debugging purposes. The default is 0 (off). This parameter should not be enabled during production use in order to avoid any overhead.

3.3.29.60 set haswitch:isolate

Synopsis: `set haswitch:isolate 0|1|default`

This boolean parameter controls a special LAN isolation mode for the haswitch module. If set to 1 (active), LANs connected to interfaces with scope “internal” can only communicate with LANs connected to interfaces with scope “external” (but not among each other).

This allows to provide internet connectivity to a set of insecure or suspect devices and to protect a trusted LAN accessing the same router at the same time (for example).

The default of this parameter is 0 (inactive).

3.3.29.61 set haswitch:timeout

Synopsis: `set haswitch:timeout <value>|default`

This parameter controls the timeout of haswitch forwarding table entries as kept in the synchronized session table. The default is one hour (3600 seconds), the minimum 10 seconds and the maximum 24 hours (86400 seconds).

3.3.30 snatrange <from> <to>

Synopsis: `snatrange <from> <to>`

`no snatrange`

This command specifies a range of IP4 addresses which will be used for source NAT (SNAT) processing if the “slb” module is active and if SNAT is enabled for a specific virtual server (see “server <n> snat”).

3.3.31 softdisable target <n>

Synopsis: `softdisable target <n>`

`softdisable targets <n1>,<n2>, ...`

This command sets one or more specified enabled targets into a special "softdisable" state. Targets in that state are still working for already existing sessions, but no new sessions will be allocated by the load balancing target selection methods.

This is very useful for smoothly taking a target machine out of service for maintenance and other service activities. As soon as the session count of the target is 0 (displayed at "show target <n>") the target machine can be safely taken into maintenance.

A target in "softdisable" state may be taken back into normal load balancing distribution with the "`enable target`" command (e.g. "`enable target 1`").

The softdisable state may be entered administratively during runtime only and is not part of the configuration and the configuration file.

Example:

```
bng# show target 1
target 1
  ipaddr    172.17.2.90
  port      any
  network   1
  protocol  any
  sessions  381
  trackval  0
  status    operational
  arp       up
  ping      up
  psent     4123
  bsent     123526
  prcvd     2345
  brcvd     213456
bng# softdis target 1
bng# show target 1
target 1
  ipaddr    172.17.2.90
  port      any
  network   1
  protocol  any
  softdis   *active*
  sessions  381
  trackval  0
  status    operational
  arp       up
  ping      up
  psent     4234
  bsent     234454
  prcvd     3345
  brcvd     323456
bng#
```

3.3.32 target <n>

Synopsis: `target <n> <subcommand> <value>`

This command is used to configure the parameters of a BalanceNG target. In general this is only possible if the target is in the "unregistered" state (see the server and target state explanations at the "`register`" command).

The target index may range from 1 to 1024, such allowing a total of 1024 target sections per BalanceNG instance.

BalanceNG is capable of handling 1024 independent targets. Each target may be referenced by the targets list of a specific server one or multiple times.

3.3.32.1 target <n> agent

Synopsis: `target <n> agent <parameters>|"off"`

Activates the `agent` healthcheck and load collector of the specified target using the supplied parameters (or switches the `agent` healthcheck off when "`off`" is supplied).

The agent healthcheck communicates with the bngagent UDP protocol with a bngagent program running on the physical target machine (see the bngagent chapter for more informations).

The parameter list consists of three numerical values, separated by commas. The first value is the UDP port being addressed on the real target machine. The second parameter is the interval in seconds to perform the agent healthcheck. The third parameter specifies the number of seconds with no answer to declare the target inoperational.

Example 1 (checking every 10 seconds on port 2000, 30 seconds of missing replies for declaring target inoperational):

```
bng# target 1 agent 2000,10,30
```

Example 2 (switching agent healthcheck off):

```
bng# target 1 agent off
```

3.3.32.2 target <n> agent6

Synopsis: `target <n> agent6 <parameters>|"off"`

Activates the IPv6 agent healthcheck and load collector of the specified target using the supplied parameters (or switches the `agent` healthcheck off when "`off`" is supplied).

The agent6 healthcheck communicates with the bngagent UDP over IPv6 protocol with a bngagent program running on the physical target machine (see the bngagent chapter for more informations). The

The parameter list consists of three numerical values, separated by commas. The first value is the UDP port being addressed on the real target machine. The second parameter is the interval in seconds to perform the agent healthcheck. The third parameter specifies the number of seconds with no answer to declare the target inoperational.

Either “target <n> agent” or “target <n> agent6” may be specified, it's not possible to run both at the same time.

Example (checking every 10 seconds on port 2000, 30 seconds of missing replies for declaring target inoperational):

```
bng# target 1 agent6 2000,10,30
```

The bngagent needs to be running in IPv6 mode on the target server and may be started like

this:

```
# bngagent -6 2000
```

3.3.32.3 target <n> ascript

Synopsis: `target <n> ascript <script>,<interval>,<timeout>`

This command allows local execution of an agent script or program. The script just needs to print one single line containing the determined integer agent value.

Before the script invocation string is passed to popen(), a set of symbols or variables is literally replaced once in the string (as it's done with “target <n> script”).

The symbols and their replacements are as follows:

```
Symbol      Replacement
-------------------------------------------------
$ipaddr$    IPv4 address of the target
$ipaddr6$   IPv6 address of the target
$port$      Port number of the target if specified
            ("0" otherwise)
$status$    Healthcheck status
$target$    Number of the target
```

Please note, that it's possible either to use “target <n> agent” or “target <n> ascript”, but not both at the same time. Additionally, “target <n> ascript” should be used in conjunction with additional healthchecks (like “target <n> script”).

Example:

The following line

```
target 1 ascript “/opt/BalanceNG/snmpload.sh $ipaddr$”,10,600
```

calls the snmpload.sh script which is part of the BalanceNG distribution (located either in /opt/BalanceNG or directly in the Linux tarball distribution).

The snmpload.sh script allows to retrieve the CPU load from a Windows system, here's the script source code:

```
#/bin/sh

VALUES=`snmpwalk -v1 -c public -O qv $1 .1.3.6.1.2.1.25.3.3.1.2
2>/dev/null`

if [ "$?" != "0" ]
then
  echo 101
  exit 0
fi

SUM=0
COUNT=0

for VALUE in $VALUES
do
  COUNT=`expr $COUNT + 1`
  SUM=`expr $SUM + $VALUE`
```

```
done

if [ "$COUNT" = 0 ]
then
  echo 101
  exit 0
else
  RESULT=`expr $SUM / $COUNT`
  RESULT=`expr $RESULT + 1`
  echo $RESULT
  exit 0
fi
```

3.3.32.4 target <n> alert

Synopsis: `target <n> alert <alertscript>`

This command specifies an external script or program which is called or executed as soon and every time the associated target gets down or inoperational. The alertscript has to be specified in double quotes and will be executed by a helper thread with the system() C-library function call. The call of this external script happens only once at every state change (e.g. from "operational" to "down" or from "initial" to "down").

This mechanism could be useful for e.g. sending a SNMP trap to a network management system or for sending an email if a target goes down (gets "inoperational").

Before the alertscript string is passed to system() a set of symbols or variables is literally replaced once in the string. The symbols and their replacements are as follows:

Symbol	Replacement
$ipaddr$	IPv4 address of the target
$ipaddr6$	IPv6 address of the target
$port$	Port number of the target if specified ("0" otherwise)
$status$	Healthcheck status
$target$	Number of the target

Example:

```
# bng control
BalanceNG: connected to PID 16624
bng# target 1 {
bng+ ipaddr 10.1.2.2
bng+ ping 2,10
bng+ alert "/usr/local/sbin/alertmail $ipaddr$ $target$"
bng+ }
bng# commit target 1
```

3.3.32.5 target <n> aoffset

Synopsis: `target <n> aoffset <offset>`

This command is a synonym for "`target <n> offset`".

3.3.32.6 target <n> ascale

Synopsis: `target <n> ascale <scale>`

This command is a synonym for "`target <n> scale`".

3.3.32.7 target <n> autodisable

Synopsis: `target <n> autodisable on|off`

If the autodisable feature is set to “on” for a specific target, this target will be automatically disabled s soon as it gets inoperational or “down” according to the associated health checks. This automatic operation is the same as entering “disable target <n>” at the same time.

The autodisable feature is switched off per default (and is not visible in the configuration in that state).

Example:

```
NodeA# edit target 1
NodeA# target 1 autodisable on
NodeA# commit target 1
  ...
NodeA# show log
  ...
  2022/11/08 22:14:25 5 target 1 down (arp:up,ping:up,tcpopen:down)
  2022/11/08 22:14:25 5 target 1 automatically disabled (autodisable=on)
NodeA# show targets
    # ipaddr           port prt net srv sessions status      name
  --- ---------------- ---- --- --- --- -------- ----------- ---------------
    1 10.10.2.21         53 any   1   1        0 disabled    test1
NodeA#
```

3.3.32.8 target <n> autodisablecount

Synopsis: `target <n> autodisablecount <value>`

`target <n> autodisablecount default`

This command controls the number of target state transitions from “operational” to “down” until a target is automatically disabled (target <n> autodisable needs to be active). The default of this value is 1, the maximum 100000.

Example:

```
NodeA# edit target 1
NodeA# target 1 autodisable on
NodeA# target 1 autodisablecount 10
NodeA# commit target 1
NodeA# show conf target 1
  target    1 {
            ipaddr 172.17.2.30
            port 80
            protocol tcp
            tcpopen 80,3,10
            autodisable on
            autodisablecount 10
  }
```

3.3.32.9 target <n> dsr

Synopsis: `target <n> dsr enable|disable`

"`target <n> dsr enable`" enables the "Direct Server Return" feature for the specified target, "`target <n> dsr disable`" disables the DSR feature.

This feature is disabled per default, and not shown in the config file if disabled.

If DSR is enabled for the specified target packets to a virtual server are forwarded to the target Layer 2 address with the virtual server destination IP address unchanged. The virtual server address has to be added as an alias to the Loopback ("lo") Interface of the Target machine.

If the associated virtual server has Proxy Mode enabled (“proxy enable”), the DSR functionality is only active für clients connecting from external (e.g. forwarded by a router).

Example:

```
bng# target 1 {
bng+ ipaddr 10.1.1.4
bng+ port 80
bng+ protocol tcp
bng+ tcpopen 80,3,10
bng+ dsr enable
bng+ }
```

3.3.32.10 target <n> ipaddr

Synopsis: `target <n> ipaddr <IPv4 address>`

`target <n> ipaddr none`

This specifies the IPv4 address of the target with the specified index <n>, “none” removes the current IPv4 address.

Example:

```
bng# target 2 ipaddr 10.1.1.3
bng# target 3 {
bng+ ipaddr 10.1.1.4
bng+ }
```

3.3.32.11 target <n> ipaddr6

Synopsis: `target <n> ipaddr6 <IPv6 address>`

`target <n> ipaddr6 none`

This specifies the IPv6 address of the target with the specified index <n>, “none” removes the current IPv6 address.

Example:

```
bng# target 2 ipaddr6 fe80::230:48ff:fe93:4d02
bng#
```

3.3.32.12 target <n> lgrp

Synopsis: `target <n> lgrp <A-Z>|none|off`

This command associates the target with a specific location group (or removes that association with “none” or “off”). Please take a look at “ipdb”, “lgrp” and “server <n> ipdb” for further information about location based server load balancing.

Example:

```
bng# lgrp A "US,GB"
bng# edit target 1
bng# target 1 lgrp A
bng# commit target 1
bng# show conf
...
lgrp      {
            A "US,GB"
            B "*,!A"
}
...
  target    1 {
            ipaddr 172.17.2.91
            port 53
            lgrp A
            ping 5,12
            tcpopen 53,5,12
            dsr enable
  }
...
bng# show lgrp A
  grp A (solved)
  txt US,GB
  key description
  --- -----------------------------------
   GB UNITED KINGDOM
   US UNITED STATES
  --- -----------------------------------
    2 total entries
bng#
```

3.3.32.13 target <n> maxagent

Synopsis: `target <n> maxagent <threshold>|0`

This command assigns a maximum score threshold for target <n> and works together with the “agent” directive. If a target total and absolute agent score exceeds the supplied threshold this particular target is silently taken out of the current load balancing distribution. A value of 0 (default) sets this threshold to “unlimited”.

Example:

```
bng# target 2 maxagent 2000
bng#
```

3.3.32.14 target <n> maxgrpsessions

Synopsis: `target <n> maxgrpsessions <threshold>|0`

This command assigns a maximum session threshold for target <n>, where all sessions of the targets sessiongroup are counted together (see target <n> sessiongroup). If the sum of all

sessions of targets belonging to the same group exceeds that value, this particular target is silently taken out of the current load balancing distribution. A value of 0 (default) sets this threshold to “unlimited”.

Example:

```
bng# target 2 sessiongroup 2
bng# target 2 maxgrpsessions 5000
```

3.3.32.15 target <n> maxsessions

Synopsis: `target <n> maxsessions <threshold>|0`

This command assigns a maximum session threshold for target <n>. If a target total and absolute number of sessions exceeds the supplied threshold this particular target is silently taken out of the current load balancing distribution. A value of 0 (default) sets this threshold to “unlimited”.

Example:

```
bng# target 2 maxsessions 2000
bng#
```

3.3.32.16 target <n> name

Synopsis: `target <n> name <name>|"none"`

Assigns a target a name for informational purposes. The string "`none`" as the name arguments deletes the name from the specified target.

The target name may be embedded in double quotes to specify a name containing spaces. Specifying an empty string in double quotes also removes the current name definition.

Example:

```
bng# edit target 1
bng# target 1 name test-target-1
bng# commit target 1
WARNING: target 1 in enabled state but not referenced
bng# show targets
    # ipaddr           port prt net srv status      name
  ------------------------------------------------------------------
    1 10.1.1.3          any any   2   0 down        test-target-1
bng#
```

3.3.32.17 target <n> offset

Synopsis: `target <n> offset <offset>`

This command allows together with "`target <n> ascale`" the modification of the return value of bngagent by applying a linear function to it. This function looks like this:

`<effective agent data> = <returned agent data> * <target ascale> + <target aoffset>`

The `aoffset` parameter has a default of 0 (where it is not displayed in the configuration file). The `ascale` parameter has a default of 1.0 (also not being displayed).

Both `aoffset` and `ascale` always have to be positive. If the unsigned integer return value is being scaled down to integer 0, then this result will be replaced by 1.

The following recommendation can be made:

- A more powerful target machine should become a scale **smaller** than the less powerful machines.
- A less powerful machine which should e.g. not be used from the very beginning should be supplied with a `aoffset > 0`. A machine with the default `aoffset` of 0 takes will receive traffic / load from the very beginning.

  The following example shows how easily the effective agent value may be modified by applying ascale and aoffset parameters. This together with the agent distribution method allows a very expressive optimization of load distribution even to very different machines.

Example:

```
root@bng1:~ # /etc/init.d/bng control
BalanceNG: connected to PID 722
bng# show target 1
target 1
  ipaddr    10.1.1.1
  port      any
  network   2
  protocol  any
  status    operational
  arp       up
  agent     4
  eff_agent 4
bng# edit target 1
bng# target 1 scale 2.0
bng# commit target 1
WARNING: target 1 in enabled state but not referenced
bng# show target 1
target 1
  ipaddr    10.1.1.1
  port      any
  network   2
  protocol  any
  status    operational
  arp       up
  agent     2
  eff_agent 4
bng# edit target 1
bng# target 1 offset 10
bng# commit target 1
WARNING: target 1 in enabled state but not referenced
bng# show target 1
target 1
  ipaddr    10.1.1.1
  port      any
  network   2
  protocol  any
  status    operational
  arp       up
  agent     1
  eff_agent 12
bng#
```

3.3.32.18 target <n> ping

Synopsis: `target <n> ping <parameters>|"off"`

Activates the ping healthcheck of the target or switches it off (by supplying "off"). The parameters consist of two values, separated by a comma. The first value is the interval in seconds to send an ICMP echo request packet to the target. The second value is the time to declare a target inoperational if no ICMP echo response packet is received in that time.

Example:

```
bng# target 2 ipaddr 10.1.1.2
bng# target 2 ping 2,10
bng# commit target 2
bng# show target 2
target 2
  ipaddr    10.1.1.2
  port      any
  network   2
  protocol  any
  status    operational
  arp       up
  ping      up
bng#
```

3.3.32.19 target <n> ping6

Synopsis: `target <n> ping6 <parameters>|"off"`

Activated the ping6 (IPv6) healthcheck of the target or switches it off (by supplying "off"). The parameters consist of two values, separated by a comma. The first value is the interval in seconds to send an ICMP echo request packet to the target. The second value is the time to declare a target inoperational if no ICMP echo response packet is received in that time. The target needs to have an IPv6 address specified.

Example:

```
bng# target 2 ipaddr6 2001:DB8::1
bng# target 2 ping6 3,10
```

3.3.32.20 target <n> port

Synopsis: `target <n> port <port>|"any"`

With this command a specific port can be associated to a target. Supplying "any" reverts this back to "any" port. An associated port of a target restricts load balancing actions to this port and allows TCP and UDP port translation (server uses different port than the associated target).

Together with the "target <n> protocol" command the load balancing actions may be additionally restricted to just one protocol family on that port (TCP or UDP).

Example:

```
bng# target 4 ipaddr 10.1.1.5
bng# target 4 port 8080
bng# target 4 protocol tcp
bng# target 4 ping 2,20
bng# target 4 tcpopen 8080,5,20
bng# commit target 4
```

```
WARNING: target 4 in enabled state but not referenced
bng# show target 4
target 4
  ipaddr    10.1.1.5
  port      8080
  network   2
  protocol  tcp
  status    down
  arp       down
  ping      down
  tcpopen   down
bng#
```

3.3.32.21 target <n> protocol

Synopsis: `target <n> protocol "tcp"|"udp"|”sctp”|"any"`

Restricts load balancing participation of the specified target either to TCP, UDP or SCTP (or reverts back to the default of any protocol by specifying "any").

Together with "`target <n> port`" the matching rules may be restricted to packets from a specific port and protocol.

Example:

```
bng# target 4 protocol tcp
bng# commit target 4
```

3.3.32.22 target <n> pseudo

Synopsis: `target <n> pseudo "enable"|"disable”`

This directive declares the target to be a special pseudo target. A pseudo target needs no associated server. A ping (IPv4 echo request) health-check (“target <n> ping”) is sent with the VRRP MAC source address and the “network <n> virt” IPv4 source address only if the node is currently VRRP master. A pseudo target may be useful in order to keep the forwarding tables of external devices updated in respect to the VRRP virtual router MAC address.

3.3.32.23 target <n> router

Synopsis: `target <n> router <IPv4-address>`

This target specifies a target specific gateway where all target related traffic should be directed instead of expecting the target locally reachable.

Such a router (gateway) specification is valid for all internal health checks and all target related load balancing traffic and allows to address indirectly reachable services in BalanceNG load balancing.

Example:

```
bng# edit target 1
bng# target 1 router 172.17.2.254
bng# commit target 1
```

3.3.32.24 target <n> scale

Synopsis: `target <n> ascale <scale>`

This target parameter allows together with `aoffset` the modification of the bngagent return

value in terms of a linear function. Please see the more detailed explanation and example at the "`target <n> aoffset`" command.

Example:

```
bng# edit target 1
bng# target 1 scale 2.0
bng# commit target 1
WARNING: target 1 in enabled state but not referenced
bng# show target 1
target 1
  ipaddr    10.1.1.1
  port      any
  network   2
  protocol  any
  status    operational
  arp       up
  agent     2
  eff_agent 4
bng# show conf
  //        configuration taken Sun Aug 24 22:56:40 2008
  //        BalanceNG 5.016 (created 2022/11/08)
  interface eth0
  interface eth1
  network   1 {
            addr 10.55.55.0
            mask 255.255.255.0
            real 10.55.55.190
            virt 10.55.55.191
            interface eth0
  }
  network   2 {
            addr 10.1.1.0
            mask 255.255.255.0
            real 10.1.1.190
            virt 10.1.1.191
            interface eth1
  }
  register  networks 1,2
  enable    networks 1,2
  target    1 {
            ipaddr 10.1.1.1
            agent 5000,5,0
            offset 10
            scale 2
  }
  register  target 1
  enable    target 1
  //        end of configuration
bng#
```

3.3.32.25 target <n> screate

Synopsis: `target <n> screate enable|disable`

This command enables or disables session creation initiated by target originated traffic for the “slb” module. The default for this setting is “disabled” (where no setting is shown in the configuration file).

Example:

```
# bng ctl
BalanceNG: connected to PID 5427
bng# edit target 1
bng# target 1 screate enable
bng# commit target 1
bng#
```

3.3.32.26 target <n> script

Synopsis: `target <n> script <healtcheckscript>,<interval>,<timeout>`

This command specifies an external health check script which will be called at the given interval in seconds. If the script returns with an exit code not equal to zero (0) or if the invocation of that script fails or if that script does not return within the specified timeout the target status will change to "inoperational" or "down".

The script is being executed by a helper thread using the system() library function. The external script is not aware of the TCP/IP stack of BalanceNG and operates under the host operating system as usual.

With that mechanism arbitrary any custom health check method can be interfaced or implemented easily.

Before the healthcheckscript string is passed to system() a set of symbols or variables is literally replaced once in the string. The symbols and their replacements are as follows:

```
Symbol      Replacement
-------------------------------------------------
$ipaddr$    IPv4 address of the target
$ipaddr6$   IPv6 address of the target
$port$      Port number of the target if specified
            ("0" otherwise)
$target$    Number of the target
```

Many different programs can be easily interfaced that way (e.g. "ping", "wget, "mon") or any other custom script or program.

Example:

This example calls an external monitor script to implement a HTTP lookup of a specific URL (http.monitor from the "mon" package, available at http://www.kernel.org/software/mon). The script is called every two seconds, the target gets inoperational/down if the script fails (returns something else than 0) or if it does not return within 7 seconds.

```
# ./bng control
BalanceNG: connected to PID 16624
bng# edit target 1
bng# target 1 script "/usr/lib/mon/mon.d/http.monitor -p $port$   -u
/healtcheck.cgi $ipaddr$",2,7
bng# commit target 1
```

3.3.32.27 target <n> script6

Synopsis: `target <n> script6 <healtcheckscript>,<interval>,<timeout>`

This command implements a second scripting health-check (in addition to target N script) which may be used to separate external IPv4 and IPv6 scripts. Parameters and usage is the same as “target N script” (see above).

3.3.32.28 target <n> sessiongroup

Synopsis: `target <n> sessiongroup default|<value>`

Targets may be optionally grouped in “target session groups”. Per default all targets belong to the target sessiongroup 0. The sessiongroup parameter may be specified in the rage from 0 to 100. Together with “target <n> maxgrpsessions” this parameter allows a simple way to limit number of sessions per target group to a desired value.

Example:

```
bng# edit target 4
bng# target 4 sessiongroup 2
bng# target 4 maxgrpsessions 5000
bng# commit target 4
bng#
```

3.3.32.29 target <n> sessionid <handler>

This command associates a specific session handler to a particular target. The “slb” (Server Load Balancing) Module needs to be part of the current module chain.

3.3.32.29.1 sip

The sessionid is based on SIP/UDP Call-ID only.

3.3.32.29.2 src

The sessionid is based only on the source IP address.

3.3.32.29.3 src+dstport

The sessionid is based on the source IP address and the destination port.

3.3.32.29.4 src+port

The sessionid is based on the source IP address and the source port.

3.3.32.29.5 src+ports

The sessionid is based on the source IP address and both the source and destination ports.

3.3.32.29.6 src+tag

The sessionid is based on the source IP address and the sessiontag (see “server <n> sessiontag” and “target <n> sessiontag”).

3.3.32.29.7 dst

The sessionid is based only on the destination IP address.

3.3.32.29.8 dst+port

The sessionid is based on the destination IP address and the destination port.

3.3.32.29.9 dst+ports

The sessionid is based on the destination IP address and both the source and destination ports.

3.3.32.29.10 dst+srcport

The sessionid is based on the source IP address and the source port.

3.3.32.29.11 dst+tag

The sessionid is based on the destination IP address and the sessiontag (see “server <n> sessiontag” and “target <n> sessiontag”).

3.3.32.30 target <n> sessiontag <tag>

Synopsis: target <n> sessiontag <tag>|default

This command modifies a target specific session tag. The tag may be used by the sessionid “dst+tag” thus allowing to group several server ports to be handled by the same session table entry. The target sessiontag and sessionid is processed on the return path for non-DSR (direct server return) topologies. A target sessiontag is either 0 (default) or any positive integer number.

3.3.32.31 target <n> tcpopen

Synopsis: target <n> tcpopen <parameters>|"off"

Activates a simple TCP open healthcheck (or switches it of with "off"). The parameter list expects three numeric parameters separated by commas: The first is the actual port to test (which may be different than the target load balancing port), the second is the interval to check in seconds and the third is the interval to declare the target down when no response is received in that period.

Example:

```
bng# edit target 4
bng# target 4 tcpopen 80,2,10
bng# commit target 4
bng#
```

3.3.32.32 target <n> tcpopen6

Synopsis: target <n> tcpopen6 <parameters>|"off"

Activates a simple IPv6 TCP open healthcheck (or switches it of with "off"). As with “target <n> tcpopen”, the parameter list expects three numeric parameters separated by commas: The first is the actual port to test (which may be different than the target load balancing port), the second is the interval to check in seconds and the third is the interval to declare the target down when no response is received in that period. A IPv6 address needs to be specified for the target.

Example:

```
bng# edit target 4
bng# target 4 ipaddr6 2001:db8:ffff::1:2
bng# target 4 tcpopen6 80,2,10
```

```
bng# commit target 4
bng#
```

3.3.32.33 target <n> upalert

Synopsis: `target <n> upalert <upalertscript>`

This command specifies an external script or program which is called or executed as soon and every time the associated target gets "up" or operational. The upalertscript has to be specified in double quotes and will be executed by a helper thread with the system() C-library function call. The call of this external script happens only once at every state change (e.g. from "down" to "operational" or from "initial" to "operational").

This mechanism could be useful for e.g. sending a SNMP trap to a network management system or for sending an email if a target goes up (gets "operational" again).

Before the upalertscript string is passed to system() a set of symbols or variables is literally replaced once in the string. The symbols and their replacements are as follows:

```
Symbol      Replacement
-------------------------------------------------
$ipaddr$    IPv4 address of the target
$ipaddr6$   IPv6 address of the target
$port$      Port number of the target if specified
            ("0" otherwise)
$status$    Healthcheck status
$target$    Number of the target
```

Example:

```
# bng control
BalanceNG: connected to PID 16627
bng# target 1 {
bng+ ipaddr 10.1.2.2
bng+ ping 2,10
bng+ upalert "/usr/local/sbin/upmail $ipaddr$ $target$"
bng+ }
bng# commit target 1
```

3.3.32.34 target <n> trackval

Synopsis: `target <n> trackval <value>`

This command associates a tracking value to the target. The default of this value is 0. If VRRP tracking is enabled and the VRRP priority is less than 255 and a target is enabled and down (not operational) the current VRRP priority is degraded by the tracking value "trackval" of the target.

Note: Tracking is enabled with “vrrp tracking enable”.

Example:

```
bng# show target 1
target 1
  ipaddr    172.17.2.90
  port      any
  network   1
  protocol  any
```

```
  sessions  0
  trackval  0
  status    operational
  arp       up
  ping      up
bng# edit target 1
bng# target 1 trackval 4
bng# commit target 1
bng# show target 1
target 1
  ipaddr    172.17.2.90
  port      any
  network   1
  protocol  any
  sessions  0
  trackval  4
  status    operational
  arp       up
  ping      up
bng#
```

3.3.32.35 target <n> via

Synopsis: `target <n> via <ipaddr>`

This command may be used as a synonym for “target <n> router”.

3.3.32.36 target <n> weight

Synopsis: `target <n> weight <value>`

This command associates a certain weight to a target, which is valid if the server distribution method “random” is used. The default weight of a target is 1 and is not displayed on “show conf”. A target weight is valid in the range of 1-100 inclusively and may reflect a percentage value.

The following examples instructs BalanceNG to choose direct 75% of new sessions to target 2 and 25% of new sessions to target 1 (if target 1 and 2 are being referenced by a server with distribution method “random”).

Example:

```
bng# target 1 weight 10
bng# target 2 weight 30
bng# commit targets 1,2
bng#
```

3.3.33 tnat

Synopsis: `tnat <target addr> <NAT addr> <protocol> <dest port>`

This command specifies selectively 1:1 Network Address Translation (NAT) for a specific destination port and protocol. This command is revertable using the "no" special command.

The "target addr" should be the address of a target. The "NAT addr" should be an exclusively reserved address in the outbound network (one reservation / allocation per tnat-entry). The parameter "protocol" may be either "udp" or "tcp".

The destination port can either be supplied specifically or the keyword “**any**” may be used to

express all ports of the specified protocol.

This command allows the target to communicate to the "outside" world by NATting its address to the specified address in the outbound network if and only if the given protocol and port matches.

Tnat is applied internally after the packets have been checked for load balancing.

Note: Another common used approach to offer outbound connectivity for the targets would be the declaration of a server reversely to offer services to the targets in the outbound direction.

Common uses are e.g. HTTP-requests from the targets to the "Internet" or DNS queries to a DNS server in a local network.

The Addresses of all tnat-entries are represented by all BalanceNG nodes with the same vrid (Virtual Router Identifier), existing outbound connections from targets via "tnat" will therefore survive any VRRP master switchover.

Different port/protocol tuples may be translated to the same outbound NAT address.

Example:

```
bng# tnat 10.1.1.1 192.168.1.230 tcp 22
bng# tnat 10.1.1.1 192.168.1.230 udp 53
bng# show vrrp
  state     MASTER
  vrid      1
  priority  255
  ip00      192.168.1.222
  ip01      192.168.1.230
  ip02      10.1.1.254
```

3.3.34 unregister

Synopsis: `unregister <target[s] | server[s]> <list>`

This is the counterpart command to the "`register`" command. Please see the more detailed state explanations there.

This command is used to transfer one or more targets or servers from the "registered" state to the "unregistered" state. Target and server parameters and configurations can only be changed in the "unregistered" state.

Example:

```
bng# unregister target 1
bng# unregister servers 1,2
```

3.3.35 vrrp

Synopsis: `vrrp <subcommand> <value>`

This command allows the configuration of the VRRP parameters. VRRP becomes activated as soon as all three VRRP parameters are defined (vrid, priority and network).

Using a "{" as a second argument opens a vrrp block which can be closed interactively by entering a "}" or an empty line.

Example:

```
bng# vrrp vrid 1
bng# vrrp priority 255
bng# vrrp network 3
bng# show vrrp
```

```
  state     MASTER
  vrid      1
  priority  255
  ip00      192.168.1.10
bng#
```

3.3.35.1 vrrp bscript

Synopsis: `vrrp bscript <script>`

This setting defines an external notification script or program which is called if the VRRP virtual router enters the BACKUP state. The script is executed in background by a separate helper thread. Setting the script parameter to nothing (“”) disables this setting.

Example:

```
NodeA# show vrrp
  state     MASTER
  vrid      14
  priority  200
  tracking  priority not degraded
  ipaddr0   172.17.2.64
NodeA# vrrp bscript "/home/tools/backup_notify"
NodeA# show vrrp
  state     MASTER
  vrid      14
  priority  200
  tracking  priority not degraded
  bscript   "/home/tools/vrrp_backup_notify"
  ipaddr0   172.17.2.64
NodeA#
```

3.3.35.2 vrrp mscript

Synopsis: `vrrp mscript <script>`

This setting defines an external MASTER state notification script accordingly to “vrrp bscript”. The script is called when BalanceNG enters the MASTER VRRP state. Setting the script parameter to nothing (“”) disables this setting again.

Example:

```
NodeA# show vrrp
  state     MASTER
  vrid      14
  priority  200
  tracking  priority not degraded
  bscript   "/home/tools/backup_notify"
  ipaddr0   172.17.2.64
NodeA# vrrp mscript "/home/tools/master_notify"
NodeA# show vrrp
  state     MASTER
  vrid      14
  priority  200
  tracking  priority not degraded
  mscript   "/home/tools/master_notify"
```

```
  bscript   "/home/tools/backup_notify"
  ipaddr0   172.17.2.64
NodeA#
```

3.3.35.3 vrrp network

Synopsis: `vrrp network <id>`

This defines the Network which will be used for VRRP communications. The following network parameters will be used for VRRP:

- VRRP advertisements will be sent out only on the interfaces associated to that network
- The VRRP primary address will be the "real" address of the network declaration.

Note: A network being referenced by the VRRP declaration using this command may be used as usual (there's no requirement to define an explicit VRRP only network).

Example:

```
bng# vrrp network 3
bng# show vrrp
  state     MASTER
  vrid      1
  priority  255
  ip00      192.168.1.10
bng#
```

3.3.35.4 vrrp priority

Synopsis: `vrrp priority <value>|none`

Sets the priority of the VRRP node to the specified value (or to unspecified if "none" is supplied). The priority must be in the range 1-255. The VRRP master priority is 255, the default backup node priority is 100.

Example1 (declare node to become VRRP master):

```
bng# vrrp vrid 1
bng# vrrp priority 255
bng# vrrp network 3
```

Example2 (declare node to be a backup router with the default backup priority of 100):

```
bng# vrrp vrid 1
bng# vrrp priority 100
bng# vrrp network 3
```

3.3.35.5 vrrp tracking

Synopsis:
```
vrrp tracking enable
vrrp tracking disable
vrrp tracking default
```

This command enables or disables VRRP tracking, respectively. VRRP tracking is disabled by default. If VRRP tracking is enabled and the VRRP priority is below 255 (which would be the so called "VRRP address owner") then the VRRP is degraded by the sum of all tracking values (see "`target <n> trackval`" and "`gateway trackval`") of all objects (targets or gateway) which are both enabled and down/inoperational.

With VRRP tracking a controlled failover to the backup BalanceNG node can be specified in case that the current master node loses connectivity to important objects which are still reachable and accessible by the backup node.

Example:

```
bng# vrrp priority 100
bng# vrrp network 1
bng# vrrp vrid 1
bng# show vrrp
  state     MASTER
  vrid      1
  priority  100
  tracking disabled (default)
  ip00      172.17.2.189
bng# vrrp tracking enable
bng# show vrrp
  state     MASTER
  vrid      1
  priority  100
  tracking enabled
  local priority not degraded
  ip00      172.17.2.189
bng#
```

3.3.35.6 vrrp vrid

Synopsis: `vrrp vrid <value>|none`

Specifies the Virtual Router Identifier of the Virtual Router of this node. All nodes in the same subnet sharing the same vrid represent all together the Virtual Router.

The supplied value must be in the range 1-255, supplying "none" sets the vrid to "unspecified" and such disables VRRP operation. See Reference /3/ for more Information about the VRRP protocol.

Example:

```
bng# vrrp vrid 1
bng# vrrp priority 255
bng# vrrp network 3
```

4 SNMP Support

BalanceNG supports SNMP by interfacing to the Net-SNMP server and comes with its own Set of Management Information Bases (MIBs).

BalanceNG supports read only SNMP access only, traps my be set out by calling snmptrap or snmp_trapsend from "alert" and "upalert" target configurations.

The BalanceNG MIB for instance 0 (as available for BalanceNG release 2) is located in the enterprise specific subtree of Inlab Networks GmbH at:
iso(1).org(3).dod(6).internet(1).private(4).enterprises(1).Inlab(2771).BalanceNG(1).

The instance specific MIBs for instances 0 to 128 are located in the enterprise specific subtree of Inlab Networks GmbH at:
iso(1).org(3).dod(6).internet(1).private(4).enterprises(1).Inlab(2771).BalanceNG(2).<Instance Number>.

All available MIBs are located in the "**MIBS**" directory in each distribution (either **/opt/BalanceNG/MIBS** oder relatively as **./MIBS** as in the tarball distribution).

4.1 Interfacing to Net-SNMP

An installed Net-SNMP software is required for interfacing to BalanceNG. This modern SNMP system is part of almost all current Linux systems.

Further information is available at the Net-SNMP website at this URL:

http://net-snmp.sourceforge.net/

The needed Debian and Ubuntu packages are snmpd and snmp (tiny-snmpd does not work for some reason).

The following readonly "com2sec" mapping is recommended (in /etc/snmp/snmpd.conf), just uncomment as follows:

```
#       sec.name  source          community
#com2sec paranoid  default         public
com2sec  readonly  default         public
#com2sec readwrite default         private
```

Additionally, the following line needs to be present in /etc/snmp/snmpd.conf in order to establish the interface between snmpd and BalanceNG standard instance 0 MIB:

```
pass .1.3.6.1.4.1.2771.1 /sbin/bng
```

If multiple instances need to be accessed by SNMP the multi-instance OID has to be specified additionally as follows:

```
pass .1.3.6.1.4.1.2771.2 /sbin/bng
```

Note: There's no need to change /etc/default/snmpd anymore (snmpd runs now as user snmp).

A typical "snmpget" command line looks like this (retrieving the "Release" string):

```
$ snmpget -v1 -c public localhost .1.3.6.1.4.1.2771.1.1
BALANCENG-MIB::Release = STRING: "3.176"
```

Retrieving the same "Release" string from BalanceNG instance 7 (for example) may be

invoked as follows:

```
$ snmpget -v1 -c public localhost .1.3.6.1.4.1.2771.1.1
BALANCENG-INSTANCE7-MIB::Release = STRING: "3.176"
```

A complete "snmpwalk" of the BalanceNG 2.x MIB can be invoked like this:

```
snmpwalk -v1 -c public localhost .1.3.6.1.4.1.2771.1
```

The BALANCENG-MIBs may be copied to the /usr/share/snmp/mibs directory (for Ubuntu/Debian Linux) like this:

```
# cd /opt/BalanceNG/MIBS
# cp *.txt /usr/share/snmp/mibs
```

The following environment variable setting makes the BalanceNG MIB available to the snmpd tools:

```
export MIBS=ALL
```

Note the difference between the two "snmpget" invocations below:

```
$ snmpget -v1 -c public localhost .1.3.6.1.4.1.2771.1.1
SNMPv2-SMI::enterprises.2771.1.1 = STRING: "2.228"
$ export MIBS=ALL
$ snmpget -v1 -c public localhost .1.3.6.1.4.1.2771.1.1
BALANCENG-MIB::Release = STRING: "2.228"
```

You may also setup a local snmp.conf like this to make this setting permanent:

```
$ mkdir -p $HOME/.snmp
$ echo "mibs ALL" >> $HOME/.snmp/snmp.conf
```

4.2 Accessing the SNMP interface directly

The following command line options are used to access the BalanceNG MIB objects:

-g <oid>	GET the specified OID
-n <oid>	GET the next OID starting from the specified OID
-s <oid> <value>	SET the OID to the specified value (not supported with bng)

Example:

The following simple shell script retrieves the current state of the VRRP using that interface:

```
#!/bin/sh
VALUE=`/usr/bin/bng -g .1.3.6.1.4.1.2771.1.21 | tail -1`
case $VALUE in
  0) STATE="OFF" ;;
  1) STATE="INITIALIZE" ;;
  2) STATE="MASTER" ;;
  3) STATE="BACKUP" ;;
  *) STATE="UNKNOWN" ;;
esac
```

```
echo "VRRP state is $STATE"
```

4.3 Testing with snmpget and snmpwalk

The installation may be tested by issuing snmpget and snmpwalk commands on a client machine (which my be the BalanceNG node itself):

Example:

To return the number of current sessions from BalanceNG node "castor" you may enter the following:

```
# snmpget -v1 -c public castor BALANCENG-MIB::Sessions
BALANCENG-MIB::Sessions = Gauge32: 1128
#
```

There are 1128 current sessions active.

A walk over instance 0 using the BalanceNG V2 OID may be invoked like this:

```
$ snmpwalk -v1 -c public localhost .1.3.6.1.4.1.2771.1
```

A “walk” over all active (running) instances using the newer multi-instance OID may be invoked like this:

```
$ snmpwalk -v1 -c public localhost .1.3.6.1.4.1.2771.2
```

4.4 MRTG relevant metrics

The following metrics would be relevant for collecting with MRTG:

Sessions (Gauge)
1.3.6.1.4.1.2771.1.9
Number of current total session table entries

InterfaceSentPackets
1.3.6.1.4.1.2771.1.40.5.X
Number of packets sent out on this interface

InterfaceSentBytes
1.3.6.1.4.1.2771.1.40.6.X
Number of bytes sent out on this interface

InterfaceReceivedPackets
1.3.6.1.4.1.2771.1.40.7.X
Number of packets received on this interface

InterfaceReceivedBytes
1.3.6.1.4.1.2771.1.40.8.X
Number of bytes received on this interface

ServerSessions (Gauge)
1.3.6.1.4.1.2771.1.60.13.X
Current number of virtual server sessions, defined to be
the sum of the sessions of all associated targets

ServerSentPackets

1.3.6.1.4.1.2771.1.60.14.X
Number of packets sent to the clients, defined to be the sum of packets being received from all associated targets

ServerSentBytes
1.3.6.1.4.1.2771.1.60.15 X
Number of bytes sent to the clients, defined to be the sum of bytes being received from all associated targets

ServerReceivedPackets
1.3.6.1.4.1.2771.1.60.16 X
Number of packets received from the clients by this virtual server, defined to be the sum of packets being sent to all associated targets

ServerReceivedBytes
1.3.6.1.4.1.2771.1.60.17.X
Number of bytes received from the clients by this virtual server, defined to be the sum of bytes being sent to all associated targets

TargetSessions (Gauge)
1.3.6.1.4.1.2771.1.70.25.X
Current number of target (real server) sessions

TargetSentPackets
1.3.6.1.4.1.2771.1.70.26.X
Number of packets sent to target (real server)

TargetSentBytes
1.3.6.1.4.1.2771.1.70.27.X
Number of bytes received from target (real server)

TargetReceivedPackets
1.3.6.1.4.1.2771.1.70.28.X
Number of packets received from target (real server)

TargetReceivedBytes
1.3.6.1.4.1.2771.1.70.29.X
Number of bytes received from target (real server)

TargetAgentData (Gauge)
1.3.6.1.4.1.2771.1.70.30.X
Performance data as returned from bngagent feedback agent

TargetTotalBandwidth (Gauge)
1.3.6.1.4.1.2771.1.70.31.X
Current total bandwidth of target in bytes per second

TargetIncomingBandwidth (Gauge)
1.3.6.1.4.1.2771.1.70.32.X
Current incoming bandwidth of target in bytes per second

TargetOutgoingBandwidth (Gauge)
1.3.6.1.4.1.2771.1.70.33.X
Current outgoing bandwidth of target in bytes per second

5 Logging

BalanceNG uses the `syslog` interface to send logging message to the Operating System syslog facility. It uses the identification "BalanceNG" to `openlog()`. Logging may be collected centrally by configuring `/etc/syslog.conf` appropriately and by maintaining a central log server machine.

The last 20 syslog messages are being collected in a cyclic buffer and can be investigated using the "`show log`" command.

BalanceNG uses a set of syslog messages to report about "normal, but significant conditions" and uses the syslog level `LOG_NOTICE` exclusively for that purpose.

The following messages may be logged that way:

`this virtual router is now MASTER`

The node participates in VRRP and has just become the MASTER VRRP router with the configured VRID (Virtual Router Identifier).

`this virtual router is now BACKUP`

The node participates in VRRP and has been superseded my a higher priority node. It has entered BACKUP state.

`vrrp off and in state INITIALIZE`

VRRP has been administratively switched off.

`target <number> operational`

The target with the specified index has just become operational because all configured health checks succeed. An "upalert" script is called (if defined for that target).

`target <number> down`

The target with the specified index has become inoperational (down). This can be either caused by failing health checks or by taking the target administratively out of "enabled" state (with the "`disable`" command). An "alert" script is called (if defined for that target).

6 Bngagent

Bngagent is a small UDP server program which runs on a UNIX target machine. Using a simple UDP protocol (the Bngagent Protocol) the "agent" health check method of BalanceNG is capable to talk to this agent.

The source code of Bngagent is made available to the customers on order to enable them to port is to the various UNIX based machines that they want to monitor.

Precompiled binaries of Bngagent are made available for Linux/x86 and macOS.

6.1 Compiling Bngagent

Recommended example compilation command lines for compiling bngagent are:

Linux: `gcc -o bngagent bngagent.c`

6.2 Starting and Stopping of Bngagent

The usage information of Bngagent (when called with no argument) is as follows:

```
# bngagent

  $Revision: 3.9 $
  bngagent is an open source part of the BalanceNG product
  Copyright (C) 2005-2009,2010 by Inlab Networks GmbH, Gruenwald, Germany
  All rights reserved - more infos at: http://www.BalanceNG.net

  usage:
    server            bngagent <options>    port
    request (test)    bngagent <options> -r address port
  options:
    -0                return 1 minute load avg    (server,default)
    -1                return 5 minute load avg    (server)
    -2                return 15 minute load avg   (server)
    -6                use IPv6 instead of IPv4
    -f                stay in foreground          (server)
    -b <address>      specify bind address        (both)
    -c <command>      specify command             (server)
    -d                enable debug and foreground (both)
    -t <targetid>     specify targetid            (request)
#
```

Bngagent with the port number as argument puts itself into background and starts listening for UDP packets being sent out by BalanceNG:

```
# bngagent 439
#
```

Starting bngagent in IPv6 mode works like this:

```
# bngagent -6 439
#
```

You may choose any available UDP port for that, if you want to use a privileged port below 1024 please use port 439 which is allowed to be used by the author for that purpose.

Stopping might be performed by using the "`pkill`" tool, e.g.:

```
# pkill bngagent
```

```
#
```

The option `-f` forces Bngagent to stay in foreground, the option `-d` enables debugging output and does an implicit `-f`.

Bngagent with no `-c` Argument calls `getloadavg()` and returns the 1 minute load average of the machine multiplied by 100 back to BalanceNG for further processing (per default). Option `-1` makes bngagent return the 5 minute load average and the option `-2` makes it return the 15 minute load average.

Thus BalanceNG is immediately capable to take the system load of the target machines into account.

You may pass control to a external script using the `-c` option. If an BalanceNG request is being received by the Bngagent calls the external script and expects one line on stdout from that script (at minimum). This value has to be in the unsigned integer range from 0 to 2^32.

Returning 0 means for BalanceNG that the target's service has become totally unavailable and will force the target to become inoperational immediately. Otherwise BalanceNG with the "agent" method will choose the target with the lowest Bngagent return value.

You may test a running bngagent instance by invoking another bngagent with the -r option (request mode). This is especially useful during the development of specific bngagent scripts. An example might look as follows:

```
# ./bngagent -c 'pgrep sshd | wc -l' 5000
# ./bngagent -r localhost:5000
target id: 0 value: 1
# ./bngagent -r localhost:5001
*timeout*
#
```

Compiling Bngagent can be done by a simple command.

For Linux you could use:

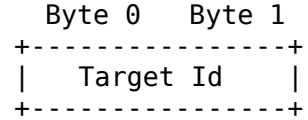
```
$ gcc -o bngagent bngagent.c
$
```

We started to add precompiled binaries of bngagent to the main distribution. Please take a look into the "`bngagent-binaries`" directory.

6.3 The Bngagent UDP Protocol

6.3.1 The Bngagent Protocol Request

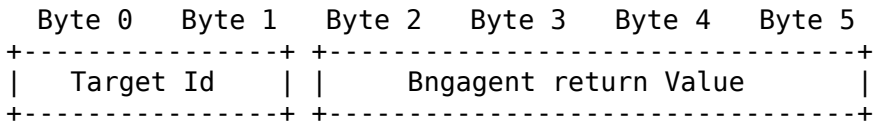
The Bngagent request packet simply consists of an unsigned short in network order which represents the target id (or target number) for which information is being requested. The total data length encapsulated in UDP is two bytes:



There's no authentication provided.

6.3.2 The Bngagent Protocol Reply

The reply sends the received Target Id back in a two byte unsigned short and additionally returns the return value as a unsigned integer in four bytes. All data is in network order. The total length of the Bngagent reply packet encapsulated in UDP is 6 bytes.



BalanceNG will choose the target with the lowest Bngagent return value when a new session has to be created. Return 0 as the Bngagent return value means that the target is currently unavailable (e.g. the service requested is down).

Sending back the Target Id allows multiple Bngagents to run on one host for different targets. BalanceNG is that way capable to assign the replies based to their respective targets.

6.4 Writing Bngagent Scripts

A script or program being called by bngagent with the "-c" option should just put out one line on stdout and then exit immediately. The line should contain one number either "0" for "service unavailable" or any other positive integer in the 32 bit unsigned int range.

As mentioned before the BalanceNG agent distribution method will 1) disable any target where the agent reports "0" and 2) will direct new sessions to the target with the lowest Bngagent return value.

One very simple two line example is to count the number of processes:

```
#!/bin/sh
pgrep httpd | wc -l
```

More complicated health checks should be implemented as scripts or programs to be handled by Bngagent.

6.5 Bngagent Source Code

The source code of bngagent is part of the BalanceNG distribution. It is being distributed under the BalanceNG license.

7 Technical Background Information

7.1 BalanceNG MAC Addresses

BalanceNG allocates non-VRRP MAC addresses within the IEEE assigned “Mac Address Block Large” (MA-L) of Inlab Networks GmbH, beginning with 34-38-AF. The lower 3 bytes are derived from the BalanceNG nodeid and the instance number (0-127).

For VRRP the standard MAC addresses are used (VRRP v2 and v3).

7.2 Session Table and NAT State Synchronization

BalanceNG may either use Non-Standard VRRP Extension Packets or the bngsync UDP protocol for synchronizing the session table and NAT states between the VRRP Master and the VRRP Backup nodes.

7.2.1 The bngsync UDP Protocol (UDP port 10439)

The bngsync UDP protocol may use either IPv4 or IPv6 as transport protocol and uses the registered UDP port 10439 (bngsync).

There are currently two defined message types (0x00 for “state synchronization”, 0x01 for “state synchronization” protocol testing messages) and there are currently 5 different message subtypes below these message types. These message subtypes range from 4 to 8 reflecting the same non-standard VRRP message type that they replace, respectively.

Testing messages with message type 0x01 have exactly the same format as message type 0x00 and are used for automatic and manual protocol testing.

Since the bngsync protocol usually runs in parallel to VRRP (replacing synchronization with the Non-Standard VRRP extensions), it requires to include the VRRP virtual router ID and the VRRP router priority in each message.

The general bngsync UDP protocol message format is as follows:

Byte Offset	Description	Value
0	bngsync protocol tag	0xFF
1	bngsync protocol version 1	0x01
2	virtual router ID (VRID)	variable
3	virtual router priority	variable
4	message type	variable
5	message subtype	variable
6	data length high byte	variable
7	data length low byte	variable
8	data start	
.		
.		
N	last data byte	

7.2.1.1 Type 0 Subtype 4: Session Table Sync Advertisement

With this message type the current VRRP master notifies the backup node (or more than one backup node) that a certain session table entry has just been created or that this session table entry is still valid and in use.

The session table entry key is a zero terminated and readable ASCII string. The last byte of this message is the terminating 0x00, the data length at byte offsets 6 and 7 is therefore 2 + 2 + 4 + strlen(key) + 1.

Byte Offset	Description	Value
0	bngsync protocol tag	0xFF
1	bngsync protocol version 1	0x01
2	virtual router ID (VRID)	variable
3	virtual router priority	variable
4	message type	0x00
5	message subtype	0x04
6	data length high byte	variable
7	data length low byte	variable

8+9	server number unsigned short network byte order	
10+11	target number unsigned short network byte order	
12-15	session timeout unsigned integer network byte order	
16	ASCII readable session key	
.		
.		
N	last data byte	0x00

7.2.1.2 Type 0 Subtype 5: GNAT State Sync Advertisement

With this message type the current VRRP master notifies the backup node (or more than one backup node) that a certain GNAT (generic NAT) table entry has just been created or that this GNAT table entry is still valid and in use.

The proto_left and proto_right unsigned short field contents are defined as follows:

Value	Description	
1	GNAT_PROTO_TCP4	IPv4 NAT of a TCP connection
2	GNAT_PROTO_UDP4	IPv4 NAT of a UDP “connection”
3	GNAT_PROTO_PING4	IPv4 NAT of ICMP ECHO/REPLY
4	GNAT_PROTO_TCP6	IPv6 NAT of a TCP connection
5	GNAT_PROTO_UDP6	IPv6 NAT of a UDP “connection”
6	GNAT_PROTO_PING6	IPv6 NAT of ICMP ECHO/REPLY

The proto_left and the proto_right values must have the same values in the same message packet (there's no IPv4 to IPv6 or IPv6 to IPv4 conversion implemented with the GNAT functionality).

IPv4 addresses in the fields addr_left and addr_rightare represented in IPv6 “mapped” format (::FFFF:a.b.c.d).

If the packet is related to GNAT_PROTO_PING4 or GNAT_PROTO_PING6, the port_left and port_right fields are containing the ICMP identifier (instead of a port).

The “left” side of the GNAT table entry corresponds to the “nat inside” network, the “right” side to the “nat outside” network in the BalanceNG configuration.

The data length of this message type is always 40 bytes (16+2+2+16+2+2).

Byte Offset	Description	Value
0	bngsync protocol tag	0xFF
1	bngsync protocol version 1	0x01
2	virtual router ID (VRID)	variable
3	virtual router priority	variable
4	message type	0x00

5	message subtype	0x05
6	data length high byte	variable
7	data length low byte	variable
8-23	16 bytes addr_left IP address IPv6-notation	
24-25	proto_left unsigned short network byte order	
26-27	port_left unsigned short network byte order	
28-43	16 bytes addr_right IP address IPv6-notation	
44-45	proto_right unsigned short network byte order	
46-47	port_right unsigned short network byte order	

7.2.1.3 Type 0 Subtype 6: Session Table Sync ACK

A backup node acknowledges with this message the receipt of a specific session table sync advertisement (type 0 subtype 4). The packet has exactly the same contents as the matching advertisement, except that the message subtype is 6.

This message type is only sent by the current VRRP backup node if “session table synchronization acknowledgement” is activated by setting the parameter sessionsyncack to 1 (active).

Byte Offset	**Description**	**Value**
0	bngsync protocol tag	0xFF
1	bngsync protocol version 1	0x01
2	virtual router ID (VRID)	variable
3	virtual router priority	variable
4	message type	0x00
5	message subtype	0x06
6	data length high byte	variable
7	data length low byte	variable
8+9	server number unsigned short network byte order	
10+11	target number unsigned short	

	network byte order	
12-15	session timeout unsigned integer network byte order	
16	ASCII readable session key	
.		
.		
N	last data byte	0x00

7.2.1.4 Type 0 Subtype 7: Session Table Resync Request

This message type a backup node requests the resynchronization of the complete session table after he has been restarted and remained in BACKUP state for 10 seconds. The data length is 0 (byte offsets 6 and 7).

The parameters sessionsync and sessionautoresync need both to be activated (set to 1). The BACKUP retries sessionarrtimeout seconds (default 60) until it receives a Session Table Resync Request ACK from the master.

Byte Offset	Description	Value
0	bngsync protocol tag	0xFF
1	bngsync protocol version 1	0x01
2	virtual router ID (VRID)	variable
3	virtual router priority	variable
4	message type	0x00
5	message subtype	0x07
6	data length high byte	0
7	data length low byte	0

7.2.1.5 Type 0 Subtype 8: Session Table Resync Request ACK

This message type is the reply of the VRRP MASTER node in response to a “Session Table Resync Request”. The data length is also 0 (byte offsets 6 and 7).

The parameters sessionsync and sessionautoresync also need both to be activated (set to 1).

Byte Offset	Description	Value
0	bngsync protocol tag	0xFF
1	bngsync protocol version 1	0x01
2	virtual router ID (VRID)	variable
3	virtual router priority	variable
4	message type	0x00
5	message subtype	0x08
6	data length high byte	0
7	data length low byte	0

7.2.2 Non-Standard VRRP Extensions

7.2.2.1 Type 2: BalanceNG V2 Session Table Sync Advertisement

BalanceNG V2 uses a VRRP extension by using a VRRP packet type 2, which is unknown and undefined by the VRRP version 2 standard.

Note: This packet type has been used with BalanceNG V2 to synchronize the session table and is no longer in use with BalanceNG V3. The BNG V3 session table synchronization is handled by VRRP extension packet types 4 and 6.

The session table information is packed into the space of 4 IP addresses which allows easy reading with network analyzers. The packing of session table information is as follows:

```
IP address 0:  Session source IP address
IP address 1:  Session source port in the lower 2 octets (not a real IP address)
IP address 2:  Server number in the higher 2 octets,
               Target number in the lower 2 octets
IP address 3:  Server specific session timeout (0 if not set).
```

Here an example of a session announcement as it could appear in the Wireshark packet analyzer (Session 172.17.2.4 port any -> 172.17.2.189 port 22; Target 1):

```
Frame 19 (70 bytes on wire, 70 bytes captured)
Ethernet II, Src: IETF-VRRP-virtual-router-VRID_0a (00:00:5e:00:01:0a),
             Dst: 01:00:5e:00:00:12 (01:00:5e:00:00:12)
Internet Protocol, Src: 172.17.2.188 (172.17.2.188), Dst: 224.0.0.18 (224.0.0.18)
Virtual Router Redundancy Protocol
    Version 2, Packet type 2 (Unknown)
    Virtual Rtr ID: 10
    Priority: 255 (This VRRP router owns the virtual router's IP address(es))
    Count IP Addrs: 5
    Auth Type: No Authentication (0)
    Adver Int: 1
    Checksum: 0x81f3 [correct]
    IP Address: 172.17.2.4 (172.17.2.4)
    IP Address: 0.0.0.0 (0.0.0.0)
    IP Address: 0.1.0.1 (0.1.0.1)
    IP Address: 0.0.0.0 (0.0.0.0)
    IP Address: 0.0.0.0 (0.0.0.0)
```

Session table information is being broadcasted by the current active master node only for active sessions and controlled by the “sessionsynciv” parameter to keep the additional traffic low.

7.2.2.2 Type 3: BalanceNG V2 NAT State Sync Advertisement

7.2.2.3 Type 4: BalanceNG V3 Session Table Sync Advertisement

7.2.2.4 Type 5: BalanceNG V3 GNAT State Sync Advertisement

7.2.2.5 Type 6: BalanceNG V3 Session Table Sync ACK

7.2.2.6 Type 7: BalanceNG V3 Session Table Resync Request

7.2.2.7 Type 8: BalanceNG V3 Session Table Resync Request ACK

8 Third Party Software Copyright Notices

8.1 LDNS (DNS Library)

BalanceNG V3 may be statically linked to the “LDNS” DNS library (available at the following URL: http://www.nlnetlabs.nl/ldns/). The “LDNS” DNS library copyright notice is as follows:

```
Copyright (c) 2005,2006, NLnetLabs
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

    * Redistributions of source code must retain the above copyright notice,
      this list of conditions and the following disclaimer.
    * Redistributions in binary form must reproduce the above copyright
      notice, this list of conditions and the following disclaimer in the
      documentation and/or other materials provided with the distribution.
    * Neither the name of NLnetLabs nor the names of its
      contributors may be used to endorse or promote products derived from this
      software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.
```

9 References

/1/ Radia Perlman: Interconnections Second Edition
Addison Wesley, ISBN 0201634481

/2/ Rich Seifert: The Switch Book
Wiley & Sons, ISBN 0471345865

/3/ R. Hinden et. al.: Virtual Router Redundancy Protocol (VRRP)
RFC3768