NFWare vCGNAT

Release 6.2.0

NFWare Authors

Mar 13, 2024

CONTENTS

CHAPTER

ONE

INSTALLATION

1.1 Virtual Machine Installation

1.1.1 Minimum Requirements

A server with an operating system installed and a KVM hypervisor are required for the virtual machine installation.

Recommended OS:

- CentOS/RHEL/Oracle Linux 7
- RHEL/Oracle Linux 8
- Ubuntu 20.04

Minimum system requirements for the Virtual Machine:

- 2 processor cores
- 6GB RAM
- 2 network interfaces (management and user traffic)

List of supported NICs:

- 10GbE – Intel X520 / Intel X710 / Intel E810 / Mellanox ConnectX-6
- 25GbE – Intel X710 / Intel E810 / Mellanox ConnectX-5 / Mellanox ConnectX-6
- 40GbE – Intel XL710 / Mellanox ConnectX-6
- 50GbE – Intel E810 / Mellanox ConnectX-6
- 100GbE – Intel E810 / Mellanox ConnectX-5 / Mellanox ConnectX-6
- 200GbE – Mellanox ConnectX-6

NAT also supports virtual network adapters `e1000`, `e1000e`, `virtio`, and `vmxnet3`. However, their usage is not recommended in commercial environments due to their performance limitations.

1.1.2 Processor Requirements

The processor must support the AVX2 (Advanced Vector Extensions 2) instruction set to run the Virtual Machine. All Intel Xeon processors starting from the Haswell (v3) family support this instruction set.

To check the processor model, `lscpu` utility can be used:

```
$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                56
On-line CPU(s) list:   0-55
Thread(s) per core:    2
Core(s) per socket:    14
Socket(s):             2
NUMA node(s):          1
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 79
Model name:            Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz
Stepping:              1
CPU MHz:               2899.951
CPU max MHz:           3300.0000
CPU min MHz:           1200.0000
BogoMIPS:              4788.95
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              35840K
NUMA node0 CPU(s):     0-55
Flags:                 fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat␣
↪pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm␣
↪constant_tsc arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc aperfmperf␣
↪eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 sdbg fma cx16 xtpr␣
↪pdcm pcid dca sse4_1 sse4_2 x2apic movbe popcnt aes xsave avx f16c rdrand lahf_lm abm␣
↪3dnowprefetch epb cat_l3 cdp_l3 invpcid_single intel_ppin intel_pt tpr_shadow vnmi␣
↪flexpriority ept vpid fsgsbase tsc_adjust bmi1 hle avx2 smep bmi2 erms invpcid rtm cqm␣
↪rdt_a rdseed adx smap xsaveopt cqm_llc cqm_occup_llc cqm_mbm_total cqm_mbm_local␣
↪dtherm ida arat pln pts
```

The `avx2` flag must be present in the `Flags` line. To easily check the presence of the flag, the following command can be used:

```
$ lscpu | grep avx2
```

If the command doesn't display anything, it means that your processor doesn't support the AVX2 instruction set, and you can not run the Virtual Machine on this processor.

1.1.3 Memory Requirements

Here you can see the requirenents for memory vs sessions if you have 1 or 2 CPU. This memory recommendation is for VM, assuming that some memory has been reserved for Host OS previously.

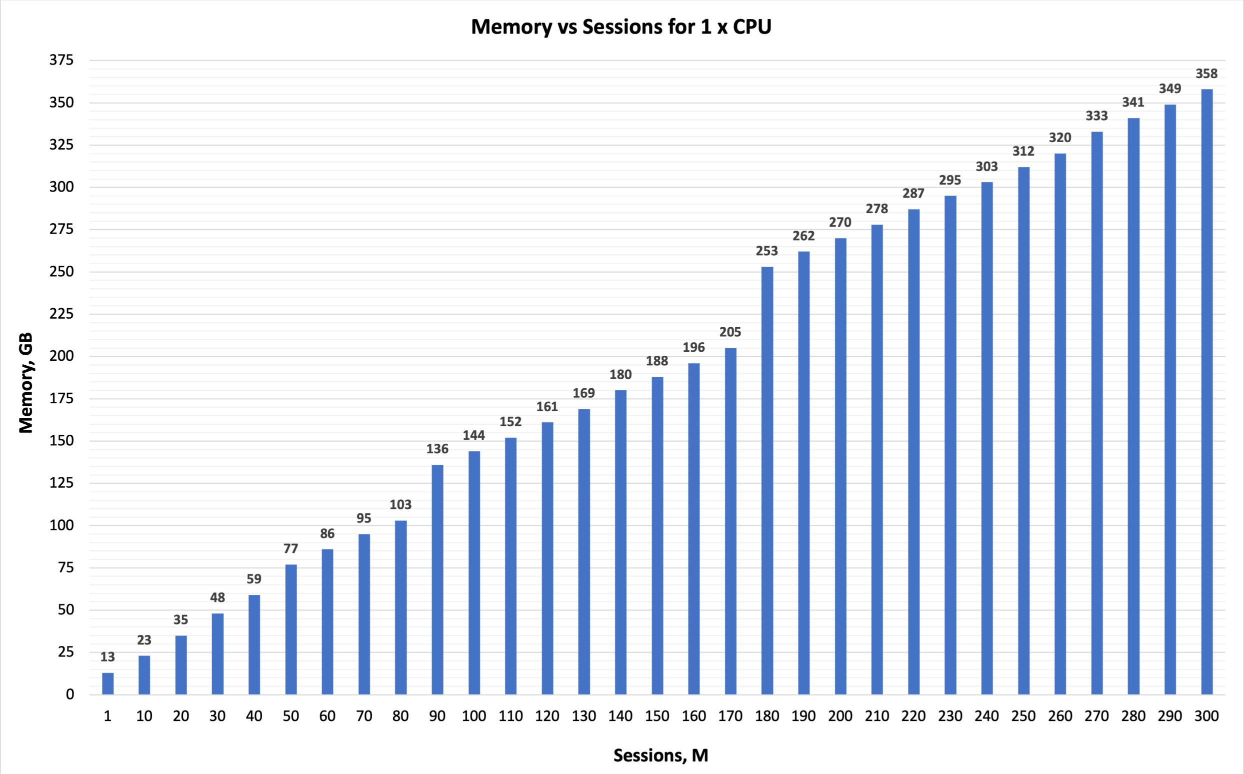


Fig. 1: Figure: Memory(GB) vs Session(M) for 1xCPU

1.1.4 Installation Steps

Before following the steps described in this chapter, make sure that the Linux minimal package is installed as the Host OS. Make sure that **Turbo Boost** mode has been disabled in the BIOS.

1.1.4.1 Network Configuration

Configure the Host OS network

The NFWare VM has two interface types.

The first type is the management interface used for OAM. It does not require high throughput and can work via a Host OS bridge. The second type is Data Plane interfaces. Data Plane interfaces are passed through PCI directly from a server to the NAT virtual machine.

You need to create the bridge and place the interface used for management in that bridge before NFWare VM starts. Example of bridge interface creation:

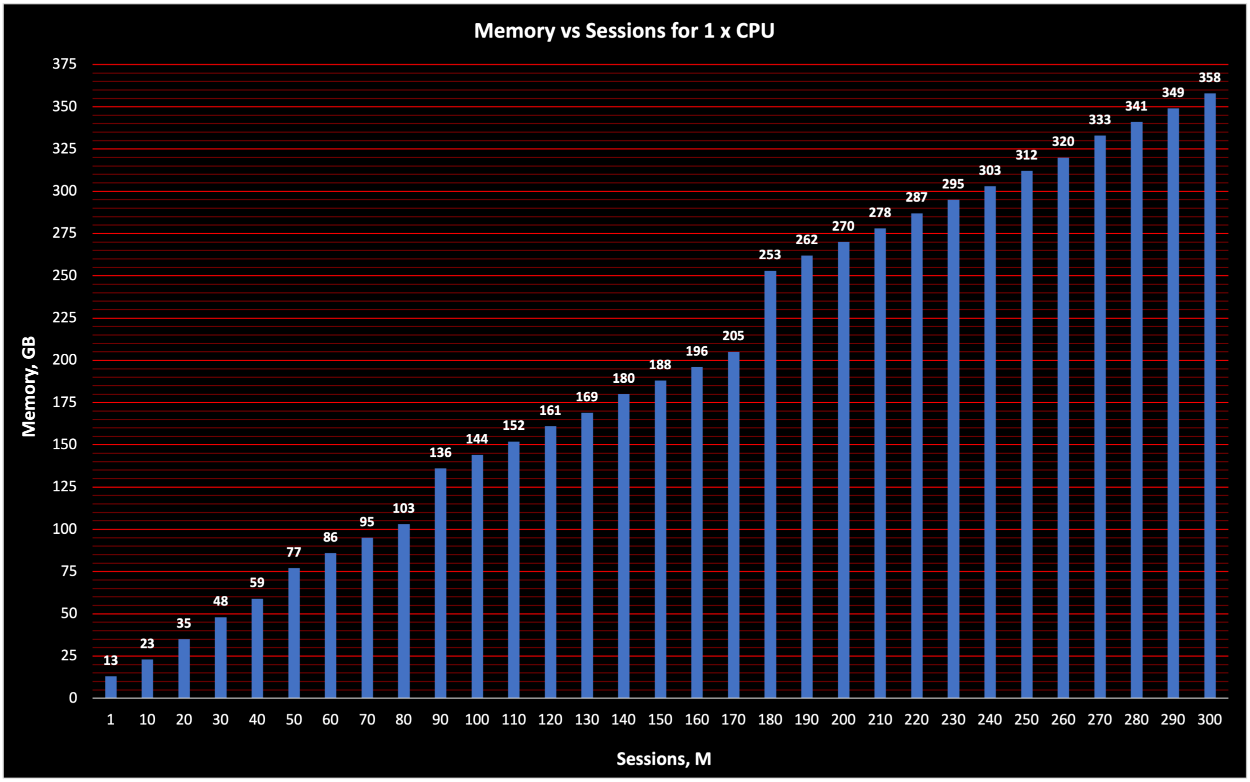


Fig. 2: Figure: Memory(GB) vs Session(M) for 1xCPU

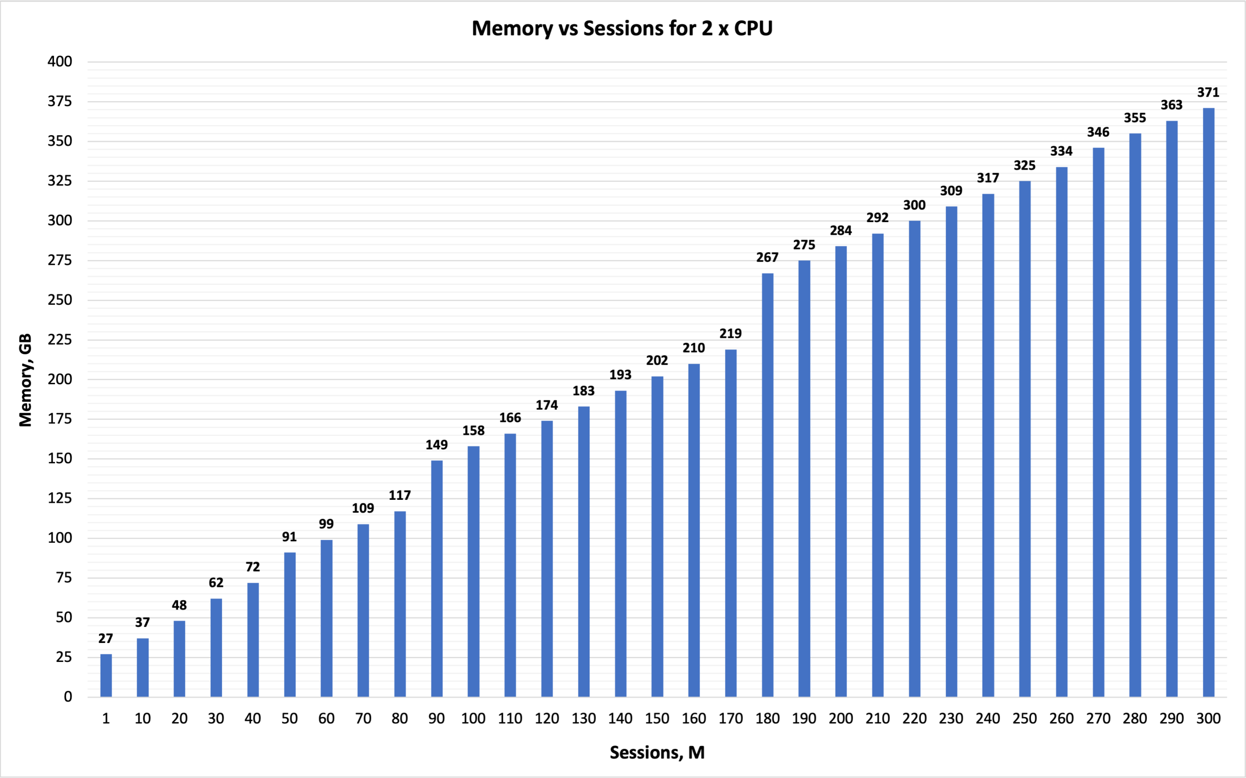


Fig. 3: Figure: Memory(GB) vs Session(M) for 2xCPU

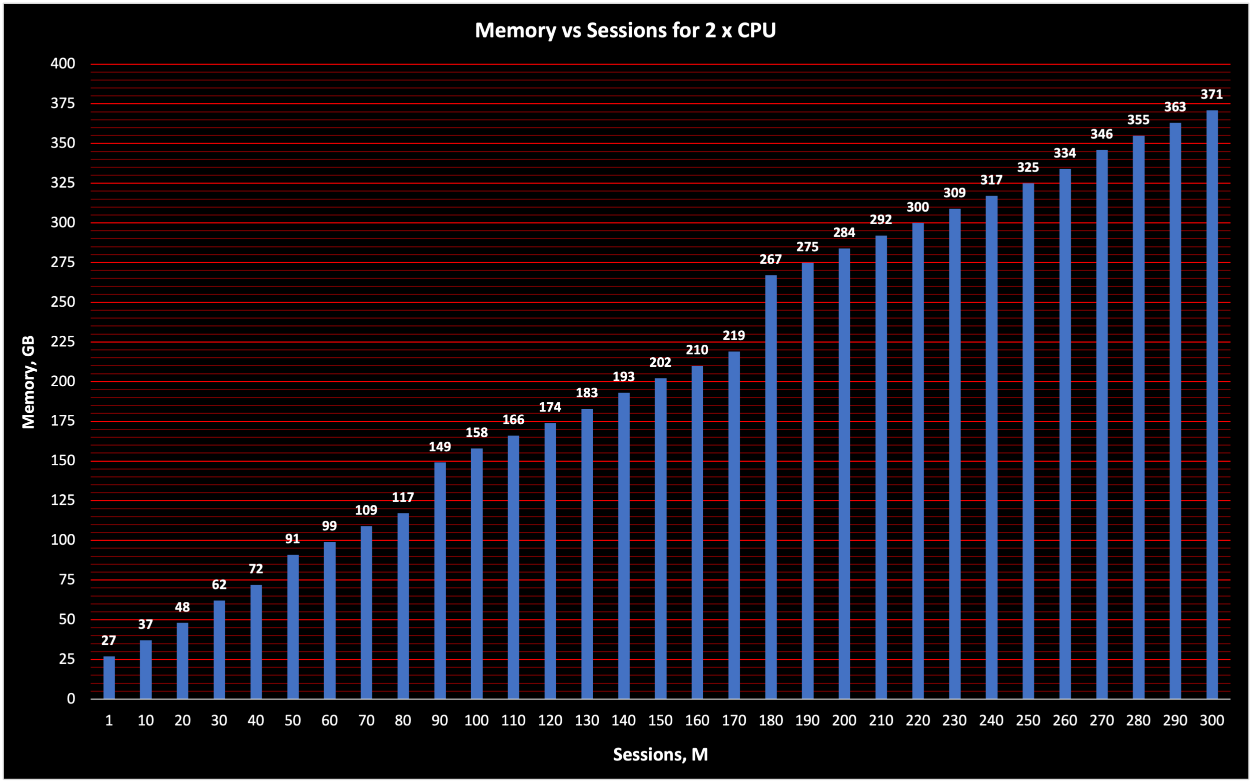


Fig. 4: Figure: Memory(GB) vs Session(M) for 2xCPU

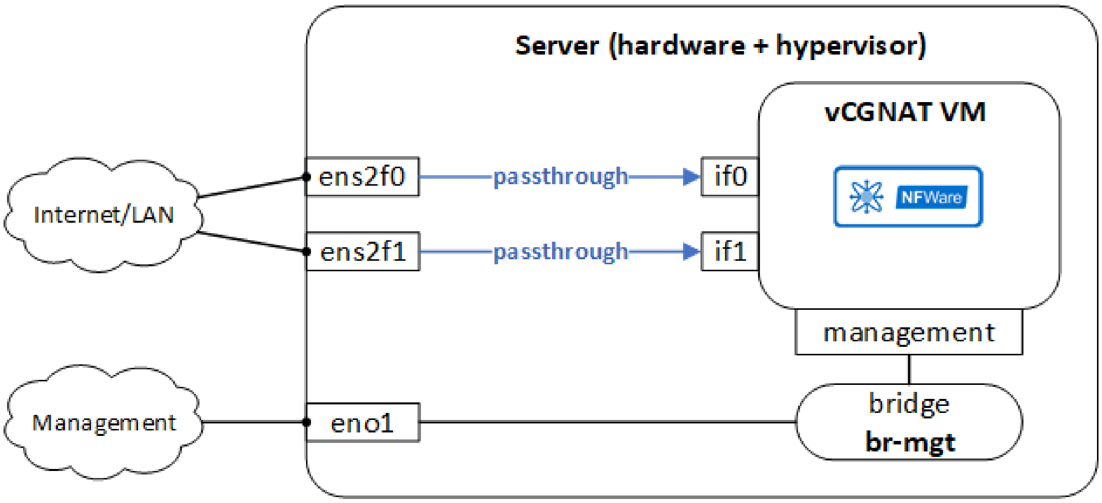


Fig. 5: **Figure 1. Network configuration**

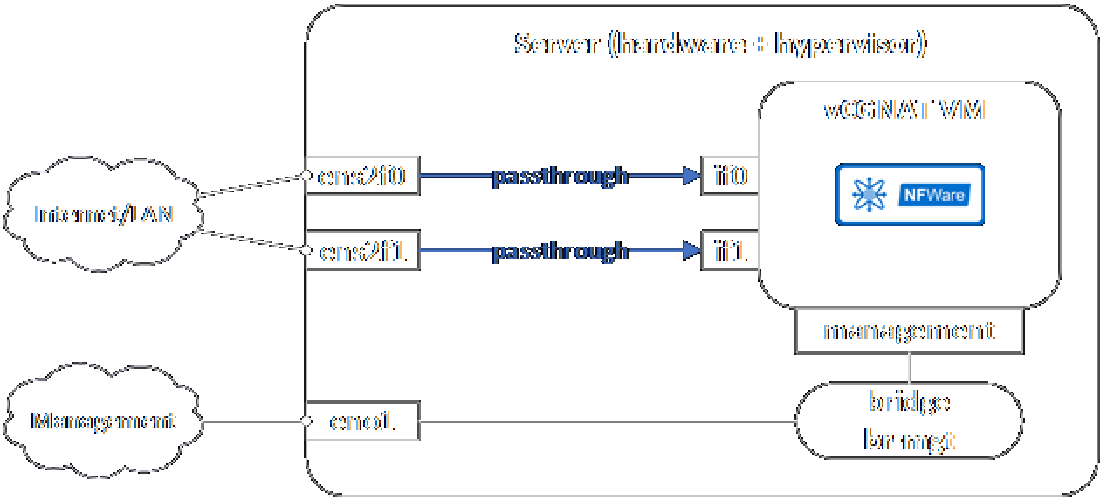


Fig. 6: **Figure 1. Network configuration**

br-mgt – the name of a bridge device

enp3s0f0 – the name of the NIC used to connect to a management network

- In CentOS/Oracle Linux, the Network Manager utility can be used:

```
nmcli connection add type bridge ifname br-mgt bridge.stp no
nmcli con add type ethernet con-name br-mgt-slave-1 ifname enp3s0f0 master br-mgt
nmcli con modify bridge-br-mgt ipv4.method auto
nmcli con up bridge-br-mgt
nmcli con up br-mgt-slave-1
```

- In Ubuntu, the Netplan utility can be used.

  Open and edit the network configuration file *yaml under `/etc/netplan/`.

```
Network:
 ethernets:
   enp3s0f0:
     dhcp4: false
 version: 2
 bridges:
   br-mgt:
     dhcp4: true
     interfaces:
         - enp3s0f0
```

  To apply the settings, use this command:

```
netplan apply
```

1.1.4.2 Preparation

Install QEMU hypervisor, dependencies, and copy vCGNAT VM installation script.

CentOS 7

1. Install libvirt with dependencies:

   ```
   yum install libvirt
   ```

2. Download the vCGNAT VM image to the following directories:

   2.1. vCGNAT VM qcow2 image to the `/var/lib/libvirt/images/` directory

   2.2. vCGNAT VM installation script to the `/var/lib/libvirt/images/hw-configuration/` directory

3. Install KVM hypervisor from a repository. The KVM hypervisor version must be 4.2 or more recent.

   ```
   yum install centos-release-qemu-ev
   yum install qemu-kvm-ev
   ```

4. Install the following packages and dependencies from a repository:

   - xorriso
   - libosinfo
   - xz-devel
   - python3
   - zlib-devel
   - wget
   - pciutils
   - msr-tools

   ```
   yum install xorriso libosinfo xz-devel python3 zlib-devel pciutils msr-tools
   ```

5. Install the required packages and dependencies from a local directory:

   - virt-install
   - python

   ```
   cd /var/lib/libvirt/images/hw-configuration/virt-install/CentOS-7
   yum localinstall *
   ```

6. After installing the KVM hypervisor and packages, reboot the machine:

   ```
   reboot
   ```

CentOS 8

1. Install libvirt with dependencies:

```
yum install libvirt
```

2. Download the vCGNAT VM image to the following directories:

   2.1. vCGNAT VM qcow2 image to `/var/lib/libvirt/images/` directory

   2.2 vCGNAT VM installation script to `/var/lib/libvirt/images/hw-configuration/` directory

3. Install KVM hypervisor from a repository. The KVM hypervisor version must be 4.2 or more recent.

```
yum install qemu-kvm
```

4. Install the following packages and dependencies from a repository:

   - msr-tools

```
yum install oracle-epel-release-el8.x86_64
yum install msr-tools
```

5. Install the required packages and dependencies from a local directory:

   - virt-install

```
cd /var/lib/libvirt/images/hw-configuration/virt-install/CentOS-8
yum localinstall *
```

6. After installing the KVM hypervisor and packages, reboot the machine:

```
reboot
```

Oracle Linux 7

1. Install libvirt with dependencies:

```
yum install libvirt
```

2. Download the vCGNAT VM image to the following directories:

   2.1. vCGNAT VM qcow2 image to `/var/lib/libvirt/images/` directory

   2.2. vCGNAT VM installation script to `/var/lib/libvirt/images/hw-configuration/` directory

3. Install KVM hypervisor, required packages, and dependencies from a repository. The KVM hypervisor version must be 4.2 or more recent.

```
yum install qemu-kvm
yum install libpmem
yum --disablerepo="*" --enablerepo="ol7_kvm_utils" update
```

4. Install the following packages and dependencies from a repository:

   - msr-tools

```
yum install msr-tools
```

5. Install the required packages and dependencies from a local directory:

   - virt-install
   - python

```
cd /var/lib/libvirt/images/hw-configuration/virt-install/Oracle-Linux-8
yum localinstall *
```

6. After installing the KVM hypervisor and packages, reboot the machine:

```
reboot
```

Oracle Linux 8

1. Install libvirt with dependencies:

```
yum install libvirt
```

2. Download the vCGNAT VM image to the following directories:

   2.1. vCGNAT VM qcow2 image to `/var/lib/libvirt/images/` directory

   2.2. vCGNAT VM installation script to `/var/lib/libvirt/images/hw-configuration/` directory

3. Install KVM hypervisor from a repository. The KVM hypervisor version must be 4.2 or more recent.

```
yum install qemu-kvm
```

4. Install the following packages and dependencies from a repository:

   - msr-tools

```
yum install oracle-epel-release-el8.x86_64
yum install msr-tools
```

5. Install the required packages and dependencies from a local directory:

   - virt-install

```
cd /var/lib/libvirt/images/hw-configuration/virt-install/Oracle-Linux-8
yum localinstall *
```

6. After installing the KVM hypervisor and packages, reboot the machine:

```
reboot
```

Ubuntu 20.04

1. Install libvirt with dependencies:

   ```
   sudo apt install libvirt-daemon-system
   ```

2. Download the vCGNAT VM image to the following directories:

   2.1. vCGNAT VM qcow2 image to `/var/lib/libvirt/images/` directory

   2.2. vCGNAT VM installation script to `/var/lib/libvirt/images/hw-configuration/` directory

3. Install KVM hypervisor from a repository. The KVM hypervisor version must be 4.2 or more recent.

   ```
   sudo apt install qemu-kvm
   ```

4. Install the following packages and dependencies from a repository:

   - msr-tools

   ```
   sudo apt install msr-tools
   ```

5. Install the required packages and dependencies from a local directory:

   - virt-install

   ```
   cd /var/lib/libvirt/images/hw-configuration/virt-install/Ubuntu-20.04
   dpkg -i *
   ```

   If you have errors that require to install the missing dependencies and broken packages, use `apt` with key `fix-broken install`:

   ```
   apt --fix-broken install
   dpkg -i *
   ```

6. After installing the KVM hypervisor and packages, reboot the machine:

   ```
   reboot
   ```

1.1.4.3 Running Configuration Script

The configuration script helps allocate server resources to the vCGNAT virtual machine and turns on the autostart for vCGNAT VM.

1. Run the configuration script in the GUI mode:

   ```
   cd /var/lib/libvirt/images/hw-configuration/
   ./configure
   ```

   Note: Use character encoding UTF-8 in the console terminal if the GUI is displayed incorrectly.

2. Enter the path to a vCGNAT qcow2 image file.

   For example: /var/lib/libvirt/images/

3. Select the CPU configuration mode: auto or manual

The auto mode is used for simple installations, for example, if there is just one vCGNAT virtual machine on an entire server. It is preferred in most cases.

Manual mode is used for complex installations, where several VMs share server resources.

4. Select a number of NUMA nodes (CPU) used for vCGNAT VM.
5. Select a number of total cores used for vCGNAT VM.
   - Minimum – two for single-CPU, or four for dual-CPU configurations.
   - It is recommended to leave at least one core for the host OS in single-CPU configurations or two cores in dual-CPU configurations.

     For example:

     Single-CPU and 20 total cores – 19 cores are allocated to vCGNAT VM

     Dual-CPU and 72 total cores – 70 cores are allocated to vCGNAT VM
6. Enter vCGNAT VM name. Default name is **nfware-vcgnat**.
7. Enter memory settings for NUMA0 (CPU1).

   Depending on a server's total RAM, it is recommended set the following memory settings for the host OS:
   - Total RAM < 50GB – 2 GB
   - Total RAM from 50GB to 100GB – 4 GB
   - Total RAM > 100GB – 8 GB

   For example: If total RAM is 187 GB, then 8 GB is to be used for the host OS, and 179 GB is to be allocated to vCGNAT VM.
8. Enter the memory settings for NUMA1 (CPU2).

   It is recommended to use the NUMA0 configuration.
9. Select the NICs to use for the Data Plane.
10. Enter the management bridge name created in *Network Configuration*.

    For example: br-mgt
11. The host configuration has been completed. Reboot the machine.

1.1.4.4 Platform Configuration

The platform configuration is the main part of vCGNAT that impacts the solution performance. To change a platform parameter, use the following command:

```
nfware# platform set <>
```

To apply the platform settings, reboot the VM:

```
nfware# reboot
```

The platform settings configuration process consists of three main sections:

- Core settings
- Subscriber settings
- Other settings

Core Settings

Core settings define the configuration for each task type and apply separately to each CPU. For example, if you have allocated only 19 cores to vCGNAT VM in the previous step for single-CPU configuration, it means that 19 cores are to be used for platform settings. If you have a dual-CPU configuration and 70 cores are allocated, the number of cores to be used for platform settings must be divided by two. There are 35 cores to use for platform settings.

The vCGNAT has four types of cores. The total number of cores must be divided into these types:

nb_cmd
nb_cmd cores are used for Control Plane (VM OS, CLI, routing protocols, BFD, SNMP, LACP, SSH).

nb_rxtx
nb_rxtx cores are used for Data Plane (interface interaction, packet delivery to work cores, balancing).

nb_work
nb_work cores are used to provide NAT functionality.

nb_log
nb_log cores are used for sending NAT translation logs to a logging server.

Core Ratio Best Practice

- **nb_cmd**: one core – most commonly, two cores – for large installations.
- **nb_work**: **2-4 times** more than **nb_rxtx**. Preferred ratio is 2.
- **nb_log** uses the same cores as **nb_work**.

Example

For example, there are a total of 70 cores for vCGNAT VM on a dual-CPU configuration. Divided by 2 = 35 cores per CPU.

For **nb_cmd** we allocate 1 core, then the total for the rest will be 34.

nb_work should be 2–4 times more than **nb_rxtx**. Let's take 3 as an example.

```
nb_rxtx = 34/3  11,33
```

Round to a smaller even number = 10.

```
nb_work = 34 - 10 = 24

nb_log = nb_work = 24
```

Summary

```
VM cores = 70
per-socker cores = 35

nb_cmd = 1
nb_rxtx = 10
nb_work = 24
nb_log = 24
```

To configure these settings in NAT CLI:

```
nfware# platform set nb_cmd 1
nfware# platform set nb_rxtx 10
```

(continues on next page)

(continued from previous page)

```
nfware# platform set nb_work 24
nfware# platform set nb_log 24
```

Core Settings for Mellanox ConnectX-6

For this NIC, the configuration of cores can be simplified. Calculating the number of cores for rxtx, work, and log tasks will not be necessary. The cores minus the cores for **nb_cmd** will be shared between these tasks without performance loss. To do this:

1. Specify the `sched_isolation_mask` parameter equal to 1

   ```
   nfware# platform set sched_isolation_mask 1
   ```

2. Specify the same number of cores for **nb_rxtx**, **nb_work**, and **nb_log**. For example, if the vCGNAT VM has 92 cores allocated on a dual-CPU server, then platform settings would be as follows:

   ```
   nfware# platform set nb_cmd 2
   nfware# platform set nb_rxtx 44
   nfware# platform set nb_work 44
   nfware# platform set nb_log 44
   ```

Subscriber Settings

Subscriber settings define how many subscribers and sessions NAT can hold. It depends on the memory size and license limitations.

```
nfware# platform set max_subscribers <>
nfware# platform set max_sessions <>
```

The maximum subscribers parameter is mandatory.

The maximum sessions is optional. If it is not specified, that parameter is set automatically:

```
maximum sessions = 200 * maximum subscribers
```

Other Settings

Other options contain several parameters used to optimize vCGNAT to have the best performance. Most commonly used:

- **A number of CPU memory channels**

  Check those in the CPU specification. By default, it is 4.

  ```
  nfware# platform set nb_mem_channels <>
  ```

- **A number of NIC RX default queue size (in a number of packets)**

  The recommended values are:

  - 1024 for Intel 1/10/40G NICs,
  - 2048 for Mellanox 100G NICs.

  The default value is 1024.

  ```
  nfware# platform set nb_rx_desc_default <>
  ```

- **A number of NIC TX default queue size (in a number of packets)**

  The recommended values are:

  - 1024 for Intel 1/10/40G,
  - 2048 for Mellanox 100G.

  By default, it is 1024.

  ```
  nfware# platform set nb_tx_desc_default <>
  ```

1.2 First Access

After the first startup of the virtual machine you can access it only through the console.

To do this, you can use a `virt-manager` GUI application or a `virsh` command-line utility. For example, if you created a virtual machine named `nfware-vcgnat`, then you can use the following command to access its console:

```
# virsh console nfware-vcgnat

Connected to domain nfware-vcgnat
Escape character is ^]

nfware login:
```

Use the default username and password `admin` to log in.

For more convenient management of the virtual machine, it is recommended to configure network access to the management interface and activate SSH access.

To set up the management interface you need to:

- Enter the configuration mode:

  ```
  nfware# configure terminal
  ```

- Enter the configuration mode of the management interface:

  ```
  nfware(config)# interface management
  ```

- Set the IP address and a default gateway, for example:

  ```
  nfware(config-management)# ip address 192.168.1.100/24
  nfware(config-management)# ip default-gateway 192.168.1.1
  nfware(config-management)# exit
  ```

To enable the SSH protocol support, run the command:

```
nfware(config)# service ssh enable
```

Do not forget to save the changes by running the command:

```
nfware(config)# do write
```

After completing these steps, you will be able to connect to the system remotely and proceed to further configuration. You can exit the console by using the keyboard shortcut `Ctrl + ]`.

1.3 License Installation

Without the license file, you will have a maximum of 2 processor cores available for running a virtual machine – one to ensure the operation of the OS and another to process the user traffic.

This should be enough to handle the traffic up to 1Gbps. In case a higher performance is required, purchase a license and install the license file to the VM.

In order to obtain a license file after you have purchased the license, you need to tell your manager the ID of the virtual machine installed, which can be obtained using the command:

```
# show license host-id
03196D973B6D24649CB55F18804652B8
```

In the example above, `03196D973B6D24649CB55F18804652B8` is the VM ID to be sent to your manager. In return, you will receive a license file according to the license acqiured.

> **Warning:** If you change the configuration of the virtual machine or transfer the virtual machine to another server, its ID may change and the license file will be invalid. Always consult your manager before performing such actions.

The obtained license file must be uploaded to the VM, for example, with the `scp` command:

```
scp 03196D973B6D24649CB55F18804652B8.license admin@vCGNAT_MANAGEMENT_IP:
```

where `vCGNAT_MANAGEMENT_IP` is the IP address assigned to the management interface of the virtual machine.

Before using the obtained license file, it is recommended to check its parameters:

```
show license limits /home/admin/03196D973B6D24649CB55F18804652B8.license
```

If you have downloaded the license file not as the `admin` user, then you'll need to replace the `admin` in the command above with the according username.

To apply the license file, run the command:

```
license file /home/admin/03196D973B6D24649CB55F18804652B8.license
```

If you have downloaded the license file not as the `admin` user, then you'll need to replace the `admin` in the command above with the according username.

After applying the license file, you need to restart the VM:

```
reboot
```

After restarting the VM, you can check the parameters of the license applied using the command:

```
show license limits
```

1.4 Running vCGNAT in Docker

Warning: This installation is only suitable for functional tests!

First, install Docker and KVM. Make sure the following modules are installed:

```
$ modprobe kvm kvm_intel
```

1.4.1 Building Docker Image

You do not need to build a Docker image yourself. You can contact our sales department for Docker and qcow2 images.

1.4.2 Running Container

Note: Here is a description of the network connection to the container using `veth` interfaces.

- Run the container by specifying the required number of dataplane interfaces in the CLAB_INTFS environment variable. In this example, the default Docker network is used for management (usually it is 172.17.0.1/16). The address from this network is assigned to the vCGNAT management interface. It is possible to create another network and connect it with the `--network` option:

  ```
  $ docker run  -d --name vcgnat -e CLAB_INTFS=1 --privileged <image_name> --username admin --passwor
  ```

- It's obligatory to create a shell access to the network namespace of the container (make network namespace available to work using the shell):

  ```
  $ export pid="$(docker inspect -f '{{.State.Pid}}' vcgnat)"
  ```

  ```
  $ sudo ln -sf /proc/$pid/ns/net "/var/run/netns/vcgnat"
  ```

- Using a pair of `veth` interfaces, associate the main network namespace and the namespace in which vCGNAT works. reate as many pairs of interfaces as there are interfaces required for the dataplane. Enable interfaces on both sides:

  ```
  $ sudo ip link add eth101 type veth peer eth1 netns vcgnat
  ```

  ```
  $ sudo ip link set eth101 up
  ```

  ```
  $ sudo ip netns exec vcgnat ip link set eth1 up
  ```

- After that, the script inside the container should be able to find the dataplane interfaces that have been already added into it, and the virtual machine should start loading:

  ```
  user:~$ docker logs vcgnat
  2022-09-08 12:04:35,470: vrnetlab   DEBUG    Creating overlay disk image
  2022-09-08 12:04:35,527: vrnetlab   DEBUG    Starting vrnetlab NFWare
  2022-09-08 12:04:35,527: vrnetlab   DEBUG    VMs: [<__main__.NFWare_vm object at␣
  ↪0x7f44598bae80>]
  2022-09-08 12:04:35,531: vrnetlab   DEBUG    VM not started; starting!
  2022-09-08 12:04:35,531: vrnetlab   INFO     Starting NFWare_vm
  2022-09-08 12:04:35,531: vrnetlab   DEBUG    number of provisioned data plane␣
  ```

(continues on next page)

(continued from previous page)

```
↪interfaces is 1
2022-09-08 12:04:35,532: vrnetlab   DEBUG    waiting for provisioned interfaces to␣
↪appear...
2022-09-08 12:05:00,556: vrnetlab   DEBUG    interfaces provisioned, continuing...
2022-09-08 12:05:00,557: vrnetlab   DEBUG    ['qemu-system-x86_64', '-enable-kvm',
↪'-display', 'none', '-machine', 'pc', '-monitor',
'tcp:0.0.0.0:4000,server,nowait', '-m', '7000', '-serial', 'telnet:0.0.0.0:5000,
↪server,nowait', '-drive', 'if=ide,file=/None_968cbe26
405a_vcgnat_4.3.2.qcow2,cache=unsafe', '-cpu', 'host', '-smp', '2', '-monitor',
↪'tcp:0.0.0.0:3000,server,nowait', '-device', 'pci-bri
dge,chassis_nr=1,id=pci.1', '-device', 'virtio-net-pci,netdev=p00,
↪mac=52:54:00:1e:cc:00', '-netdev', 'user,id=p00,net=10.0.0.0/24,tft
p=/tftpboot,hostfwd=tcp::2022-10.0.0.15:22,hostfwd=udp::2161-10.0.0.15:161,
↪hostfwd=tcp::2830-10.0.0.15:830,hostfwd=tcp::2080-10.0.0.1
5:80,hostfwd=tcp::2443-10.0.0.15:443', '-device', 'virtio-net-pci,netdev=p01,
↪mac=52:54:00:e2:38:01,bus=pci.1,addr=0x2', '-netdev', 't
ap,id=p01,ifname=tap1,script=/etc/tc-tap-ifup,downscript=no']
2022-09-08 12:05:00,557: vrnetlab   DEBUG    joined cmd: qemu-system-x86_64 -enable-
↪kvm -display none -machine pc -monitor tcp:0.0.0.
0:4000,server,nowait -m 7000 -serial telnet:0.0.0.0:5000,server,nowait -drive␣
↪if=ide,file=/None_968cbe26405a_vcgnat_4.3.2.qcow2,cache
=unsafe -cpu host -smp 2 -monitor tcp:0.0.0.0:3000,server,nowait -device pci-bridge,
↪chassis_nr=1,id=pci.1 -device virtio-net-pci,netd
ev=p00,mac=52:54:00:1e:cc:00 -netdev user,id=p00,net=10.0.0.0/24,tftp=/tftpboot,
↪hostfwd=tcp::2022-10.0.0.15:22,hostfwd=udp::2161-10.0
.0.15:161,hostfwd=tcp::2830-10.0.0.15:830,hostfwd=tcp::2080-10.0.0.15:80,
↪hostfwd=tcp::2443-10.0.0.15:443 -device virtio-net-pci,netde
v=p01,mac=52:54:00:e2:38:01,bus=pci.1,addr=0x2 -netdev tap,id=p01,ifname=tap1,
↪script=/etc/tc-tap-ifup,downscript=no
```

- Loading takes a few minutes, and once it is finished, you can connect to vCGNAT via ssh:

```
user:~$ ssh admin@172.17.0.2
admin@172.17.0.2's password:
Last login: Thu Sep  8 12:09:14 2022

Hello, this is NFWare OS.

vcgnat# sh int brief
Interface       Status  VRF             Addresses
---------       ------  ---             ---------
if0             up      default
lo              up      default

vcgnat#
```

- Add `veth` interfaces in the main network namespace to bridges and send traffic using them according to your needs:

$ sudo ip link add br-vcgnat type bridge

$ sudo ip link set up br-vcgnat

$ sudo brctl addif br-vcgnat eth101

```
$ sudo ip a add 10.0.100.1/24 dev br-vcgnat
```

CHAPTER

TWO

ANNOTATION

2.1 Notations

The list of symbols used in commands and their explanation is below:

Name	Symbols	Description
Parentheses	(. . .)	Choose from a range of values
Square brackets	[. . .]	This filter or filters are optional
Braces	{. . . }	Choose one or several filters
Angle brackets	<. . . >	Selection from one of the values is required

2.2 Glossary

The explanation of the words and expressions used in the documentation for vCGNAT is below:

Words and Expressions	Description
Entries	Means the quantity of the entries at the moment
Creations	Means the quantity of the creations during for the entire time of operation of the device
Overall counters	Counters during the whole period of the operation

CHAPTER

THREE

CLI

3.1 CLI Access

By default, the command prompt is available through the console only.

To access the command prompt, you can use a `virt-manager` GUI application or a `virsh` command-line utility. For example, if you created a virtual machine named `nfware-vcgnat`, then you can use the following command to access its console:

```
# virsh console nfware-vcgnat

Connected to domain nfware-vcgnat
Escape character is ^]

nfware login:
```

Use the following set of commands to enable access to a command prompt via the SSH protocol:

```
nfware# configure terminal
nfware(config)# service ssh enable
```

Use the following commands to manage command prompt parameters:

`service ssh enable`
    Enable access to the command prompt via the SSH protocol.

`show cli sessions`
    List active user connections to the command prompt.

`clear cli sessions (1-4294967295)`
    Terminate user connection to the command prompt.

3.2 Basic Commands

3.2.1 CLI Modes

There are several modes in CLI:

View mode
    Basic mode with read-only commands available.

Enable mode
A privileged mode of operation with read-only and write commands available.

Configuration mode
Device configuration commands are available in this mode.

3.2.2 View Mode

The following commands are available in the view mode:

`enable`
Switch to the enable mode.

`ping WORD [vrf NAME]`

`ping management WORD`
Send ping requests to the `WORD` host. Basic command `ping` works through the data interfaces. To send requests through the management interface the `ping management` command is used.

`traceroute WORD [vrf NAME]`

`traceroute management WORD`
Send traceroute requests to the `WORD` host. Basic command `traceroute` works through the data interfaces. To send requests through the management interface the `traceroute management` command is used.

`tcpdump IFNAME`
Run the `tcpdump` utility on the `IFNAME` interface.

`ssh WORD`
Connect to the `WORD` host via the SSH protocol.

`telnet WORD [PORT]`
Connect to the `WORD` host’s `PORT` port via the Telnet protocol.

`terminal paginate`
Enable paginated output. This mode is enabled by default.

`exit`

`quit`
Exit from the command prompt.

There are also many `show` commands available in this mode to view and clear the system state. Their description is given in the corresponding sections.

3.2.3 Enable Mode

All the view mode commands are available in the enable mode, as well as the following commands:

`disable`
Switch to the view mode.

`configure [terminal]`
Switch to the configuration mode.

`start-shell [{bash|zsh}]`
Enter the operating system shell.

`list [permutations]`

List all commands available in this mode. With the `permutations` option commands are listed according to all possible parameters permutations.

`find REGEX...`

Search for a command in the CLI matching the `REGEX...` regular expression. The regular expression is entered in the POSIX Extended Regular Expressions format without the quotes. Spaces are allowed.

`show running-config`

View the running configuration of the system.

`show startup-config`

View the startup configuration of the system.

`show backup-config`

View the backup configuration of the system.

`copy running-config startup-config`

`write`

Save the running configuration of the system as the startup one. The previous startup configuration is then written as the backup configuration.

`copy backup-config startup-config`

Restore the system's startup configuration from the backup.

`copy running-config FILENAME`

Save the running configuration of the system to the `FILENAME` file.

`copy startup-config FILENAME`

Save the system's startup configuration to the `FILENAME` file.

`copy FILENAME startup-config`

Overwrite the system's startup configuration from the `FILENAME` file.

`vim FILENAME`

`nano FILENAME`

Edit the `FILENAME` file using the specified editor.

`rm FILENAME`

Remove the `FILENAME` file.

`ls`

List the files in the current working directory.

`scp FROM TO`

Run the `scp` utility to copy files from `FROM` to `TO` in a secure way.

`reboot`

Restart the system.

`poweroff`

Shutdown the system.

`exit`

quit

Exit the command prompt.

There are also many `show` and `clear` commands available in this mode to view and clear the system state. Their description is given in the corresponding sections.

3.2.4 Configuration Mode

The configuration mode is used to change the running system settings. Available configuration commands are described in the corresponding sections. For the most of the configuration commands, a version with the `no` prefix for the cancellation is also available.

For example, if you entered the `ip route 0.0.0.0/0 192.168.1.1` command, use the `no` prefix to cancel it: `no ip route 0.0.0.0/0 192.168.1.1`. All commands in the documentation are described without the `no` prefix.

All commands from the enable mode are available in the configuration mode, use the `do` prefix to run them. This allows you to quickly check the system state during configuration without exiting configuration mode, for example:

```
nfware# configure terminal
nfware(config)# interface if0
nfware(config-if)# ip address 192.168.1.100/24
nfware(config-if)# do show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, T - Table, v - VNC,
       V - VNC-Direct,
       > - selected route, * - FIB route, q - queued, r - rejected, b - backup

C>* 192.168.1.0/24 is directly connected, if0, 00:00:01
```

During the configuration process, you can enter deeper configuration modes for the individual parts of the system. For example, the interface configuration mode as shown in the example above. To immediately exit from the configuration mode to the enable mode, use the `end` command:

end

3.2.5 Pipe Actions

You can perform additional actions with the output of CLI commands using the pipe actions.

... | include REGEX...

Filter the CLI output with the `REGEX...` regular expression. The regular expression is entered in the POSIX Extended Regular Expressions format without the quotes. Spaces are allowed.

3.3 Show Commands

show memory dataplane

Display the amount of memory used by objects of the Data Plane. You can use this command to find out how much memory is used by different objects like IP pools in NAT, ACL, etc.

show memory dataplane pools

Display resource usage for each entity (like subscribers, sessions, etc) per socket:

```
# show memory dataplane pools
------------------------------------------------------------------------------------
↪---
Statistics for Socket 0
------------------------------------------------------------------------------------
↪---
                       Total         Used          Free          Drops             ␣
↪Load
------------------------------------------------------------------------------------
↪---
Subscribers            1527648       161520        1366128       0                ␣
↪10.6%
Mappings               85027648      1096285       83931363      0                 ␣
↪1.3%
Sessions               85027648      1164505       83863143      0                 ␣
↪1.4%
Fragments              65536         0             65536         0                 ␣
↪0.0%
Pending Fragments      1024          2             1022          0                 ␣
↪0.2%
------------------------------------------------------------------------------------
↪---
------------------------------------------------------------------------------------
↪---
Statistics for Socket 1
------------------------------------------------------------------------------------
↪---
                       Total         Used          Free          Drops             ␣
↪Load
------------------------------------------------------------------------------------
↪---
Subscribers            1527648       144786        1382862       0                 ␣
↪9.5%
Mappings               85027648      970063        84057585      0                 ␣
↪1.1%
Sessions               85027648      1030252       83997396      0                 ␣
↪1.2%
Fragments              65536         4             65532         0                 ␣
↪0.0%
Pending Fragments      1024          2             1022          0                 ␣
↪0.2%
------------------------------------------------------------------------------------
↪---
------------------------------------------------------------------------------------
↪---
                       Total         Used          Free          Drops             ␣
↪Load
------------------------------------------------------------------------------------
↪---
IPv4 neighbors         1024          12            1012          0                 ␣
↪1.2%
IPv6 neighbors         1024          0             1024          0                 ␣
↪0.0%
```

(continues on next page)

(continued from previous page)

```
ARP wait_ctx 0          69632          15            69617          0                ␣
↪0.0%
ARP wait_ctx 1          69632          15            69617          0                ␣
↪0.0%
-----------------------------------------------------------------------------------------
↪---
```

clear memory dataplane pools

Clear `Drops` statistics in memory dataplane pools table.

show memory dataplane debug NAME...

Display detailed information on the amount of memory used by the Data Plane objects. Also, you can see the memory addresses of the objects.

show overruns

clear overruns

All of these counters are packet losses that have occurred because a particular internal queue has run out of space:

Counter	Description
`Interna Overrun`	Internal Overruns are internal packet queues sent to the operating system kernel. This is all traffic that goes to the IP addresses configured on the NAT interface
`Work Overrun`	After we have read a packet from the queue of the network interface, the RXTX tasks balance the Work tasks where all the packet processing takes place. The overrun data tells us that the packet was lost because it could not be passed to the next processing stage after being read from the interface
`TX Overrun`	After processing, the packets are delivered to the RXTX tasks that send them to the network interface. These losses indicate an overflow of queues for sending between Work and RXTX tasks

show cpu

Display the maximum load of the processor core at the moment. If, for example, you have 50 cores and, for some reason, only one is loaded, and all the others are idle, then the command will show exactly this load, and there will be no averaging.

show cpu debug

Display the load for each processor core:

- `socket` - processor ID
- `core` - the number of the physical core
- `lcore` - the number of the logical cores. One physical core may have several logical ones

show cpu task

Display which core is responsible for which tasks.

show debugging nat-hash-stats

Display bucket information for Session, Mapping, Subscriber, and IP fragment: the amount of the elements in the bucket and how many buckets with such elements are in the system.

show debugging packets-pool

Display information about packet pools. NAT is the device that handles packet processing. After a packet arrives at the NIC, we read it into our internal mbuf data structure. The packet pool is a predefined and prepared pool from which we can take the mbuf internal packet data structure. The size of the pool is the number of packets that can be in the system at the same time.

`show debugging queues-internal`

Display debugging information on internal queues.

`show license host-id`

Display the host ID that is used for licensing.

`show license limits [FILENAME]`

Display the limits on the host issued by the license. When specifying the file name, check this information from it.

`show platform settings`

Display the low-level vCGNAT settings.

`show port [(0-255)]`

Display full information about the physical port.

CHAPTER

FOUR

SYSTEM CONFIGURATION

4.1 System Name Configuration

`hostname NAME`

Set the system name. By default, the `nfware` name is used.

4.2 DNS Configuration

`ip dns <primary|secondary> A.B.C.D`

Add a DNS server.

`ip dns domain DOMAIN`

Set the DNS domain.

4.3 Time Configuration

4.3.1 Local Clock Configuration

`clock time (0-23) (0-59) (0-59) (1-31) (1-12) (1971-2099)`

Set the current system time. This command is not available when using NTP.

`clock timezone TIMEZONE`

Set the system time zone. The list of time zones available for setting can be obtained by entering the question mark: `clock timezone ?`. The default time zone is UTC.

`show clock`

Show current system time.

4.3.2 NTP Configuration

`service ntp enable`

Enable NTP. If NTP is enabled, but no NTP server is configured, the `pool.ntp.org` server is used by default.

`service ntp server <HOSTNAME|A.B.C.D>`

Add an NTP server.

`show ntp`

Check time synchronization status via NTP.

4.4 Logging

Important events that occur in the system are logged locally using the Syslog protocol.

The following commands are available to configure operational logging:

`log syslog [<emergencies|alerts|critical|errors|warnings|notifications|informational|debugging>]`

Set the level of logged events. This command affects subsystems related to routing and traffic processing. Logging of other operating system modules is always enabled. When using the command without specifying the level, the `debugging` level is used.

`log facility [<kern|user|mail|daemon|auth|syslog|lpr|news|uucp|cron|local0|local1|local2|local3|local4|`

Set the facility of the syslog events. See **RFC 5424** for reference. The default is `daemon`.

`log server <tcp|udp> A.B.C.D (1-65535)`

This command allows you to enable additional sending of operational logs to an external Syslog server.

`show logging`

Show logged events.

CHAPTER

FIVE

AAA CONFIGURATION

AAA stands for authentication, authorization and accounting. This subsystem is implemented when it is necessary to provide an identification of a user, determine what commands the user has authority to execute and track all user's actions.

Important: Changes in the AAA settings do not apply to sessions that have already been created!

5.1 Local User Configuration

There is one default local user in the system named `admin` with the password `admin` in the group `admin` with all configuration commands available to it.

Commands to manage users are listed below.

`user USERNAME group <guest|operator|admin> encryption <md5|sha-256|sha-512> password PASSWORD`

Add a new user to the specified group. The password is entered in plain text and will be hashed using the `encryption` algorithm.

`user USERNAME group <guest|operator|admin> password PASSWORD_HASH`

Add a new user to the specified group. The password is entered in a hashed form in the Modular Crypt Format: `($<id>[$<param>=<value>(,<param>=<value>)*][$<salt>[$<hash>]])`. The MD5 (`$1$`), SHA-256 (`$5$`), and SHA-512 (`$6$`) algorithms are supported.

`user USERNAME group <guest|operator|admin>`

Move the user to the specified group.

To configure the password for Enable Mode, use the following commands:

`enable encryption <md5|sha-256|sha-512> password PASSWORD`

`enable password PASSWORD_HASH`

Note: For RADIUS and TACACS+ Enable Mode configuration is applied on the servers.

5.2 RADIUS and TACACS+ Configuration

First, set the settings for RADIUS or TACACS+ servers to which vCGNAT will connect:

`aaa radius id (0-99) A.B.C.D auth-port (1-65535) acct-port (1-65535) secret SECRET`

Set connection to the RADIUS server.

Keys	Argument	Description
`id`	`(0-99)`	Set the ID of the RADIUS server
`HOST`	`A.B.C.D` or `hostname`	RADIUS server IP address or hostname
`auth-port`	`(1-65535)`	RADIUS authentication port. The defaul is 1812
`acct-port`	`(1-65535)`	RADIUS accounting port. The default is 1813
`secret`	`SECRET`	A password which vCGNAT will use to connect to RADIUS server

The attribute `cisco-avpair` (9,1) is used:

`shell:priv-lv`	For privilege level
`shell:cmd=`	To forbidden an individual command (all commands whose beginning coincides with the specified one will be forbidden)

The password configuration for Enable Mode is done by creating user `$enab15$` on the RADIUS server.

```
test    Cleartext-Password := "test"
        cisco-avpair := "shell:priv-lvl=7", cisco-avpair := "shell:cmd=show interface"

$enab15$ Cleartext-Password := "enable"
```

`aaa tacacs+ id (0-99) HOST port (1-65535) secret SECRET`

Set a connection to the TACACS+ server.

Keys	Argument	Description
`id`	`(0-99)`	Set the ID of the TACACS+ server
`HOST`	`A.B.C.D` or `hostname`	TACACS+ server IP address or hostname
`port`	`(1-65535)`	TACACS+ server port. The default is 49
`secret`	`SECRET`	A password which vCGNAT will use to connect to TACACS+ server

Note: You can add several RADIUS or TACACS+ servers to increase fault tolerance of the system. In that case, vCGNAT will connect to the server which comes with the highest ID number.

`aaa server <radius|tacacs+> id (0-99) set-id (0-99)`

Change the server ID. If the server with the new ID already exists, servers are swapped.

`no aaa <radius|tacacs+> [id (0-99)]`

Remove all connections to RADIUS or TACACS+ servers or specified by ID.

5.3 Accounting

`aaa accounting <console|ssh> {local|radius|tacacs+}`

Set the type of connection (via SSH or console) and the location where users' actions will be sent to. Logs contain the beginning/end of the session and all commands entered by the user with the authorization result.

Pay attention: `radius` and `tacacs+` here mean two groups of servers. For example, if you add two RADIUS and three TACACS servers, and specify `local radius tacacs+` in the accounting, then the vCGANT will sent logs to the local, and to the RADIUS server with the highest ID in the group of the RADIUS servers, and to the TACACS server with the highest ID in the group of the TACACS servers.

`no aaa accounting <console|ssh>`

Disable accounting.

5.4 Authentication

`aaa authentication <console|ssh> {local|radius|tacacs+}`

Set a type of connection (via SSH or console) and a method for authentication. You can specify several methods and the vCGNAT will use the first of the specified. `radius` and `tacacs+` methods mean two groups of servers. For example, if you specify for authentication `radius tacacs` and add several RADIUS and TACACS servers, then the vCGNAT will first try to connect to the RADIUS server with higher ID and so on, and then, if all RADIUS servers are not available, proceed to TACACS servers.

Pay attention to two things:

1. The local server is always available, so the following servers will not be polled.
2. Denied access is interpreted as a successful response, so the next server will not be polled.

During authentication, a privilege level (priv-lvl) is requested from the same source that allowed access. According to the received privilege level, the operator role is set:

admin	=15
operator	>=7; <15
guest	<7

`no aaa authentication <console|ssh>`

Disable remote authentication.

5.5 Authorization

`aaa authorization <console|ssh> {local|radius|tacacs+}`

Set a type of connection (via SSH or console) and a method for authorization. You can specify several methods and the vCGNAT will use the first of the specified. `radius` and `tacacs+` methods mean two groups of servers. For example, if you specify for authorization `radius tacacs` and add several RADIUS and TACACS servers, then the vCGNAT will first try to connect to the RADIUS server with higher ID and so on, and then, if all RADIUS servers are not available, proceed to TACACS servers.

Pay attention to two things:

1. The local server is always available, so the following servers will not be polled.

2. Denied access is interpreted as a successful response, so the next server will not be polled.

Some important points

When the local authorization has been implemented at the user's privilege level, individual commands cannot be allowed/forbidden.

RADIUS authorization allows only to prohibit commands (you need to choose a higher privilege level initially).

If the session has not been authenticated through RADIUS, RADIUS authorization will not be available and other methods will be used in that order.

TACACS+ authorization allows you to configure any access rights. It is not necessary to use TACACS+ authentication before using authorization.

`no aaa authorization <console|ssh>`

Disable remote authorization.

5.6 Show Commands

`show aaa server debug`

Display debug information for all configured connections. The output is as follows:

```
nfware# show aaa server debug
Radius server #0: 192.168.1.10 auth-port:1812 acct-port:1813
Accept: 0
Reject: 0
Connection error: 0

Tacacs+ server #0: 192.168.1.11:49
Accept: 95
Reject: 0
Connection error: 0
```

`show aaa server radius`

Display information about RADIUS connections: IP address or hostname, auth-port and acct-port.

`show aaa server tacacs+`

Display information about TACACS+ connections: IP address or hostname and port.

`show aaa sessions`

Display information about all sessions. The output includes: Username, TTY, Remote IP, Authenticator, Start Time, Last Activity, ID.

`show aaa sessions username [USERNAME]`

Display information about sessions for the specified user.

`clear aaa session SESSION_ID`

Clear the specified session.

CHAPTER

SIX

INTERFACES CONFIGURATION

As explained in the *Virtual Machine Installation* section, the first interface of the virtual machine is always used to manage the system and is called `management`, the rest are used to process the user traffic and are called `ifX`, where `X` - is the interface index.

For example, if you pass three interfaces to the VM, you will then have the following set of interfaces:

```
interface management
interface if0
interface if1
```

For data interfaces, it is possible to create VLAN subinterfaces, as well as aggregate them using the LACP protocol.

The configuration of various types of interfaces is described in the corresponding sections:

6.1 Management Interface

The management interface is used to connect to the system over the network for its further configuration and monitoring. This interface is also used for NTP, SNMP operation and to send operational logs to external servers.

To configure the management interface, enter its configuration mode:

`interface management`

In the configuration mode, the following commands are available:

`ip address A.B.C.D/M`

Configure the IPv4 address of the management interface. The management interface can have only one address. When you re-enter this command, the interface address will be replaced with a new one.

`ip default-gateway A.B.C.D`

Configure the default gateway for the management interface. The management interface can access other networks only through the default gateway. You cannot configure dynamic routing protocols for the management interface.

6.2 Data Interfaces

6.2.1 Configuration

To configure the data interface with the name `IFNAME`, you need to enter its configuration mode:

```
interface IFNAME
```

All commands described below are entered in the interface configuration mode. This section describes the basic commands to configure interfaces. Commands related to the specific dynamic routing protocols are described in the corresponding sections.

`description LINE...`

Set the interface description.

`shutdown`

Administratively shutdown the interface. By default, all interfaces are enabled.

`mac X:X:X:X:X:X`

Set the MAC address of the interface.

`ip address A.B.C.D/M`

Assign an IPv4 address to the interface. You can add several addresses to one interface.

`ipv6 address X:X::X:X/M`

Assign an IPv6 address to the interface. You can add several addresses to one interface.

`ip nat <inside|outside>`

Enable NAT for IPv4 traffic on the interface. The `inside` type should be set on interfaces facing subscribers, and the `outside` type should be set on interfaces facing the Internet. By default, NAT is not enabled for IPv4, and the traffic is routed without address translation.

`ipv6 nat <inside|outside>`

Enable NAT for IPv6 traffic on the interface. The `inside` type should be set on interfaces facing subscribers, and the `outside` type should be set on interfaces facing the Internet. By default, NAT is not enabled for IPv6, and the traffic is routed without address translation.

`mtu (68-9216)`

Set the L2 MTU of the interface. By default, the L2 MTU is 1500 bytes.

`ip mtu (68-9216)`

Set the IPv4 MTU of the interface. This MTU will be used in case it is necessary to fragment IPv4 packets when sending. By default, IPv4 MTU is equal to L2 MTU.

`ipv6 mtu (1280-9216)`

Set the IPv6 MTU of the interface. This MTU will be used in case it is necessary to fragment IPv6 packets when sending. By default, IPv6 MTU is equal to L2 MTU.

`bond IFNAME`

Add the interface to the `IFNAME` bond. The aggregation configuration is described in the *Link Aggregation* section.

`vrf VRFNAME`

Add the interface to the `VRFNAME` VRF. The VRF configuration is described in the *VRF* section.

6.2.2 Show Commands

`show interface [KEYS]`

Display the following information for all interfaces:

- Admin and Oper status
- Ethernet status
- MTU
- IPv4 and IPv6 NAT type
- IPv4 and IPv6 addresses
- Interface speed
- Duplex
- Autonegotiation On/Off
- Input/Output rate
- RX and TX statistics

You can specify following keys:

Keys	Description
`IFNAME`	Display information for the specified interface
`IFNAME nexthop-group`	Display nexthop group information for the specified interface
`IFNAME transceiver`	Display SFP transceiver information for the specified interface
`brief`	Display brief information for all interfaces such as: interface name, its status, VRF belonging, and IP address
`debug`	Display debugging information for RX and TX statistics in addition to general one: • `RX Overruns` There are no free packets in the packet pool to allocate to read data from the network interface queue. There are no drops in this case. When the network interface queue is full, the `Discard counter` will start increasing • `RX Invalid VLAN` The packet came on the interface with VLAN ID that is not configured on this interface • `RX Unsupported` Unsupported Ethernet types • `TX Overruns` There is not enough space in the network interface queue to put a packet in to send it. No loss in this case. The attempts to send the packet will be continued
`description`	Display description for each interface
`rate`	Display I/O traffic loading (in pps and bps) for all interfaces
`vrf NAME brief`	Display brief information of the interfaces for the specified VRF
`vrf all brief`	Display brief information of the interfaces for all VRF

`clear interface IFNAME`

Clear RX and TX statistics for all interfaces or for the specified one.

6.3 Subinterfaces

A single *data interface* or *bond* can be divided into several logical subinterfaces using a VLAN ID tag in accordance with the IEEE 802.1Q standard.

Logical subinterfaces can be used to isolate traffic from different VLANs using VRFs, as well as, for example, to separate inbound and outbound NAT traffic.

To create a logical subinterface, use the command:

`interface IFNAME`

The `IFNAME` parameter is specified in the `NAME.ID` format, where `NAME` - is the name of the interface for which the subinterface is being created, and `ID` - is the VLAN ID which traffic will be isolated.

The same commands are available for configuring subinterfaces as for normal *data interfaces*, with the following limitations:

- the subinterface cannot be added to the bond,
- the subinterface inherits the MTU of the parent interface.

6.4 Link Aggregation

Multiple *data interfaces* can be combined into a single logical bond interface to increase fault tolerance and simplify routing configuration. Currently, aggregation of interfaces via the LACP protocol is supported in accordance with the IEEE 802.3 standard.

Use the following command to create a bond interface:

`interface IFNAME`

The `IFNAME` parameter is specified in the `bondX` format, where X is the index of the created interface (can be any number).

To configure the bond interfaces, the same commands as for normal *data interfaces*, as well as additional commands described below are available.

`lacp min-links (1-64)`

Set the minimum number of interfaces with active connections in the bond, required to consider the bond active. By default, this number is 1.

`lacp timeout <fast|slow>`

Set the interval between sending the LACP protocol control messages. `fast` - 1 second, `slow` - 30 seconds. By default, the `slow` interval is used.

6.4.1 Show Commands

`show bond IFNAME`

Display all bonds and the interfaces that are bonded into them. You can specify the bond by `IFNAME`.

The LACP port state (also known as the actor state) field is a single byte, each bit of which is a flag indicating a particular status. In this table, mux (i.e., a multiplexer) refers to the logical unit aggregating the links into a single logical transmitter/receiver.

Bit ID	Bit value	Name	Description
0	0 = Passive 1 = Active	LACP_Acti	The Device intends to transmit periodically to find potential members for the aggregate. This is toggled by the mode active in the channel-group configuration on the member interfaces. The default is active
1	0 = Long Timeout 1 = Short Timeout	LACP_Tim	The LACP Timeout flag indicates that the participant wishes to receive frequent periodic transmissions and will aggressively times out received information.
2	0 = No 1 = Yes	Aggregation	The flag indicates that the participant will allow the link to be used as part of an aggregate. Otherwise, the link will be used as an individual link, i.e., not aggregated with any other. This flag is set or reset as a consequence of local key management: the participant may know that the link has a unique key and, hence, will not be aggregated. Signaling this information allows the receiving actor to skip protocol delays that are otherwise invoked. This allows all links with the same system ID and key combinations to be collected into one aggregate port without successive rapid changes to aggregate ports and accompanying higher-layer protocol disruption. If set the flag communicates Aggregatable, if reset Individual
3	0 = Not in sync 1 = In sync	Synchronization	The flag indicates that the transmitting participant's mux component is in sync with the system id and key information transmitted. This accommodates multiplexing hardware that takes time to set up or reconfigure
4	0 = No 1 = Yes	Collecting	TRUE (encoded as a 1) means collection of incoming frames on this link is definitely enabled; i.e., collection is currently enabled and is not expected to be disabled in the absence of administrative changes or changes in received protocol information
5	0 = No 1 = Yes	Distributing	FALSE (encoded as a 0) means the distribution of outgoing frames on this link is definitely disabled; i.e., distribution is currently disabled and is not expected to be enabled in the absence of administrative changes or changes in received protocol information
6	0 = via LACP PDU 1 = default settings	Defaulted	If TRUE (encoded as a 1), this flag indicates that the Actor's Receive machine uses Defaulted operational Partner information, administratively configured for the Partner. If FALSE (encoded as a 0), the operational Partner information in use has been received in a LACPDU
7	0 = No 1 = Yes	Expired	If TRUE (encoded as a 1), this flag indicates that the Actor's Receive machine is in the EXPIRED state; if FALSE (encoded as a 0), this flag indicates that the Actor's Receive machine is not in the EXPIRED state

In most cases, lacp fast mode is chosen. To understand that the link aggregation is established in this mode, look at the `port state` line for actor LACPDU and partner LACPDU. The state should be `63`. Convert this value from decimal to binary. The binary will be `00111111`, where the leftmost bit is the high bit and its `bit ID` is `7`, and the rightmost bit is the low bit and its `bit ID` is `0`. So, the state `63` will mean LACP Activity is active, and the Actor's Receive machine is not in the EXPIRED state.

CHAPTER

SEVEN

ROUTING CONFIGURATION

7.1 VRF

VRF support allows to create multiple independent routing tables on the same device.

The traffic is distributed between the VRFs according to the binding of the data interfaces. Initially, only the default VRF exists on the device, and all data interfaces are bound to it. To create a new VRF, use the command:

`vrf NAME`

When executing this command, a new VRF will be created and you will enter its configuration mode.

After creating a VRF, you need to bind the interfaces which traffic you want to route in this VRF. This is done using the *`vrf VRFNAME`* command in the interface configuration mode.

7.1.1 Show Commands:

`show vrf NAME`

Display the information about created VRF (its name and index number). Index number is useful for searching this VRF in the NAT logs.

`show vrf NAME counters [<ip|ipv6>]`

Display counters for specified VRF. Counters for IPv4 and IPv6 are the same.

Counters	Description
`Fragment No Memory Drops`	Fragmentation is required when the IP packet size exceeds the MTU value of a network interface. The IP fragmentation is proceeded at the packet sending stage. vCGNAT creates N new packets, which are taken from the packet pool. If there are not enough packets, then we drop the whole packet and add 1 to the counter. To see packet-pool load, use the command `show debugging packets-pools`
`Packet Too Big Drops`	The IP packet size is exceeded MTU of the interface and the fragmentation cannot be done. For IPv4 case the fragmentation is not performed if DF flag (Don't fragment" flag) is set. For IPv6 the fragmentation is not performed at all. The exception is NAT64. If fragmentation is needed after address translation from IPv4 to IPv6, it will be done if the DF flag was not set
`TTL Drops`	Time to live is expired which indicates a loop in your network or TTL value of the IP packet is set very low
`No Neighbor Drops`	Failed to find next-hop MAC address in ARP table
`No Route Drops`	Cannot find the path to the destination host in the route table
`No Source Address Drops`	The source IPv4/IPv6 node address field cannot be filled during packet generation. These types of errors occur only for locally generated packets such NAT logs or ICMP errors
`Invalid Packet Length Drops`	Invalid IPv4/IPv6 header length

`clear vrf NAME counters [<ip|ipv6>]`

Clear all VRF counters. You can specify counters for IPv4 or IPv6 to be cleared.

7.2 Static Routing

NFWare vCGNAT can be used as a router if NAT is not configured on interfaces. You can add static routes for your purposes, for example, to define a default route, for routing redistribution, etc. To configure the static route, use the command in configuration mode:

`ip route`

with the following keys:

Keys	Description
`A.B.C.D/M`	IP destination with short prefix (e.g. 10.0.0.0/8)
`A.B.C.D`	IP destination with long prefix (e.g. 10.0.0.0 255.0.0.0)
After specifying the destination, choose the following parameters:	
`INTERFA`	The name of the interface to use as next-hop
`A.B.C.D`	The IP address to use as next-hop
`Null0`	The purpose of configuring a static summary route to null0 is to ensure that traffic will be dropped on the local router if a more specific route does not exist.
`blackho`	It is used silently to discard unwanted or undesirable traffic
`reject`	Create an unreachable route that rejects traffic with ICMP "Destination Unreachable" messages
After specifying the obligatory parameters, you can use additional keys:	
`(1-255)`	Distance value for this route – is a metric used by routers to choose the best path. If you want the dynamic route to override the static one, specify this distance greater than for the dynamic route
`nexthop`	Allows to create a leaked route with a nexthop in the specified vrf
`tag`	Set tag for this route. It is a 32-bit value attached to routes. Route tags are used to filter routes and apply administrative policies
`onlink`	When both `INTERFACE` and `A.B.C.D` are specified, it binds the route to the specified interface. In this case, it is also possible to specify onlink to force the routing subsystem to consider the next-hop as "on link" on the given interface. For example, if you want to add a route to the network `192.168.2.0/30` via the `if3` and GW does not belong to the connected network on the interface if3, you should specify the command `ip route 192.168.2.0/30 1.1.1.1 if3 onlink`

To add a static route in the specified VRF, first enter it via the command `vrf NAME` and then use `ip route`.

`show <ip|ipv6> route fib [summary] [vrf NAME]`

Display Forwarding Information Base table. If VRF is configured, you can specify its name. Specifying the key `summary` shows the total number of routes. There are two planes in the architecture of the router:

1. RIB (Routing Information Base) – receives all routes from all routing protocols. Routes are assigned different Administrative Distances using special algorithms, and the route with the smallest Administrative Distance is considered the best and stored in FIB

2. FIB (Forwarding Information Base) – is used for forwarding IP packets. Here, the route with the longest prefix matching the destination address in the IP packet is considered the best.

`show <ip|ipv6> route`

Display IP routing table.

7.3 ARP

To configure state timeouts for Neighbor Cache entry, use the following commands. Three states are used in the vCGNAT ARP-table: INCOMPLETE, REACHABLE and STALE (see **RFC 4861#section-7.3.2**):

`ip neighbor timeout reachable (1-604800000)`

Set timeout for REACHABLE state in milliseconds. The set value is multiplied by a random number in the range from 0.5 to 1.5. The default is `30000(ms) x random(0.5;1.5)`.

`ip neighbor timeout stale (1-604800000)`

Set timeout for STALE state in milliseconds. The default is 30000 ms.

To configure the period during which the router will send ARP requests to the neighbor, use the following command:

`ip neighbor timeout retransmit (1-604800000)`

Static ARP

Sometimes, it is necessary to configure a static entry in the ARP table. To do that, use the command:

`<ip|ipv6> neighbor <A.B.C.D|X:X::X:X> interface NAME mac X:X:X:X:X:X`

Set a static entry for ARP table for IPv4 or IPv6.

`show <ip|ipv6> neighbor`

Display ARP table.

```
vCGNAT01# show ip neighbor
------------------------------------------------------------------------------------
↪-------------------------
Neighbor                                        State       Interface              ␣
↪         MAC address
------------------------------------------------------------------------------------
↪-------------------------
172.23.40.50                                    Permanent   if2                    ␣
↪         0C:C4:83:76:00:03
------------------------------------------------------------------------------------
↪-------------------------
```

CHAPTER

EIGHT

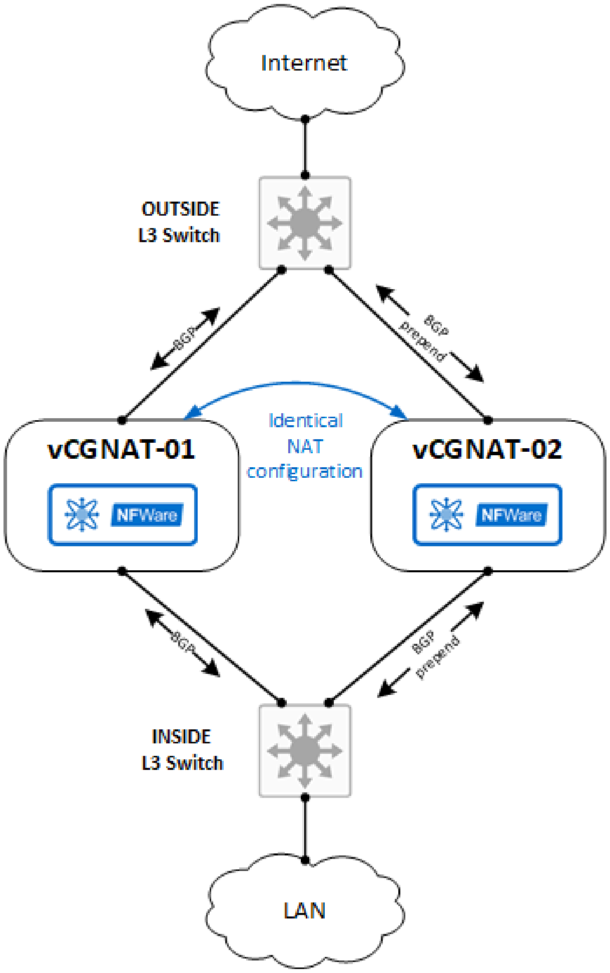
HIGH AVAILABILITY

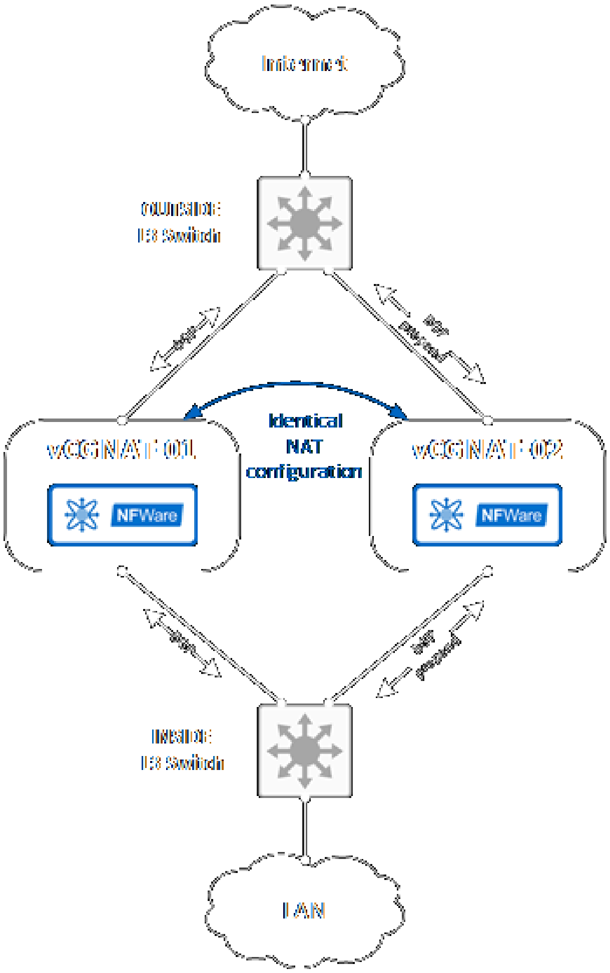
8.1 vCGNAT Deployment Scenarios

vCGNAT supports the scenarios below that involve integration into an existing network in order to guarantee redundancy and scalability:

1. Active/Standby
2. Active/Active (N+1)

8.1.1 Active/Standby





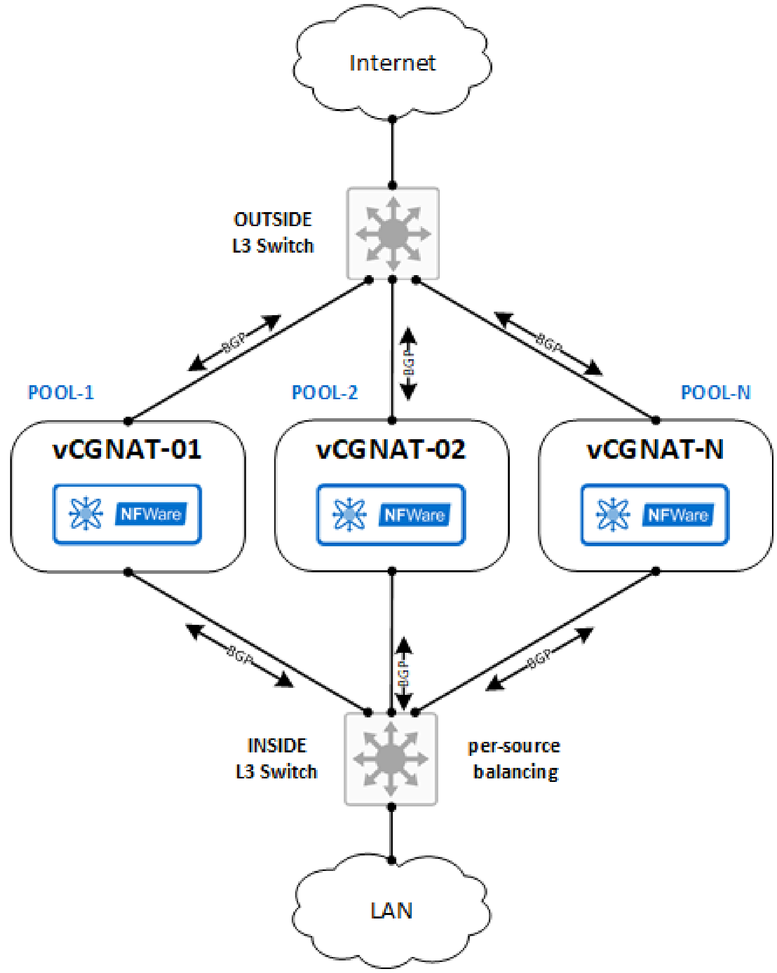
In an Active/Standby scenario, each vCGNAT node has the same NAT configuration. This same NAT configuration refers to public pool addresses, subscriber groups and rules settings.

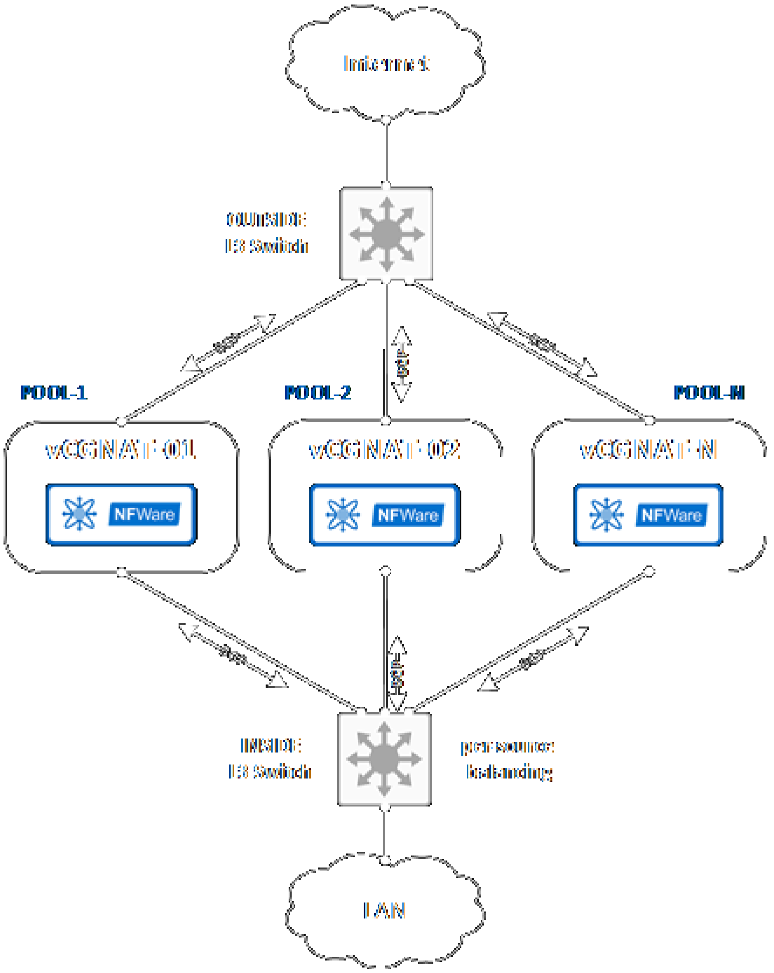
Each vCGNAT is active and ready for operation. Neighbor devices: L3 switches or routers decide which vCGNAT is used for traffic processing. Routing protocols are used for traffic path selection, and the most commonly used protocol is Border Gateway Protocol (BGP).

The AS-path prepend option is used when routing updates pass through the second vCGNAT. The routing updates from the first vCGNAT receive a higher priority, and traffic goes through it.

In case of a failure with the first vCGNAT, the routing tables on the neighbor devices are rebuilt, and the path through the second vCGNAT is used. The Bidirectional Forwarding Detection (BFD) protocol reduces node failure response time in conjunction with BGP.

8.1.2 Active/Active (N+1)





In an Active/Active (N+1) scenario, each vCGNAT node has a different configuration. Each vCGNAT is active. The NAT rules are created so that each vCGNAT can handle traffic from any subscriber inside the network, but different public pools are used, i.e., the traffic is balanced. To balance inside network (LAN) traffic, per-source balancing on the downlink L3 device is used. Neighboring devices run BGP for the announcement of routing updates. Each vCGNAT announces its subnet which corresponds to a public address pool for NAT translations. Therefore, reverse traffic from the internet always returns to the same vCGNAT node as traffic from the inside network.

8.2 VRRP

VRRP (Virtual Router Redundancy Protocol) is a computer network protocol designed to increase the availability of routers acting as a default gateway. This is achieved by combining a group of routers into one virtual router and assigning them a common IP address which will be used in the same L2 network. vCGNAT implements VRRP v3 (**RFC 5798**).

8.2.1 Definitions

VRRP Router	A router on which the VRRP is run. It can participate in one or more virtual routers
Virtual Router	An abstract object managed by VRRP acts as the host's default router. In fact, a virtual router is a group of router interfaces on the same network that share a Virtual Router Identifier (VRID) and a virtual IP address
Virtual IP address	It is an IP address is selected from the set of real interface addresses. VRRP advertisements are always sent using the primary IP address as the source of the IP packet
IP Address Owner	It is a VRRP router that uses the IP address assigned to the virtual router as the real IP address
Virtual Router Master	It is a VRRP router responsible for forwarding packets sent to the IP address associated with the virtual router and responding to ARP requests sent to that address. If the IP address owner is available, then it always becomes the Master
Virtual Router Backup	It is a group of idle routers that are ready to take over the role of Virtual Router Master as soon as the current VRRP Master router becomes unavailable
Virtual MAC	The virtual MAC address `0000:5E00:01xx`, where `xx` is the VRRP group number

8.2.2 Configuring

To configure VRRP, enter the data interface configuration mode and use the command:

`vrrp (1-255)`

where `(1-255)` is a VRID.

The following keys are available:

Key	Argument	Description
`ip`	`A.B.C.D`	Set Virtual Router IPv4 address
`ipv6`	`X:X:`	Set Virtual Router IPv6 address
`preemp`		Preempt mode controls whether the Backup router with a higher priority will try to take over the Master role from the current Master router with a lower priority. The exception is that a VRRP router will always become Master if it is an IP Address Owner. By default, this mode is disabled. Enter this command to enable it
`preemp delay minimu`	`1-65`	Delay in seconds between Backup router with higher priority goes to backup state and starts to sent VRRP advertisements to become Master
`priori`	`(1-2`	Set the priority for this VRRP router that will be used to select a VRRP Master. Higher values equal higher priority. The value `255` is reserved for the router that owns the IP address associated with the virtual router. The value 0 is used when the current Master router should be stopped from participating in VRRP. Backup routers will then begin selecting the Master without waiting for the current Master to timeout. The default value is 100
`timers advert`	`(1-4`	The interval (in seconds) between sending VRRP advertisements. The default setting is 1 second

8.2.3 Show Commands

`show vrrp (1-255) counters [ip|ipv6]`

Display VRRP counters for IPv4 or IPv6 protocol. The output will be as follows:

```
Router1# show vrrp 1 counters
------------------------------------------------------------
Counter                                 Value
------------------------------------------------------------
VRRP advertisement packets sent         14336
VRRP advertisement packets received     0
Transitions to INIT state               1
Transitions to BACKUP state             1
Transitions to MASTER state             1
VRRP priority 0 adv packets sent        0
Allocation packet buffer drops          0
Sending nd announce packet drops        0
------------------------------------------------------------
```

Note: if you have several VRRP Routers and do not point out the VRID, then the counters will be summarized.

`clear vrrp (1-255) counters [ip|ipv6]`

Clear VRRP counters for IPv4 or IPv6 protocol.

`show vrrp (1-255)`

Display various information about VRRP Router:

```
if1 - Group 1 - Address-Family IPv4
State is MASTER
State duration 28 mins 34.826 secs
Virtual IP address is 192.168.3.14
Virtual MAC address is 00:00:5E:00:01:01
Advertisement interval is 100 csec
Preemption enabled
Priority is 100
Master Router is 192.168.3.4, priority is 100
Master Advertisement interval is 100 csec (expires in 67 csec)
```

8.3 Real Time Management System

The Real Time Management System (RTM) is a real-time event reaction system. Events are created and described using track. The track can now be created for VRRP instances and routes from the FIB. Events are described using a sequence of CLI commands and/or scripts, the path to which can be specified in the RTM settings.

Using this subsystem, you can logically link dynamic routing protocols with VRRP. For example:

- When a route, received through BGP (or any other), disappears from the FIB, we can lower the priority of VRRP.
- When VRRP Master switches to backup state (for example, VRRP fails, but the router is available itself and continues to announce routes), you can add the prefix-list with `as-prepend` to the BGP neighbour, thus lowering the priority of its routes.

To create tracks and configure RTM, use following commands:

track ID <ip|ipv6> route IP/ MASK ecmp-number <equal|greater-equal|less-equal> N [vrf NAME]

Create track associated with the route and specify the trigger condition by setting the number of the nexthops.

track ID vrrp VRID interface NAME ip[v6] state <backup|master>

Create track associated with VRID and specify the trigger condition by setting the state.

no track ID

Delete track with the specified ID.

rtm cli-policy ID

Create Real Time Management policy. After creating specify following rules:

event track ID state negative|positive

Set the condition under which the actions will apply.

action ID cli COMMAND

Set an action ID with cli-command to run. Pay attention, first you need to enter the configuration mode.

action ID script PATH

Set an action ID with a path to the script with the commands to run.

no action ID

Delete the action with the specified ID. If you want to change the action, first you need to delete the old one.

8.4 Synchronization

Session synchronization is used to ensure that all vCGNAT instances in the cluster have complete information about all open ports and sessions. In this case, if one vCGNAT instance fails, user sessions are not dropped when traffic moves to another instance. Failure of one or more (depending on the selected network architecture) vCGNAT instances will pass unnoticed by the client. Data exchange is performed over the UDP protocol.

8.4.1 onfiguration

All the settings described below are performed in the configuration mode. Session synchronization should be configured at least on two vCGNATs; otherwise, the messages sent by the first vCGNAT will be dropped by the second one.

nat sync destination-ip A.B.C.D port (1-65535) [{vrf NAME|source-ip A.B.C.D}]

Set the destination IP address and port to which data will be sent via UDP protocol. Additionally, VRF and the source IP address can be specified. To send Synchronization messages to multiple devices at once, use a dedicated L2 segment and a broadcast network address as the destination IP address.

no nat sync

Disable created synchronization

<nat|nat64> sync timeout-delay (1-604800000)

Set a delay interval (in milliseconds) that is added to the timers when sessions are created via sync messages. The lifetime of the sessions will be `timeout-delay` ms longer than on the server where they are created. The delay is needed so that the primary server has time to send us a message with updates on a given session before we delete it because of recreating a session has much overhead. The default is 1000 ms.

no <nat|nat64> sync timeout-delay (1-604800000) [vrf NAME]

Reset timeout delay to the default value.

nat sync start-delay (1-3600)

Set start delay in seconds. This delay is necessary for the Active-Active redundancy scenario. It is used at system startup and is needed to add a new instance to the cluster transparently. NAT instance will not respond to ARP request during this delay, respectively traffic will not be redistributed to it. At the same time, other devices will send synchronization messages and this new instance will fill its session table. When delay ends, the new instance, with already full table of sessions, will pick up some traffic without losing users' connections.

Important! In this case, synchronization must be configured to the broadcast address of the network, because ARP during this delay will not respond.

no nat sync start-delay

Disable start delay

If you do not want to wait for the start delay, use this command (in View mode) to start traffic processing immediately:

```
nat sync start
```

8.4.2 Show commands

show <nat|nat64> counters [vrf NAME] sync

Display NAT or NAT64 counters for synchronization. See *Show and Clear Commands* for their description.

```
nfware# show nat counters sync
------------------------------------------------------------
Counter                                 Value
------------------------------------------------------------
Subscriber Creations                    0
Address Map Creations                   0
Port Map Creations                      0
Session Creations                       0
Hairpinning Sessions                    0
Address Map Entry Limit Drops           0
Address Map Failure Drops               0
No IP Suggested Drops                   0
Out of Synchronization Drops            0
Deletion of Non-Existent Session        0
Outdate Messages                        0
No NAT Rule Drops                       0
No NAT Rule Group Drops                 0
No Pool Drops                           0
Port Map Entry Limit Drops              0
Port Map Failure Drops                  0
Session Entry Limit Drops               0
Subscriber Limit Drops                  0
Subscriber Session Limit Drops          0
Subscriber Session Rate Limit Drops     0
------------------------------------------------------------
```

clear <nat|nat64> counters [vrf NAME] sync

Clear NAT or NAT64 counters for synchronization.

show nat sync

Display counters for synchronization.

```
nfware# show nat sync
nat sync destination-ip 172.23.40.50 port 10 source-ip 172.23.40.49
Counters:
 Sent:
  Packets:  7
  Messages: 7
  No packet drops: 0
  No VRF drops: 0
 Received:
  Packets:  0
  Messages: 0
  Invalid msg length:   0
  Errors on processing: 0
  Unsupported msg type: 0
```

clear nat sync

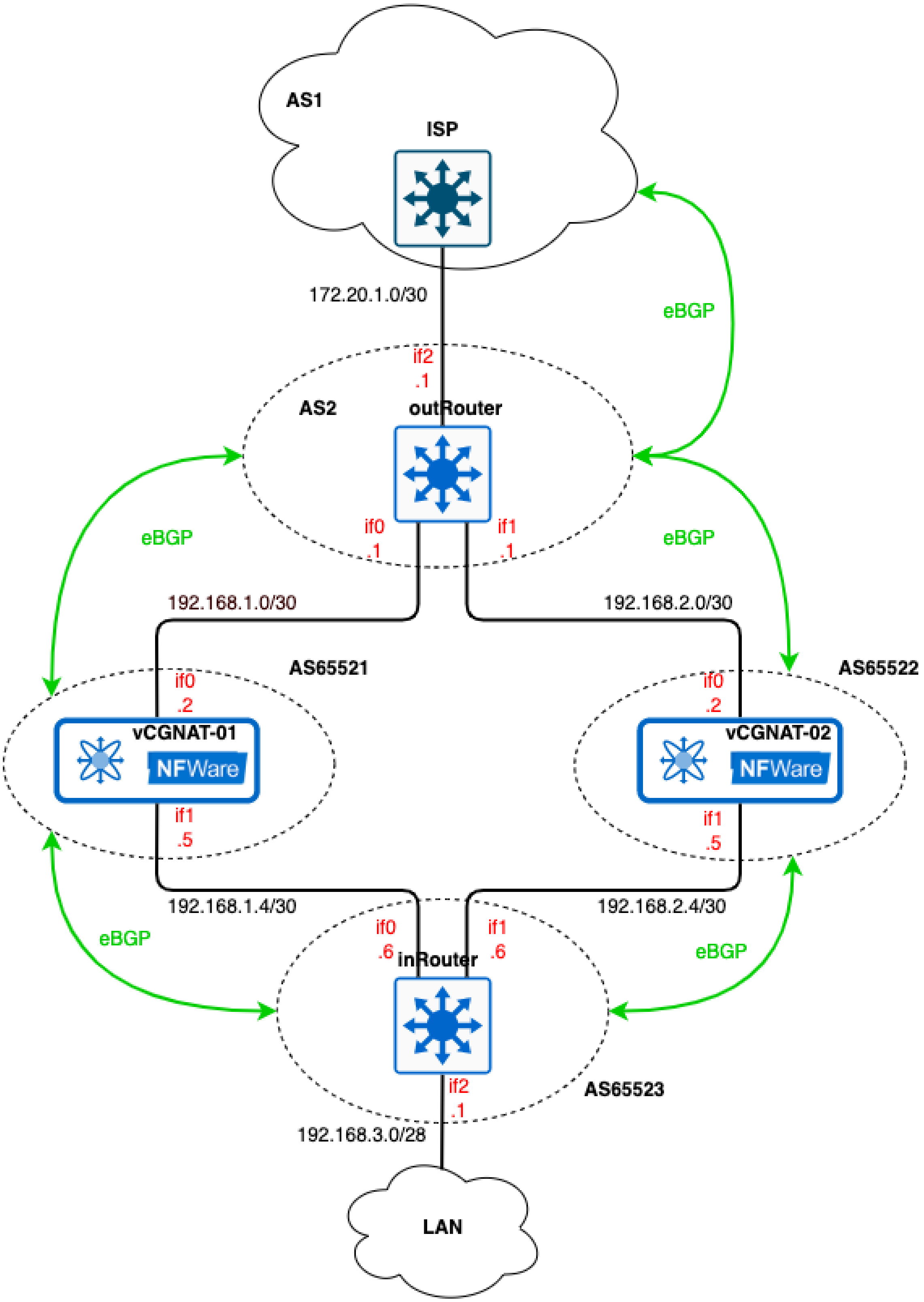
Clear counters for synchronization.

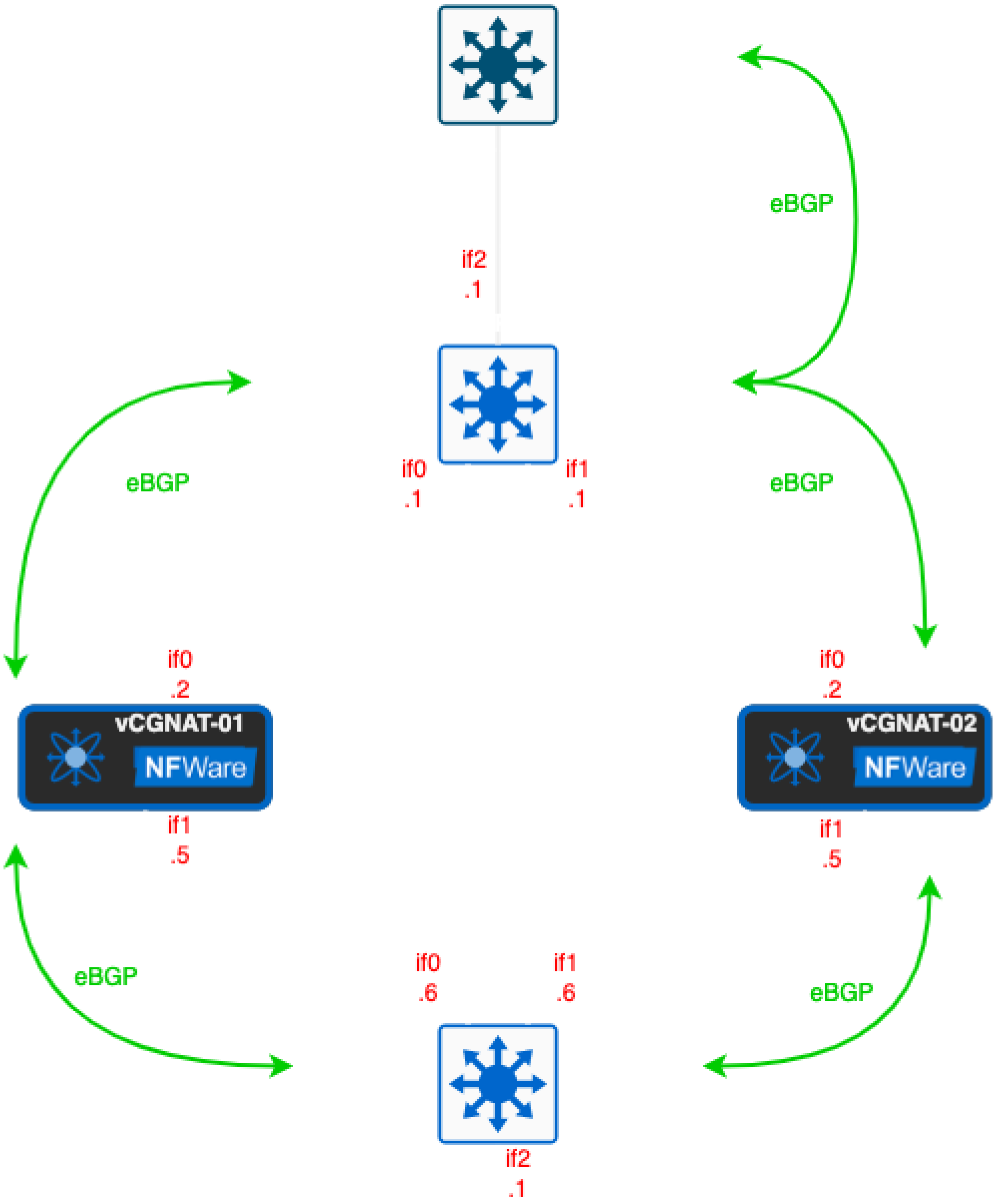
8.5 Typical Configuration

8.5.1 BGP

8.5.1.1 Description

The configuration is done according to the Active/Standby mode (see the section *vCGNAT Deployment Scenarios* for more information), where each vCGNAT is active and ready for operation. Neighbor devices: L3 switches or routers determine which vCGNAT to use for traffic processing. Border Gateway Protocol (BGP) is used for traffic path selection. The figure below shows a simple scheme where the border router is connected to ISP. Suppose the provider advertise us default path 0.0.0.0/0 through BGP. Public AS numbers are taken as an example. Note that this simple diagram topology is intended to show the use of the vCGNAT together with the BGP routing protocol. In real case, more complex network configuration is applied with VRFs, link aggregation, and several border routers, each connected to different ISPs.


AS1
ISP
172.20.1.0/30
eBGP
if2
.1
AS2
outRouter
eBGP
if0
.1
if1
.1
eBGP
192.168.1.0/30
192.168.2.0/30
AS65521
AS65522
if0
.2
if0
.2
vCGNAT-01
NFWare
vCGNAT-02
NFWare
if1
.5
if1
.5
192.168.1.4/30
192.168.2.4/30
if0
.6
if1
.6
eBGP
inRouter
eBGP
if2
.1
AS65523
192.168.3.0/28
LAN


if2
.1
eBGP
eBGP
eBGP
if0
.1
if1
.1
if0
.2
vCGNAT-01
NFWare
if1
.5
if0
.2
vCGNAT-02
NFWare
if1
.5
eBGP
eBGP
if0
.6
if1
.6
if2
.1

8.5.1.2 Configuration

outRouter

```
!
hostname outRouter
!
interface if0
 ip address 192.168.1.1/30
!
interface if1
 ip address 192.168.2.1/30
!
interface if2
 ip address 172.20.1.1/30
!
interface if3
!
interface lo
!
interface management
!
router bgp 2
 bgp log-neighbor-changes
 neighbor 172.20.1.2 remote-as 1
 neighbor 172.20.1.2 bfd
 neighbor 192.168.1.2 remote-as 65521
 neighbor 192.168.1.2 bfd
 neighbor 192.168.2.2 remote-as 65522
 neighbor 192.168.2.2 bfd
 !
 address-family ipv4 unicast
  neighbor 172.20.1.2 remove-private-AS
  neighbor 172.20.1.2 soft-reconfiguration inbound
  neighbor 172.20.1.2 prefix-list ALLOW_ALL in
  neighbor 172.20.1.2 prefix-list DENY_PRIVATE out
  neighbor 192.168.1.2 soft-reconfiguration inbound
  neighbor 192.168.1.2 prefix-list ALLOW_ALL in
  neighbor 192.168.1.2 prefix-list ALLOW_ALL out
  neighbor 192.168.2.2 soft-reconfiguration inbound
  neighbor 192.168.2.2 prefix-list ALLOW_ALL in
  neighbor 192.168.2.2 prefix-list ALLOW_ALL out
 exit-address-family
!
ip prefix-list DENY_PRIVATE seq 5 deny 192.168.0.0/16 ge 16 le 32
ip prefix-list DENY_PRIVATE seq 10 permit any
ip prefix-list ALLOW_ALL seq 5 permit any
!
end
```

1. See the section *Data Interfaces* to configure interfaces.
2. To configure neighborhood and switch on BFD (Bidirectional Forwarding Detection), use the commands `neighbor <IP-address of the neighbor router> remote-as <AS number>` and `neighbor <IP-address of the neighbor router> bfd`.

```
router bgp 2
 bgp log-neighbor-changes
 neighbor 172.20.1.2 remote-as 1
 neighbor 172.20.1.2 bfd
 neighbor 192.168.1.2 remote-as 65521
 neighbor 192.168.1.2 bfd
 neighbor 192.168.2.2 remote-as 65522
 neighbor 192.168.2.2 bfd
```

3. To allow some networks to be announced to neighbors and to deny others, create rules via prefix-list. Here, two prefix-lists are applied: DENY_PRIVATE prohibits the advertisement of our private networks to the outside and allows the advertisement of others to external routers, and ALLOW_ALL allows the advertisement of the networks to our internal neighbors.

```
ip prefix-list DENY_PRIVATE seq 5 deny 192.168.0.0/16 ge 16 le 32
ip prefix-list DENY_PRIVATE seq 10 permit any
ip prefix-list ALLOW_ALL seq 5 permit any
```

4. To apply the prefix-lists, use the command `neighbor neighbor_IP prefix-list prefix_list_name in or out`. The words `in` and `out` mean applying the rule to incoming and outgoing network advertisement, respectively.

```
address-family ipv4 unicast
  network 172.20.1.0/30
  network 172.20.1.4/30
  neighbor 172.20.1.2 remove-private-AS
  neighbor 172.20.1.2 soft-reconfiguration inbound
  neighbor 172.20.1.2 prefix-list DENY_PRIVATE out
  neighbor 172.20.1.6 remove-private-AS
  neighbor 172.20.1.6 soft-reconfiguration inbound
  neighbor 172.20.1.6 prefix-list DENY_PRIVATE out
  neighbor 192.168.1.2 soft-reconfiguration inbound
  neighbor 192.168.1.2 prefix-list ALLOW_ALL in
  neighbor 192.168.1.2 prefix-list ALLOW_ALL out
  neighbor 192.168.2.2 soft-reconfiguration inbound
  neighbor 192.168.2.2 prefix-list ALLOW_ALL in
  neighbor 192.168.2.2 prefix-list ALLOW_ALL out
exit-address-family
!
```

4. As we do not want to advertise our private AS numbers to the Internet, we use command `neighbor neighbor_IP remove-private-as` to remove them from AS-path. Please note, if the AS-path has a mix of public and private AS numbers, then the router will not remove anything.

inRouter

The inRouter will have the similar configuration except DENY_PRIVATE prefix-list:

```
!
hostname inRouter
```

(continues on next page)

(continued from previous page)

```
!
interface if0
 ip address 192.168.1.6/30
!
interface if1
 ip address 192.168.2.6/30
!
interface if2
 ip address 192.168.3.1/28
!
interface lo
!
interface management
!
router bgp 65523
 bgp log-neighbor-changes
 neighbor 192.168.1.5 remote-as 65521
 neighbor 192.168.1.5 bfd
 neighbor 192.168.2.5 remote-as 65522
 neighbor 192.168.2.5 bfd
 !
 address-family ipv4 unicast
  network 192.168.3.0/28
  neighbor 192.168.1.5 soft-reconfiguration inbound
  neighbor 192.168.1.5 prefix-list ALLOW_ALL in
  neighbor 192.168.1.5 prefix-list ALLOW_ALL out
  neighbor 192.168.2.5 soft-reconfiguration inbound
  neighbor 192.168.2.5 prefix-list ALLOW_ALL in
  neighbor 192.168.2.5 prefix-list ALLOW_ALL out
 exit-address-family
!
ip prefix-list ALLOW_ALL seq 5 permit any
!
```

vCGNAT01

```
!
hostname vCGNAT01
!
ip route 203.0.113.0/25 Null0
!
interface if0
 ip address 192.168.1.2/30
 ip nat outside
!
interface if1
 ip address 192.168.1.5/30
 ip nat inside
```

(continues on next page)

(continued from previous page)

```
!
interface if2
!
interface lo
!
interface management
!
router bgp 65521
 bgp log-neighbor-changes
 neighbor 192.168.1.1 remote-as 2
 neighbor 192.168.1.1 bfd
 neighbor 192.168.1.6 remote-as 65523
 neighbor 192.168.1.6 bfd
 !
 address-family ipv4 unicast
  network 192.168.1.0/30
  network 192.168.1.4/30
  network 203.0.113.0/25
  neighbor 192.168.1.1 soft-reconfiguration inbound
  neighbor 192.168.1.1 prefix-list ALLOW_ALL in
  neighbor 192.168.1.1 prefix-list ALLOW_ONLY out
  neighbor 192.168.1.6 soft-reconfiguration inbound
  neighbor 192.168.1.6 prefix-list ALLOW_ALL in
  neighbor 192.168.1.6 prefix-list IN_POLICY out
 exit-address-family
!
ip prefix-list ALLOW_ONLY seq 10 permit 203.0.113.0/25
ip prefix-list IN_POLICY seq 5 deny 203.0.113.0/25
ip prefix-list IN_POLICY seq 10 permit any
ip prefix-list ALLOW_ALL seq 5 permit any
!
nat pool default-pool
 range 203.0.113.1 203.0.113.127
 enable
!
nat subscriber-group default-group
 pool default-pool
!
nat rule subnet 192.168.3.0/28 subscriber-group default-group
!
end
```

1. See the section *Typical Configuration* to configure NAT.
2. The network 203.0.113.0/25 must have the exact route, otherwise it will not be added to the BGP table - this is a prerequisite. To bypass this restriction, traffic from 203.0.113.0/25 must go to Null0 `ip route 203.0.113.0/25 Null0`. This route means that all packets to this subnet will be discarded. However, normal operation will not be disturbed. If there are more exact routes (with a mask greater than /23), they will be preferred according to the Longest Prefix Match rule.
3. Here we use three prefix-lists to configure network advertisement to neighbor:
   - ALLOW_ONLY to advertise only 203.0.113.0/25 network to the outRouter. The pool of the IP addresses from this network will be used for mapping.

- IN_POLICY to deny advertisement 203.0.113.0/25 to our internal network and permit all others.
- ALLOW_ALL to accept all networks from the outRouter and the inRouter.

vCGNAT02

The configuration for vCGNAT02 is almost identical:

```
!
hostname vCGNAT02
!
ip route 203.0.113.0/25 Null0
!
interface if0
 ip address 192.168.2.2/30
 ip nat outside
!
interface if1
 ip address 192.168.2.5/30
 ip nat inside
!
interface lo
!
interface management
!
router bgp 65522
 bgp log-neighbor-changes
 neighbor 192.168.2.1 remote-as 2
 neighbor 192.168.2.1 bfd
 neighbor 192.168.2.6 remote-as 65523
 neighbor 192.168.2.6 bfd
 !
 address-family ipv4 unicast
  network 192.168.2.0/30
  network 192.168.2.4/30
  network 203.0.113.0/25
  neighbor 192.168.2.1 soft-reconfiguration inbound
  neighbor 192.168.2.1 prefix-list ALLOW_ALL in
  neighbor 192.168.2.1 prefix-list ALLOW_ONLY out
  neighbor 192.168.2.1 route-map AS_PATH_PREP out
  neighbor 192.168.2.6 soft-reconfiguration inbound
  neighbor 192.168.2.6 prefix-list ALLOW_ALL in
  neighbor 192.168.2.6 prefix-list IN_POLICY out
  neighbor 192.168.2.6 route-map AS_PATH_PREP out
 exit-address-family
!
ip prefix-list ALLOW_ONLY seq 10 permit 203.0.113.0/25
ip prefix-list IN_POLICY seq 5 deny 203.0.113.0/25
ip prefix-list IN_POLICY seq 10 permit any
ip prefix-list ALLOW_ALL seq 5 permit any
```

(continues on next page)

(continued from previous page)

```
!
route-map AS_PATH_PREP permit 10
 set as-path prepend 65522
!
nat pool default-pool
 range 203.0.113.1 203.0.113.127
 enable
!
nat subscriber-group default-group
 pool default-pool
!
nat rule subnet 192.168.3.0/28 subscriber-group default-group
!
end
```

except for one thing - we want all traffic to go through the vCGNAT01, and the vCGNAT02 should be in the standby mode. If something goes wrong with the vCGNAT01 or route, the traffic will go through the vCGNAT02. For that, use AS-Path prepend mechanism which will make the route through the vCGNAT02 longer by adding an extra hop:

1. First, create the rule AS_PATH_PREP `route-map AS_PATH_PREP permit 10` and set up the as-path added by extra hop `route-map AS_PATH_PREP permit 10`.
2. Then apply this rule to the neighbors: `neighbor 192.168.2.1 route-map AS_PATH_PREP out` and `neighbor 192.168.2.6 route-map AS_PATH_PREP out`.

8.5.1.3 Verification

To check if BGP is configured correctly and path selection works, use the following commands:

- `show ip bgp` Display information about BGP routes in the BGP routing table.
- `show ip bgp neighbors neighbor_IP advertised-routes` Display the routes advertised to the neighbor.
- `show ip bgp neighbors neighbor_IP received-routes` Display the routes received from the neighbor.
- `show ip route` Display the current state of the routing table. Use this command to check connectivity between all hosts on the network.

Now, if we look at what route the outRouter receives from the vCGNAT02, we will see an extra hop 65522:

```
outRouter# show ip bgp neighbors 192.168.2.2 received-routes
BGP table version is 4, local router ID is 192.168.2.1, vrf id 0
Default local pref 100, local AS 2
Status codes:  s suppressed, d damped, h history, * valid, > best, = multipath,
               i internal, r RIB-failure, S Stale, R Removed
Nexthop codes: @NNN nexthop's vrf id, < announce-nh-self
Origin codes:  i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

   Network          Next Hop            Metric LocPrf Weight Path
*> 203.0.113.0/25   192.168.2.2              0             0 65522 65522 i

Total number of prefixes 1
```

Additionally, traffic will pass through the vCGNAT01 because this route is shorter than through the vCGNAT02:

```
inRouter# show ip bgp neighbors 192.168.2.5 received-routes
BGP table version is 10, local router ID is 192.168.3.1, vrf id 0
Default local pref 100, local AS 65523
Status codes:  s suppressed, d damped, h history, * valid, > best, = multipath,
               i internal, r RIB-failure, S Stale, R Removed
Nexthop codes: @NNN nexthop's vrf id, < announce-nh-self
Origin codes:  i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

   Network          Next Hop            Metric LocPrf Weight Path
*> 0.0.0.0/0        192.168.2.5                            0 65522 65522 2 1 i
*> 192.168.1.0/30   192.168.2.5                            0 65522 65522 65523 65521 i
*> 192.168.1.4/30   192.168.2.5                            0 65522 65522 65523 65521 i
*> 192.168.2.0/30   192.168.2.5              0             0 65522 65522 i
*> 192.168.2.4/30   192.168.2.5              0             0 65522 65522 i
*> 192.168.3.0/28   192.168.2.5                            0 65522 65522 65523 i

Total number of prefixes 6
```

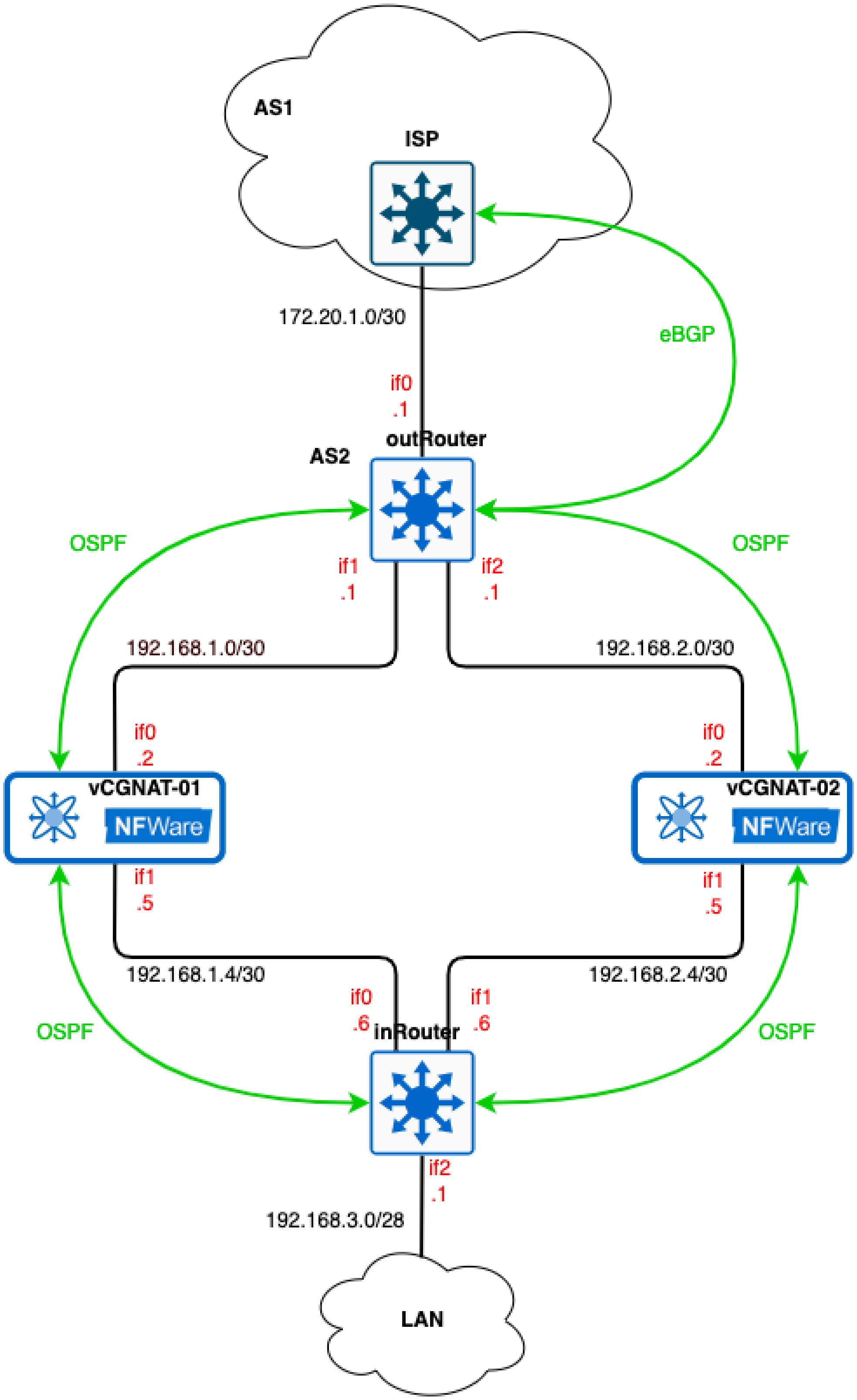
Make sure the ISP does not receive any private ASN and IP addresses:

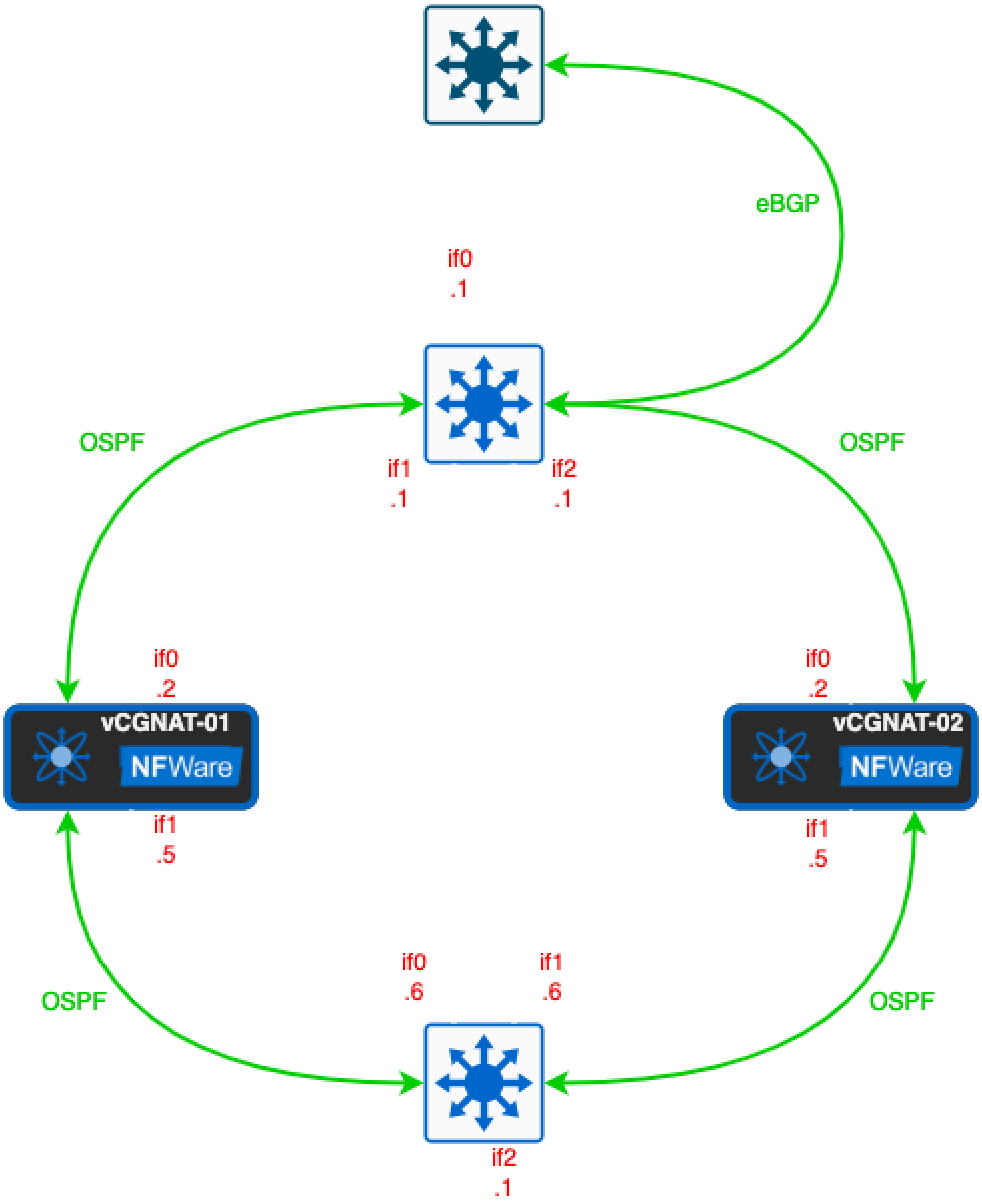
```
ISP# show ip bgp
BGP table version is 1, local router ID is 172.20.1.2, vrf id 0
Default local pref 100, local AS 1
Status codes:  s suppressed, d damped, h history, * valid, > best, = multipath,
               i internal, r RIB-failure, S Stale, R Removed
Nexthop codes: @NNN nexthop's vrf id, < announce-nh-self
Origin codes:  i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

   Network          Next Hop            Metric LocPrf Weight Path
   0.0.0.0/0        0.0.0.0                  0         32768 i
*> 203.0.113.0/25   172.20.1.1                             0 2 i

Displayed  2 routes and 2 total paths

ISP# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, T - Table, v - VNC,
       V - VNC-Direct,
       > - selected route, * - FIB route, q - queued, r - rejected, b - backup
       t - trapped, o - offload failure

C>* 172.20.1.0/30 is directly connected, if0, 00:03:39
B>* 203.0.113.0/25 [20/0] via 172.20.1.1, if0, weight 1, 00:03:35
```

8.5.2 OSPF

8.5.2.1 Description

Below is the description of the routers and vCGNAT (in Active/Standby mode) configurations that intended to be performed if you use the OSPF protocol to distribute routing information throughout the internal network. Suppose the provider advertise us default path 0.0.0.0/0 through BGP. The logic design topology remains the same. BGP is used to distribute routes across the external network.


AS1
ISP
172.20.1.0/30
eBGP
if0
.1
outRouter
AS2
OSPF
OSPF
if1
.1
if2
.1
192.168.1.0/30
192.168.2.0/30
if0
.2
if0
.2
vCGNAT-01
NFWare
vCGNAT-02
NFWare
if1
.5
if1
.5
192.168.1.4/30
192.168.2.4/30
if0
.6
inRouter
if1
.6
OSPF
OSPF
if2
.1
192.168.3.0/28
LAN


eBGP
if0
.1
OSPF
if1
.1
if2
.1
OSPF
if0
.2
vCGNAT-01
NFWare
if1
.5
if0
.2
vCGNAT-02
NFWare
if1
.5
if0
.6
if1
.6
OSPF
OSPF
if2
.1

8.5.2.2 Configuration

outRouter

```
!
hostname outRouter
!
ip router-id 172.16.255.1
!
interface if0
 ip address 172.20.1.1/30
 ip ospf passive
!
interface if1
 ip address 192.168.1.1/30
!
interface if2
 ip address 192.168.2.1/30
 ip ospf cost 100
!
interface if3
!
interface lo
 ip address 172.16.255.1/29
!
interface management
!
router bgp 2
 bgp log-neighbor-changes
 neighbor 172.20.1.2 remote-as 1
 !
 address-family ipv4 unicast
  network 172.20.1.0/30
  network 203.0.113.0/25
  neighbor 172.20.1.2 soft-reconfiguration inbound
  neighbor 172.20.1.2 prefix-list ALLOW_ALL out
 exit-address-family
!
router ospf
 network 192.168.1.0/30 area 0
 network 192.168.2.0/30 area 0
 default-information originate
 exit
!
ip prefix-list ALLOW_ALL seq 5 permit any
```

1. See detailed instruction in the section *Data Interfaces* to configure interfaces.
2. The BGP Routing Process configuration is shown in the section BGP above. Note that since we do not use BGP internally, the public address pool `203.0.113.0/25` must be announced on the border router, not vCGNAT.
3. First, it is necessary to set the router-id. By default, it is the largest address of the loopback interfaces. We will use network 172.16.255.0/29 for that purpose.

```
!
ip router-id 172.16.255.1
!
interface lo
 ip address 172.16.255.1/29
!
```

4. Then, start the OSPF process on the router and announce which networks will be advertised.

```
router ospf
 network 192.168.1.0/30 area 0
 network 192.168.2.0/30 area 0
```

5. To forward the subscribers' traffic to the Internet, distribute the default route throughout the internal network by the command `default-information originate`.

6. To force packets to take one route (through vCGNAT01) and disable ECMP (Equal-cost multi-path routing), increase the route's cost to the vCGNAT02. To do this, use the command `ip ospf cost 100` on the interface that looks towards the vCGNAT02. Use the same command for the inRouter.

inRouter

The configuration is simple and resembles the outRouter:

```
!
hostname inRouter
!
ip router-id 172.16.255.4
!
interface if0
 ip address 192.168.1.6/30
!
interface if1
 ip address 192.168.2.6/30
 ip ospf cost 100
!
interface if2
 ip address 192.168.3.1/30
!
interface lo
 ip address 172.16.255.4/29
!
interface management
!
router ospf
 network 192.168.1.4/30 area 0
 network 192.168.2.4/30 area 0
 network 192.168.3.0/30 area 0
 exit
!
```

vCGNAT01

```
!
hostname vCGNAT01
!
ip router-id 172.16.255.2
ip route 203.0.113.0/25 Null0
!
interface if0
 ip address 192.168.1.2/30
 ip nat outside
!
interface if1
 ip address 192.168.1.5/30
 ip nat inside
!
interface lo
 ip address 172.16.255.2/29
!
interface management
!
router ospf
 redistribute static
 network 192.168.1.0/30 area 0
 network 192.168.1.4/30 area 0
 exit
!
nat pool default-pool
 range 203.0.113.1 203.0.113.127
 enable
!
nat subscriber-group default-group
 pool default-pool
!
nat rule subnet 192.168.3.0/30 subscriber-group default-group
!
```

1. See the detailed instruction in *Typical Configuration* section to configure the NAT.
2. To make the internal devices know about 203.0.113.0/25 network, add a route into Null0 and enable static redistribution.

```
!
ip route 203.0.113.0/25 Null0
!
router ospf
 redistribute static
```

vCGNAT02

The configuration for the vCGNAT02 is the same with appropriate changes to the IP addresses:

```
!
hostname vCGNAT02
!
ip router-id 172.16.255.3
ip route 203.0.113.0/25 Null0
!
interface if0
 ip address 192.168.2.2/30
 ip nat outside
!
interface if1
 ip address 192.168.2.5/30
 ip nat inside
!
interface lo
 ip address 172.16.255.3/29
!
interface management
!
router ospf
 redistribute static
 network 192.168.2.0/30 area 0
 network 192.168.2.4/30 area 0
 exit
!
nat pool default-pool
 range 203.0.113.1 203.0.113.127
 enable
!
nat subscriber-group default-group
 pool default-pool
!
nat rule subnet 192.168.3.0/30 subscriber-group default-group
!
```

8.5.2.3 Verification

If everything has been done correctly, the routes, for example, on the outRouter, will look as follows:

```
outRouter# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, T - Table, v - VNC,
       V - VNC-Direct,
       > - selected route, * - FIB route, q - queued, r - rejected, b - backup
       t - trapped, o - offload failure

S>* 0.0.0.0/0 [1/0] via 172.20.1.2, if0, weight 1, 00:12:19
C>* 172.16.255.0/29 is directly connected, lo, 00:12:20
C>* 172.20.1.0/30 is directly connected, if0, 00:12:20
```

(continues on next page)

(continued from previous page)

```
O   192.168.1.0/30 [110/10] is directly connected, if2, weight 1, 00:12:18
C>* 192.168.1.0/30 is directly connected, if2, 00:12:19
O>* 192.168.1.4/30 [110/20] via 192.168.1.2, if2, weight 1, 00:11:27
O   192.168.2.0/30 [110/100] is directly connected, if3, weight 1, 00:12:19
C>* 192.168.2.0/30 is directly connected, if3, 00:12:19
O>* 192.168.2.4/30 [110/110] via 192.168.2.2, if3, weight 1, 00:11:34
O>* 192.168.3.0/30 [110/30] via 192.168.1.2, if2, weight 1, 00:11:27
O>* 203.0.113.0/25 [110/20] via 192.168.1.2, if2, weight 1, 00:11:26
```

You see a default route to the ISP router and a route to the 203.0.113.0/25 network. Since we have increased the route's cost through the vCGNAT02, only route through the vCGNAT01 got into the routing table. If the link or the vCGNAT01 goes down, a route to the 203.0.113.0/25 network via the vCGNAT02 `O>* 203.0.113.0/25 [110/20] via 192.168.2.2, if2, weight 1,` will be added instead.

8.5.3 VRRP

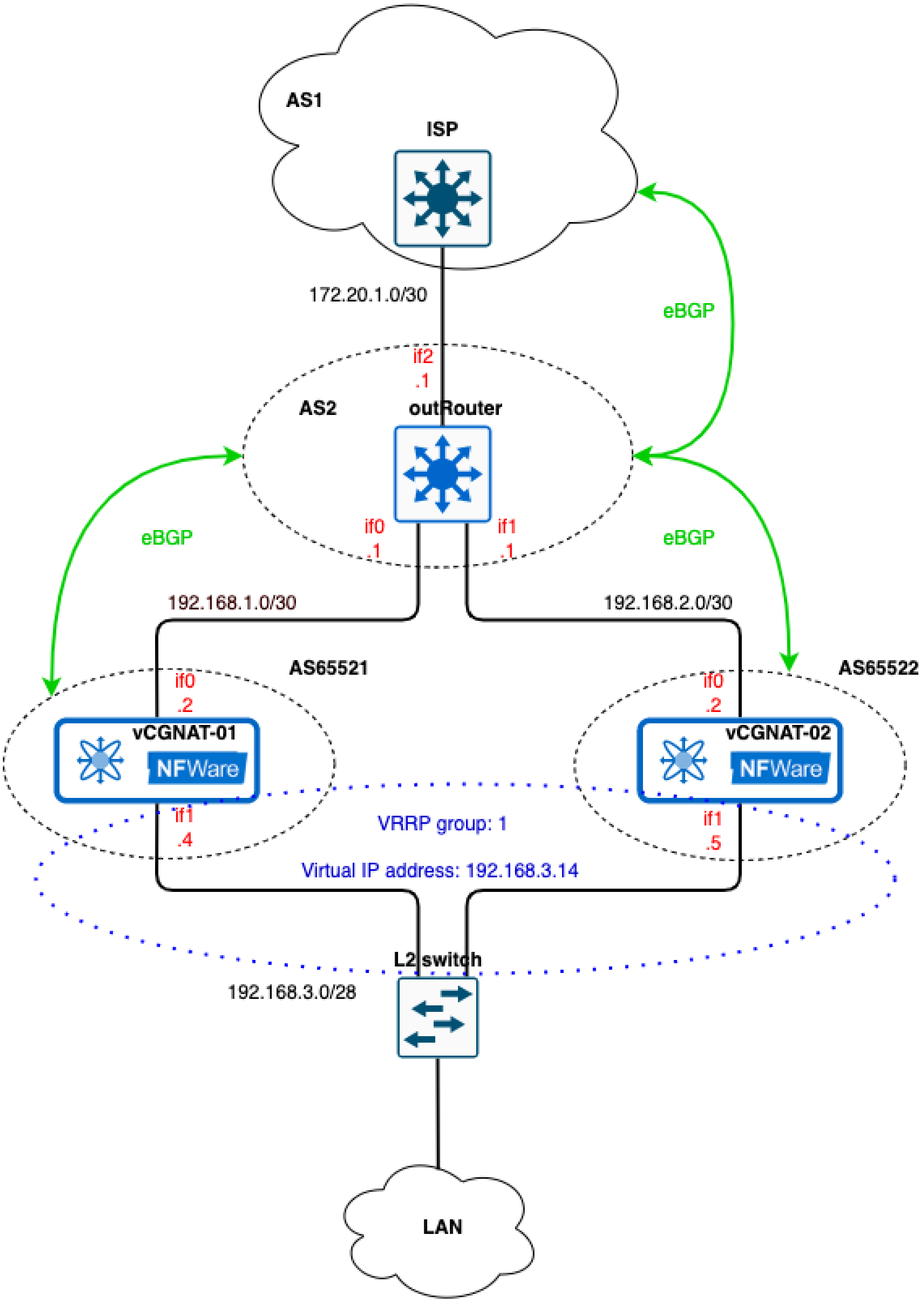
8.5.3.1 Description

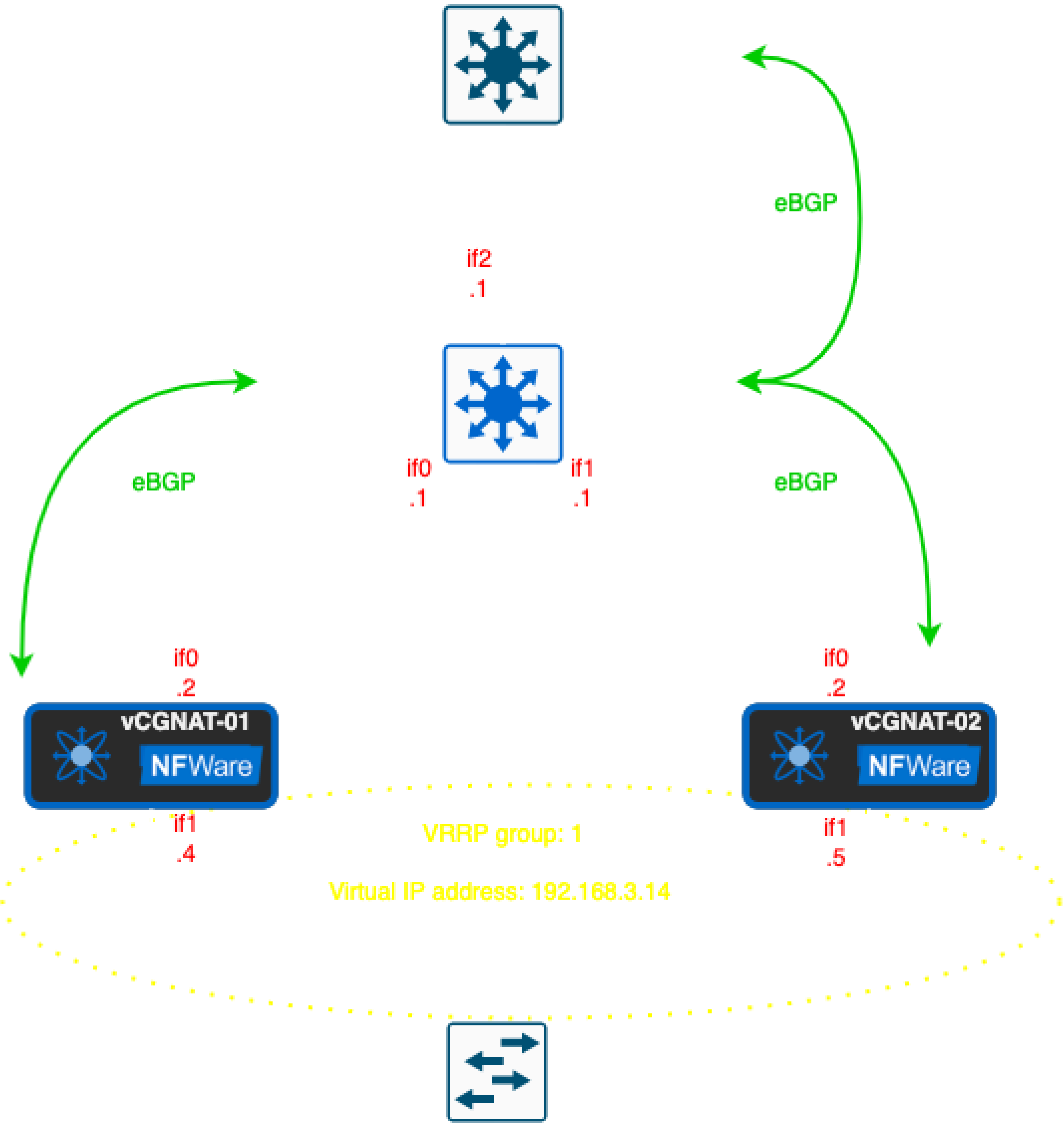
The main idea of High Availability using VRRP is that we have active and standby vCGNATs. Active means that vCGNAT forwards both traffic from customers to the Internet and from the Internet to customers. Standby means that vCGNAT should serve all traffic in case Active vCGNAT loses its uplink or downlink connectivity.

Tracks and RTM CLI policies are used to achieve that. First, we need to track the VRRP state and number of next hops for a particular route (a default route in a typical setup). When VRRP changes its state or number of the next hops for the chosen route, RTM CLI policy kicks in and does what it is programmed for.

The network diagram below represents user hosts directly connected to vCGNATs through the L2 switch. The `if1` interfaces of the vCGNATs, the user hosts, and the Virtual IP address are in the same network `192.168.3.0/28`.

All traffic initially goes through the vCGNAT01, and the vCGNAT02 is in the standby mode. To achieve that, use AS-Path prepend mechanism which will make the route through the vCGNAT02 longer by adding an extra hop. The full configuration of the network nodes can be found in *BGP* section.


AS1
ISP
172.20.1.0/30
eBGP
if2
.1
AS2
outRouter
eBGP
if0
.1
if1
.1
eBGP
192.168.1.0/30
192.168.2.0/30
AS65521
AS65522
if0
.2
if0
.2
vCGNAT-01
NFWare
vCGNAT-02
NFWare
if1
.4
VRRP group: 1
if1
.5
Virtual IP address: 192.168.3.14
L2 switch
192.168.3.0/28
LAN


eBGP
if2
.1
eBGP
if0
.1
if1
.1
eBGP
if0
.2
if0
.2
vCGNAT-01
NFWare
vCGNAT-02
NFWare
if1
.4
VRRP group: 1
if1
.5
Virtual IP address: 192.168.3.14

8.5.3.2 VRRP Configuration

The configuration is very simple: you need to set the VRID, the priority, virtual ip address and enable preempt mode:

vCGNAT01

```
interface if1
 ip address 192.168.3.4/28
 ip nat inside
 vrrp 1 priority 120
 vrrp 1 preempt
 vrrp 1 preempt delay minimum 60
 vrrp 1 ip 192.168.3.14
```

The delay `vrrp 1 preempt delay minimum 60` means that the vCGNAT01 will wait for 60 seconds before becoming Master, and this delay is necessary if, for some reason, BGP is down and it needs enough time to be established and advertise the routes. But, if the *session synchronization* is enabled, you should take into account the time required for the sessions to be synchronizated from vCGNAT02 to vCGNAT01. The delay should be chosen so that there are as few unsynchronized sessions as possible.

The same configuration is for the vCGNAT02, but as it is a VRRP backup router, the priority will be less than 120:

vCGNAT02

```
interface if1
 ip address 192.168.3.5/28
 ip nat inside
 vrrp 1 priority 90
 vrrp 1 preempt
 vrrp 1 ip 192.168.3.14
```

Note: VRID number must be unique in your network, because Virtual MAC depends on VRID and you have to avoid the situations when VRID groups have the same Virtual MAC.

8.5.3.3 Track System Configuration

The track system configuration is applied **only** for the vCGNAT01:

```
!
track 1 vrrp 1 interface if1 ip state master
track 2 ip route 0.0.0.0/0 ecmp-number less-equal 0
!
rtm cli-policy 1
 event track 1 state positive
 action 1 cli configure terminal
 action 2 cli router bgp 65521
 action 3 cli address-family ipv4 unicast
 action 4 cli network 203.0.113.0/25
!
rtm cli-policy 2
 event track 1 state negative
 action 1 cli configure terminal
 action 2 cli router bgp 65521
```

(continues on next page)

(continued from previous page)

```
 action 3 cli address-family ipv4 unicast
 action 4 cli no network 203.0.113.0/25
!
rtm cli-policy 3
 event track 2 state negative
 action 1 cli configure terminal
 action 2 cli interface if1
 action 3 cli vrrp 1 priority 120
!
rtm cli-policy 4
 event track 2 state positive
 action 1 cli configure terminal
 action 2 cli interface if1
 action 3 cli vrrp 1 priority 1
!
```

8.5.3.4 Conclusion

Let us explain what these policies do:

- `RTM cli policy 1` starts announcing NAT pool to BGP when process vrrp 1 changes its state to Master.
- `RTM cli policy 2` stops announcing NAT pool to BGP when process vrrp 1 changes its state from Master to anything else.
- `RTM cli policy 3` increments VRRP priority to a normal value when the number of next hops for the default route becomes bigger than 0.
- `RTM cli policy 4` decrements VRRP priority to a lower value when the number of next hops for the default route becomes 0, which means that route is not in the RIB.

This configuration allows to avoid traffic blackhole in the following cases:

- When the downlink of the Active vCGNAT01 node goes down, the VRRP state changes from Master to Down. `RTM cli policy 2` triggers and withdraws advertising NAT pool network, effectively redirecting all the incoming traffic to the Backup vCGNAT02 node. When the downlink of the Active vCGNAT01 node goes up again, the VRRP state changes to Backup and then to Master, causing `RTM cli policy 1` to announce NAT pool network again, restoring normal traffic flow.
- When the uplink of the Active vCGNAT01 node goes down, the default route is removed from the RIB, which means there are no next hops for it. `RTM cli policy 4` triggers and decrements VRRP priority, effectively redirecting all traffic from customers to the Backup vCGNAT02 node. When uplink is brought up again, the default route reappears in the RIB, and `RTM cli policy 3` changes VRRP priority to normal value, moving customers' traffic to the vCGNAT01 again.
- When the Active vCGNAT01 node reboots, it is essential to steer traffic from inside and outside simultaneously. Otherwise, it may be blackholed for some time. After the vCGNAT01 boots, VRRP establishes the relationship first, and the vCGNAT01 waits for the preemption delay timer. When setting this timer, one should keep in mind that it must be sufficient for BGP to come up and for NAT sessions database to synchronize. While waiting, the vCGNAT01 will be in VRRP Backup state, which means that `RTM cli policy 2` will not allow advertising NAT pool network to the BGP neighbors. When BGP comes up, and the vCGNAT01 receives the default route, `RTM cli policy 3` will trigger and set VRRP priority to normal value, but vCGNAT will still wait in the Backup state. Now, everything is ready, and when the preemption delay timer expires, the vCGNAT01 becomes VRRP Master. `RTM cli policy 1` triggers and announces NAT pool network to the BGP neighbors. Since that moment, all inbound and outbound traffic goes through the vCGNAT01.

CHAPTER

NINE

NAT CONFIGURATION

9.1 Overview

NAT is a method of remapping one IP address space into another by modifying network address information in IP packet headers while they are in transit across a traffic routing device. NAT allows ISP (Internet Service Provider) subscribers to be represented to the public Internet by a different IP address space than the one being used internally.

By taking advantage of transport layer port numbers, NAT also allows mapping many-to-one or many-to-few private to public IP addresses. It means that one or a few public IP addresses can represent many subscribers with private IP addresses. It is possible to use a single public IP address to represent up to 128 subscribers without affecting their Internet experience.

NAT also provides a level of security which is an essential part of any networking implementation. NAT essentially hides all the private IP addresses from the public Internet, providing an excellent additional security measure against hackers.

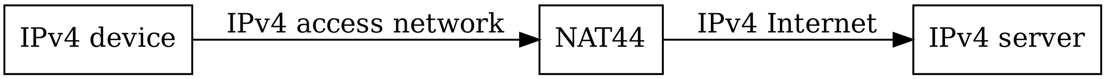
A device enabled with NAT traditionally has at least one interface connected to the inside network and one connected to the outside network. NAT is configured at the border or edge devices between an internal ISP network and a public network in a typical environment. NAT translates the private address into a globally unique public address when a packet leaves the internal network. NAT translates the public destination address back into a private address when a packet enters the internal network.

NAT supports two modes of operation – NAT44 and NAT64.

9.1.1 NAT44

NAT44 operation mode maps private IPv4 address space into public IPv4 address space.

A simplified network diagram for NAT44 is represented on the following figure:



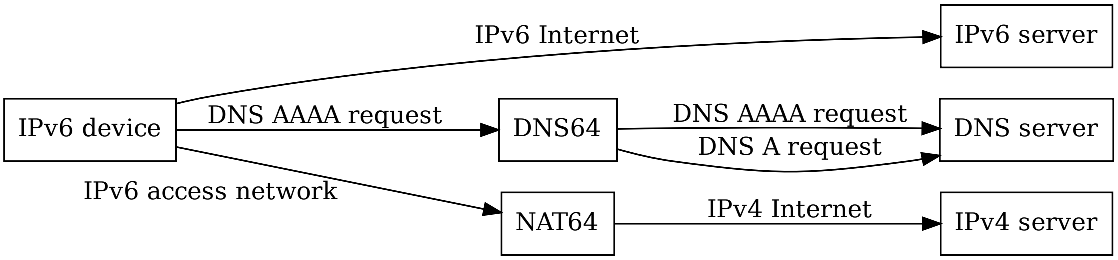
In the internal ISP network, subscriber is represented by a private IPv4 address. When subscriber traffic leaves the network, NAT translates the private IPv4 address into a public IPv4 address.

NAT44 is implemented according to the following RFCs: **RFC 4787**, **RFC 5382**, **RFC 5508**, **RFC 6888** and **RFC 7857**.

9.1.2 NAT64

NAT64 operation mode maps private IPv6 address space into public IPv4 address space. It allows an ISP to have an IPv6-only internal network while maintaining the ability to access IPv4-only Internet services for its subscribers.

A simplified network diagram for NAT64 is represented on the following figure:



When an IPv6 subscriber wants to access some Internet service, it first needs to know the IP address of this service. To find out the address, it sends a DNS `AAAA` request to a special network entity called DNS64.

The idea of DNS64 is to allow the subscriber to directly access IPv6-capable Internet services when possible. Therefore, it first redirects the `AAAA` request to a global DNS server to get an IPv6 address. If it gets a response, it redirects it back to the subscriber. If it doesn't succeed, it sends an `A` request to a global DNS server to get an IPv4 address. If it succeeds, it generates a special IPv6 representation of an IPv4 address and sends it to the subscriber. For this special representation it prepends a Well-Known Prefix `64:ff9b::/96` to an IPv4 address, for example, `1.1.1.1` is mapped to `64:ff9b::1.1.1.1` or, in IPv6 representation, `64:ff9b::0101:0101`. There's more to read about how DNS64 works in **RFC 6147**.

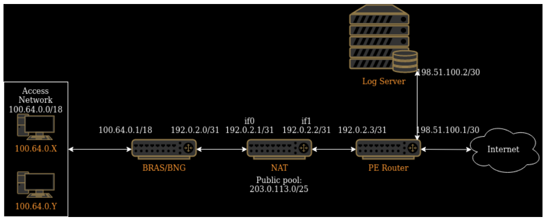
When the subscriber wants to access an actual IPv6 service, it opens a connection directly to the service. When the subscriber wants to access an IPv4-only service, and it found out the special IPv6 representation using the procedure above, it opens a connection through the NAT64. When NAT64 receives a connection with destination address from the Well-Known Prefix, it translates the subscriber's IPv6 address into a public IPv4 address. Also, it translates the destination IPv6 representation back to an IPv4 address. For example, when the subscriber with address `2001:DB8::1` opens a connection to `64:ff9b::0101:0101`, NAT64 translates it into a connection from `203.0.113.1` to `1.1.1.1`.

NAT64 is implemented according to **RFC 6146**.

9.2 Typical Configuration

This section provides an example of a simple core network topology and explains how to configure the NAT to work in it.

9.2.1 Network Topology



The main elements of the example topology are:

1. **BRAS** – aggregates traffic from the *Access Network*.
2. **NAT** – translates private IP addresses into public IP addresses.
3. **PE router** – connects to the *Internet*.
4. **Log Server** – receives and stores session logs from the NAT.

The outbound traffic is processed in the following way:

1. The BRAS receives the subscribers' traffic, enforces the billing policies, and sends the traffic to the NAT.
2. The NAT translates IP addresses into the public space and sends the traffic to the Provider Edge router.
3. The PE router then sends the traffic to the Internet according to the routing information received from the uplink router.

The inbound traffic is processed in the following way:

1. The PE router receives the traffic from the Internet and sends it to the NAT.
2. The NAT translates IP addresses back into the private space and sends the traffic to the BRAS.
3. The BRAS again enforces the billing policies and sends the traffic to the necessary subscriber.

9.2.2 NAT Configuration

First, let's look at the whole NAT config needed to work in the topology described above:

```
!
interface if0
 ip address 192.0.2.1/31
 ip nat inside
!
interface if1
 ip address 192.0.2.2/31
 ip nat outside
!
ip route 0.0.0.0/0 192.0.2.3
ip route 100.64.0.0/18 192.0.2.0
ip route 198.51.100.0/30 192.0.2.3
!
nat log server 0 type ipfix ip 192.168.100.2 port 4739
nat log type session enable
nat log enable
!
nat pool default-pool
 range 203.0.113.1 203.0.113.127
 enable
!
nat subscriber-group default-group
 pool default-pool
!
nat rule subnet 100.64.0.0/18 subscriber-group default-group
!
```

As you can see, the configuration is pretty small and simple. Now let's understand every part of it.

9.2.2.1 Interfaces

```
!
interface if0
 ip address 192.0.2.1/31
 ip nat inside
!
interface if1
 ip address 192.0.2.2/31
 ip nat outside
!
```

To ensure IP connectivity between the NAT and its neighbors, we have to set the necessary IP addresses on the interfaces. Interface `if0` connects to the BRAS and has IP address `192.0.2.1/31`. Interface `if1` connects to the PE router and has IP address `192.0.2.2/31`.

In addition to setting the IP addresses, we need to let the NAT know how to process the traffic received by the interfaces. We need to set the proper NAT type on them to do this. Interface `if0` faces the subscribers (inside network), so we set `ip nat inside` on it. Interface `if1` faces the Internet (outside network), so we set `ip nat outside` on it.

You can find more information about configuring interfaces in section *Interfaces Configuration*.

9.2.2.2 Routing

```
!
ip route 0.0.0.0/0 192.0.2.3
ip route 100.64.0.0/18 192.0.2.0
ip route 198.51.100.0/30 192.0.2.3
!
```

First, NAT must be able to forward the subscribers' traffic to the Internet. To allow this, we need to add the default route `0.0.0.0/0` through the PE router `192.0.2.3`.

Second, NAT must be able to forward the reverse traffic back to the subscribers. To allow this, we need to add the route to the access network `100.64.0.0/18` through the BRAS `192.0.2.0`.

Third, NAT must be able to send logging messages to the log server. To allow this, we need to add the route to the log server network `198.51.100.0/30` through the PE router `192.0.2.3`.

In addition to static routing, NAT also supports various dynamic routing protocols like BGP, OSPF, ISIS and RIP.

You can find more information about configuring routing in section *Routing Configuration*.

9.2.2.3 Logging

```
!
nat log server 0 type ipfix ip 198.51.100.2 port 4739
nat log type session enable
nat log enable
!
```

To comply with various government requirements, it is sometimes needed to log all subscriber connections passing through the NAT.

In the example above, we configure the NAT to send `session` logs to the server `198.51.100.2` on UDP port `4739` using the `IPFIX` protocol.

The `nat log enable` command enables the logging. Removing this command makes it possible to temporarily disable the logging without losing the whole logging configuration.

You can find more information about configuring logging in section *Logging*.

9.2.2.4 NAT

```
!
nat pool default-pool
 range 203.0.113.1 203.0.113.127
 enable
!
nat subscriber-group default-group
 pool default-pool
!
nat rule subnet 100.64.0.0/18 subscriber-group default-group
!
```

Finally, we need to configure the address translation.

First, we create and enable the public address pool named `default-pool` with IP addresses in the range from `203.0.113.1` to `203.0.113.127`.

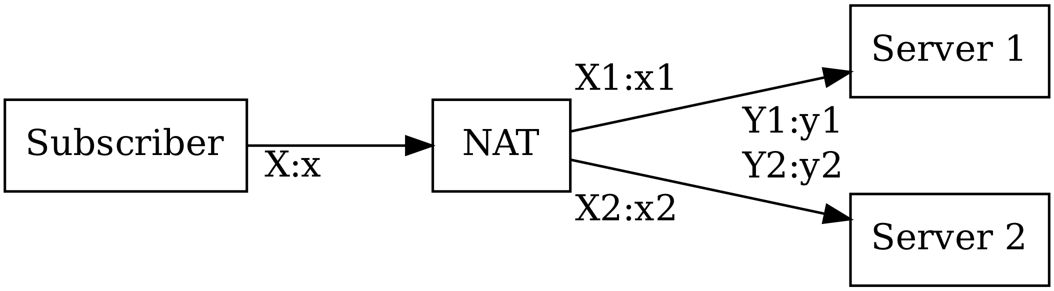
Second, we create a subscriber group named `default-group` and configure it to use the `default-pool` for translation.

Third, we create a rule to process all traffic coming from subnet `100.64.0.0/18` using the subscriber group `default-group`.

You can find more information about configuring pools, groups, and rules in sections *Pools*, *Subscriber Groups* and *Rules*.

9.3 Behaviors

9.3.1 Mapping Behavior



When a subscriber initiates a connection from a private IP address and port `X:x` to some server `Y1:y1`, NAT allocates a public IP address and port for that connection, for example, `X1:x1`. Mapping mode controls how NAT reuses this public IP and port for subsequent connections of this subscriber, for example, a connection to server `Y2:y2`.

There are three possible mapping modes defined in **RFC 4787**:

Endpoint-Independent

NAT always uses the same public IP address and port for all connections initiated from the same private IP address and port. According to the figure above, `X1:x1` always equals `X2:x2`.

Address-Dependent

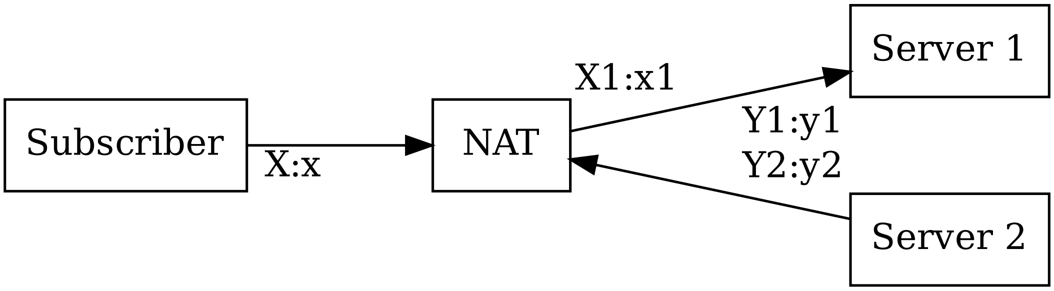
NAT uses the same public IP address and port for connections initiated from the same private IP address and port only when these connections go to the same destination IP address. According to the figure above, `X1:x1` equals `X2:x2` only when `Y2` equals `Y1`.

Address-and-Port-Dependent

NAT never uses the same public IP address and port for more than one connection. According to the figure above, `X1:x1` equals `X2:x2` only when `Y2:y2` equals `Y1:y1`, meaning it is the same connection.

According to REQ-1 of **RFC 4787**, NAT works only in Endpoint-Independent mode.

9.3.2 Filtering Behavior



When a subscriber initiates a connection from a private IP address and port `X:x` to some server `Y1:y1`, NAT allocates a public IP address and port for that connection, for example, `X1:x1`. Filtering mode controls how NAT processes inbound connections to the allocated public IP address and port, for example, a connection coming from server `Y2:y2`.

There are three possible filtering modes defined in **RFC 4787**:

Endpoint-Independent

NAT allows all inbound connections to the allocated public IP address and port. According to the figure above, NAT allows all connections coming to X1:x1.

Address-Dependent

NAT allows inbound connections only from the servers that already have active connections from the subscriber. According to the figure above, NAT allows connections to X1:x1 only when Y2 equals Y1.

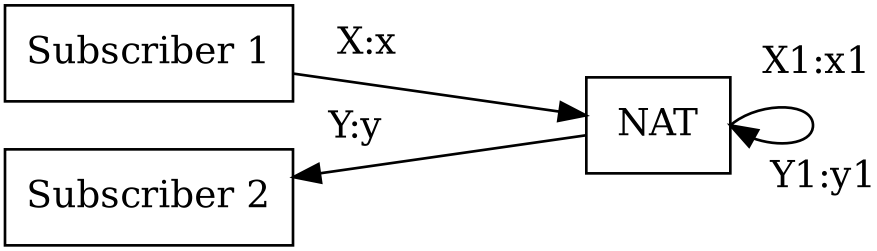
Address-and-Port-Dependent

NAT never allows new inbound connections. According to the figure above, NAT allows connections to X1:x1 only when Y2:y2 equals Y1:y1, meaning it is an existing connection.

According to the REQ-8 of **RFC 4787**, NAT supports all three Filtering modes. The default mode is Endpoint-Independent. To configure a different mode, use the following command:

```
<nat|nat64> filtering <endpoint-independent|address-dependent|address-and-port-dependent> [vrf NAME]
```

9.3.3 Hairpinning Behavior



Imagine there are two subscribers X and Y behind the same NAT. The second subscriber Y has a public IP address and port Y1:y1 allocated by the NAT. Now the first client X tries to connect to this public address and port. When the NAT receives the connection, it has to determine that Y1:y1 belongs to the second subscriber and relay the traffic back to the internal side. This behavior is called hairpinning.

According to the REQ-9 of RFC 4787, NAT supports hairpinning. However, it is disabled by default, and subscribers' traffic to the public IP addresses of the NAT is silently dropped. To enable hairpinning, use the following command:

`<nat|nat64> service hairpinning enable [vrf NAME]`

9.4 Pools

In order for NAT to provide subscribers with access to the global network, it needs to specify a set of public IP addresses to which the private IP addresses of subscribers will be translated. To do this, you need to create at least one pool, and set at least one IP address range for it. This section describes how to do this, as well as additional features for configuring pools.

9.4.1 Pool Creation

`nat pool NAME`

> Create a `NAME` pool and switch to its configuration mode.

All the settings described below are performed in the pool configuration mode.

9.4.2 Adding IP addresses

To begin working with a pool, you need to add at least one range of public IP addresses to it.

`range A.B.C.D A.B.C.D`

> Two `A.B.C.D` parameters are used to set the range of IP addresses added to the pool. The first parameter is the first address in the range, the second parameter is the last address in the range. If you need to add only one IP address, you can do this by using this address as both the first and second parameter, for example, `range 203.0.113.1 203.0.113.1`.
>
> One pool can contain several ranges, both adjacent and not adjacent to each other by IP addresses. The number of ranges, as well as their size, is not limited, but the total number of IP addresses in one pool should not exceed 65536.
>
> To delete a range, use the same command with the `no` prefix. You can delete previously added ranges partially. For example, if you created the `range 203.0.113.1 203.0.113.5` range, but you no longer want to use `203.0.113.2` and `203.0.113.3` addresses in it, you can delete them: `no range 203.0.113.2 203.0.113.3`. Subscriber sessions that use these addresses will be automatically deleted without any impact on subscribers who use another public IP addresses from this pool. The system will automatically split the pool range on two one:

```
nat pool default-pool
range 203.0.113.1 203.0.113.1
range 203.0.113.4 203.0.113.5
enable
```

> If you want to return it, type `range 203.0.113.2 203.0.113.3` and the system will automatically merge these two pool ranges.

9.4.3 Pool Type Configuration

There are four different types of pools:

NAPT (default type)
Pools of this type tranlate both the IP addresses and ports, meaning a single public IP address can be used by many subscribers.

NAT
Pools of this type translate only the IP addresses, meaning each subscriber is assigned a public IP address, and it can no longer be used by other subscribers. Ports are not translated in this mode.

Port Block Allocation
This type is used to enable the Port Block Allocation mode, which allows you to reduce the number of messages sent to the logging server. This mode of operation is described in detail in the *Port Block Allocation* section.

Deterministic
This type is used to enable the Deterministic NAT mode, which avoids the need to use a logging server. This mode of operation is described in detail in the *Deterministic NAT* section.

```
type <nat|napt|port-block-allocation block-size (64-64512)|deterministic block-size (64-64512)>
```

Set the pool type. You can change the type only for the disabled pools. Enabling and disabling pools is described in *Pool Enable* section. For the `port-block-allocation` and `deterministic` types the size of the port blocks must be specified.

9.4.4 Address Allocation Modes

Two modes of public IP addresses allocation are supported:

Paired (default mode)
In this mode, when a subscriber establishes a connection via NAT for the first time, it gets a port on a random public IP address from the pool allocated to this subscriber. This IP address is paired with the subscriber and is used for all its subsequent sessions. This mode can be useful when a subscriber is working with applications that use several connections at the same time and expect that all the connections will be made from a single IP address.

If all the ports of the IP address paired for the subscriber are exhausted, the subscriber won't be able to establish any new connections, and all packets for such connections will be dropped. For example, this may occur, if one IP address gets paired with too many subscribers. In order to avoid this, you can configure the maximum number of subscribers a single public IP address can be paired with. This setting is described in the *Limitations* section.

Arbitrary
In this mode, for each new subscriber connection a port on a random public IP address will be assigned from the pool allocated to this subscriber. A randomly selected address may or may not already have ports allocated to this subscriber.

```
pooling (arbitrary|paired)
```

Set the address allocation mode. You can change the mode only for disabled pools. Enabling and disabling pools is described in *Pool Enable* section.

9.4.5 Pool VRF

By default, after NAT translates the packet, this packet is forwarded in the same VRF in which NAT received it. To change the VRF in which the packet is forwarded, use the following command:

`vrf NAME`

Configure the VRF in which packets are forwarded after being translated using this pool. Use the `no` version of this command to get back to the default behavior.

> **Caution:** If the interface `ip nat inside` is in a VRF, the VRF in the pool must be specified even if it is the default VRF.

9.4.6 Pool Enable

`enable`

This command enables the pool. After enabling the pool, it will not be possible to change its *type* and *address allocation mode*.

If you need to change any of the settings listed above, you need to disable the pool using the `no enable` command.

Before disabling the pool, you need to make sure that:

- the pool is not used by any of the subscriber groups (`no pool` in the group configuration mode),
- the pool is not used in NAT access lists (`no <ip|ipv6> dp-access-list ...`),
- the pool is not used in static mappings (`no nat static ...`).

Otherwise, you will not be able to disable such pool.

Subscriber connections that use this pool will be automatically deleted when it is disabled.

9.4.7 Limitations

When using the *paired* IP address allocation mode, a situation may occur in which too many subscribers are paired with a single public IP address. This may quickly exhaust all the available ports on this address and make the subscribers unable to create new connections.

In order to avoid this, you can configure the maximum number of subscribers a single public IP address can be paired with.

`ratio (1-65535)`

Set the maximum number of subscribers on one external IP address when using the *paired* mode. By default, the number of subscribers is unlimited.

9.4.8 Resource Usage Thresholds

To automatically monitor pools state, you can configure the upper and lower resource usage thresholds. What exactly is considered as a resource depends on the pool type.

NAPT
For the pools of this type, ports available for allocation on public IP addresses are the resource.

NAT
State monitoring is not provided for the pools of this type.

Port Block Allocation
For the pools of this type, port blocks available for allocation on public IP addresses are the resource.

Deterministic
For the pools of this type, port blocks available for allocation on public IP addresses are the resource.

To get notified when the usage thresholds are exceeded, you need to configure the *SNMP traps*.

`thresholds low (0-100)`

Set a lower threshold as a percentage. If the number of resources used is below the threshold, the system will generate `natv2NotificationPoolUsageLow` notifications in accordance with **RFC 7659**.

`thresholds high (0-100)`

Set an upper threshold as a percentage. If the number of resources used exceeds the threshold, the system will generate `natv2NotificationPoolUsageHigh` notifications in accordance with **RFC 7659**.

`thresholds notification-interval <1-3600>`

Set an interval between notifications about exceeding the specified resource usage thresholds. By default, notifications are sent every 20 seconds.

9.4.9 Show Commands

`show nat pool`

Display all created NAT pools with the ICMP, UDP, TCP, and GRE port usage table.

`show nat pool NAME`

If you want to see the particular pool, specify its NAME. For example:

```
nfware# show nat pool default-pool
----------------------------------
Protocol  Used      Free      Load
----------------------------------
ICMP      0         8322945     0%
TCP       0         8322945     0%
UDP       0         8322945     0%
GRE       0         8322945     0%
```

Where `Free = Number_of_IP_addresses_from_NAT_pool x Number_of_available_ports_(65535)`.

`show nat pool NAME counters`

`clear nat pool NAME counters`

Display how many Port Map Entries/Creations/Failure Drops were. `Entries` mean the current value, and `Creations` mean the all-time value.

show nat pool NAME ip

Display the information (how many ports for ICMP, UPD, TCP, and GRE are busy by a specific IP address as a percentage) for all IP addresses in the specified pool.

show nat pool NAME ip A.B.C.D

You can also specify the IP address and then see how many ports for each protocol are used, free and loaded (as a percentage).

show nat pool NAME paired

Display which internal IP address is bound to the external one.

9.5 Subscriber Groups

In order to configure general traffic processing rules for one or more subscriber subnets, use subscriber groups. They can help to identify which pool is used to translate connections, as well as to limit the resource usage. Subscriber groups are linked to the subscriber subnets using NAT *rules*. This section describes the creation and configuration of the subscriber groups.

9.5.1 Subscriber Group Creation

nat subscriber-group NAME

Create a `NAME` subscriber group and switch to its configuration mode.

All the settings described below are performed in the subscriber group configuration mode.

9.5.2 Assigning Pool

If there's no pool assigned to the subscriber group, its traffic will be processed without address translation. In order for NAT to start translating addresses, you need to specify which pool's public addresses will be used for translation.

pool NAME [secondary NAME]

Assign the `NAME` pool to the subscriber group. The pool must be *created* and *enabled* in advance. A single pool can be assigned to several subscriber groups at the same time.

If the `NAME` pool is a *NAT*, type pool, then you can use the `secondary` option to specify a backup pool that will be used if all of the IP addresses of the primary pool are exhausted.

To unassign the pool, use the `no pool` command.

9.5.3 Assigning Access Lists

NAT access lists provide more precise control over how the subscriber group's traffic is translated, for example:

- translate the connections to some IP addresses or ports using a different pool,
- do not translate connections to some IP addresses or ports at all,
- prohibit traffic to some IP addresses or ports.

These are only some of the NAT access lists capabilities. A detailed description of all the features can be found in the *Access Lists* section.

<ip|ipv6> dp-access-list NAME <inside|outside>

After executing this command, the NAME access list of the specified type (ip or ipv6) will be assigned to the subscriber group.

Access lists are checked only when a subscriber creates a new connection. Linking or changing the access list does not affect existing connections. When using the inside option, the access list will be checked for outbound subscriber connections. When using the outside option, the access list will be checked for inbound subscriber connections.

9.5.4 Limits

To prevent any single subscriber from using too much of the common resources, such as public ports or NAT sessions, it is possible to set limitations:

limits port-map-entries (1-536870911)

Set the maximum number of ports that a single subscriber can use. In case a set limit would be reached and a new port will be required, NAT will not allocate any additional ports and drop all traffic sent through this connection. Existing ports will not be affected.

limits port-block-entries (1-536870911)

Set the maximum number of public port blocks that a single subscriber can use. In case a set limit would be reached and a new port block will be required, NAT will not allocate any additional port blocks and drop all traffic sent through this connection. Existing port blocks will not be affected.

limits session-entries (1-536870911)

Set the maximum number of NAT sessions that a single subscriber can use. In case a set limit would be reached and a new connection will be required, NAT will not create any new sessions and drop all traffic sent through this connection. Existing connections will not be affected.

9.5.5 Show Commands

show <nat|nat64> subscribers [vrf NAME]

Display the information on connected subscribers: internal IP address, number of opened ports, and sessions.

```
nfware# show nat subscriber
------------------------------------------------------------
Subscriber          Port Map Entries    Session Entries
------------------------------------------------------------
192.168.96.81        110                 110
192.168.225.203      120                 122
192.168.162.86       9                   9
192.168.254.197      55                  55
192.168.237.222      2                   2
192.168.55.176       37                  41
------------------------------------------------------------
```

show nat64 subscribers X:X::X:X [{counters|vrf NAME}]

show nat subscribers A.B.C.D [{counters|vrf NAME}]

Display current (at the moment) and overall (over a whole period of the operation) counters for specified subscriber.

Current Counters	Description
`Ports`	The number of opened ports
`Port Blocks`	The number of allocated port blocks. You can configure a pool in Port Block Allocation mode (see *Port Block Allocation* section) to allocate not only one port but also blocks. For example, to reduce logs
`Sessions`	The number of opened sessions

Overall Counters	Description
`Ports opened`	The number of opened ports
`Port Blocks allocated`	The number of allocated port blocks
`Sessions created`	The number of created sessions
`Port Map Failure Drops`	See **RFC 7659** for reference: • the limit on the number of ports/blocks per subscriber is exceeded • there are no free ports (taking into account paired limits and pool settings)

show nat subscriber-group NAME <ip|ipv6> dp-access-list <inside|outside>

Display the ACL configuration for inside and outside network and the number of times each ACL rule has been matched.

9.6 Rules

In order to specify to which subscriber group subscribers from a certain subnet belong, NAT rules are used. For the subscribers that are not assigned to any subscriber group by a rule, traffic will be dropped.

nat rule subnet A.B.C.D/M subscriber-group NAME [vrf NAME]

nat64 rule subnet X:X::X:X/M subscriber-group NAME [vrf NAME]

Add a standard NAT rule. After executing this command, subscribers from the subnet specified by the `subnet A.B.C.D/M` option will be assigned to the subscriber group specified by the `subsriber-group NAME` option. The `vrf NAME` option allows to specify the VRF in which the rule will be created. Without using this option, the rule will be created in the default VRF.

nat rule subnet A.B.C.D/M passthrough [vrf NAME]

nat64 rule subnet X:X::X:X/M passthrough [vrf NAME]

Add a passthrough NAT rule. For the subscribers that fall under this rule traffic will be processed without address translation, sessions creations, limits tracking, etc. If you need to pass subscriber traffic without translation, but at the same time monitor sessions and set limits, use a standard NAT rule with a subscriber group specified, but don't assign any public addresses pool to this group.

If a new rule's subnet intersects with other existing rules, then the most specific rule will be applied, just as it works with routing. For example, if you have rules for subnets `100.64.0.0/24` and `100.64.0.128/25`, then for the `100.64.0.129` subscriber the second rule will be applied.

If a new rule's subnet completely matches one of the existing rules, the existing rule will be replaced with a new one.

Creating and changing rules affects already existing subscribers - if at any point in time rule A will be switched to a new rule B, all existing subscriber sessions will still be working according to the rule A, yet all the new subscriber sessions will be created according to the new rule B.

Deleting a rule affects only the new connections established by subscribers from a subnet. For all the existing sessions of such subscribers traffic will be processed. If you also want to delete the current sessions, you can use the `clear nat sessions int-ip A.B.C.D` command.

9.7 Access Lists

NAT access lists provide more precise control over how the subscriber traffic is translated. They allow you to perform special actions on certain types of traffic using filters.

Filters can include:

- Protocol (UDP, TCP, ICMP, GRE, ESP),
- Source and destination IP addresses,
- Source and destination ports (for UDP and TCP),
- Message type and code (for ICMP).

Action types:

Permit (`permit`)
: This is the default action - sessions are processed according to the action specified in the subscriber group.

Deny (`deny`)
: Prohibits creation of new sessions. All traffic for the sessions will be dropped.

Passthrough (`passthrough`)
: New sessions will be passed transparently, without translation of addresses and ports. At the same time, session state will be monitored as in the normal NAT mode.

Pool (`pool`)
: Sessions will be translated using the pool specified in the action.

9.7.1 Configuration

Note: After configuration, an access list must be assigned to a subscriber-group as described in *Assigning Access Lists*.

Pay attention: after the creation of the access list, the rule `deny any any` will be added, and it will not be shown in the running configuration file. If you want to permit any traffic, you should explicitly set the rule `permit any any`.

The following commands are available to configure access lists:

```
ip dp-access-list NAME SEQ ACTION <any|udp|tcp|icmp|gre|esp> src-ip <any|A.B.C.D/M> dst-ip <any|A.B.C.D/M>
```

```
ipv6 dp-access-list NAME SEQ ACTION <any|udp|tcp|icmp|gre|esp> src-ip <any|X:X::X:X/M> dst-ip <any|X:X::X:X/M>
```

Create a filter by IP addresses. These commands are available for all protocols.

Key	Argument	Description
NAME		Specify access list name
SEQ	(1-536870911	Set sequence number
ACTION	<deny|passthrough|permit|pool>	Set the action
<any|udp|tcp|icmp|gre|es		Set the particular or any protocol
src-ip	<any|A.B.C.D/M> or <any|X:X::X:X/M>	Set source address subnet
dst-ip	<any|A.B.C.D/M> or <any|X:X::X:X/M>	Set destination address subnet

ip dp-access-list NAME SEQ ACTION <udp|tcp> src-ip <any|A.B.C.D/M> dst-ip <any|A.B.C.D/M> src-port (0-65535) (0-65535) dst-port (0-65535) (0-65535)

ipv6 dp-access-list NAME SEQ ACTION <udp|tcp> src-ip <any|X:X::X:X/M> dst-ip <any|X:X::X:X/M> src-port (0-65535) (0-65535) dst-port (0-65535) (0-65535)

Create a filter by IP addresses and ports. These commands are available only for the TCP and UDP protocols.

Key	Argument	Description
NAME		Specify access list name
SEQ	(1-536870911	Set sequence number
ACTION	<deny|passthrough|permit|pool>	Set the action
<udp|tcp>		Set the particular or any protocol
src-ip	<any|A.B.C.D/M> or <any|X:X::X:X/M>	Set source address subnet
dst-ip	<any|A.B.C.D/M> or <any|X:X::X:X/M>	Set destination address subnet
src-port	(0-65535) (0-65535)	Set the source port range start and end
dst-port	(0-65535) (0-65535)	Set the destination port range start and end

ip dp-access-list NAME SEQ ACTION icmp src-ip <any|A.B.C.D/M> dst-ip <any|A.B.C.D/M> icmp-type (0-255) (0-255) icmp-code (0-255) (0-255)

ipv6 dp-access-list NAME SEQ ACTION icmp src-ip <any|X:X::X:X/M> dst-ip <any|X:X::X:X/M> icmp-type (0-255) (0-255) icmp-code (0-255) (0-255)

Create a filter by IP addresses, the `Type`, and the `Code` fields of the ICMP header. These commands are available only for ICMP protocol.

Key	Argument	Description
NAME		Specify access list name
SEQ	(1-536870911	Set sequence number
ACTION	<deny|passthrough|permit|pool>	Set the action
icmp		Set the particular or any protocol
src-ip	<any|A.B.C.D/M> or <any|X:X::X:X/M>	Set source address subnet
dst-ip	<any|A.B.C.D/M> or <any|X:X::X:X/M>	Set destination address subnet
icmp-type	(0-255) (0-255)	Set ICMP type range start and end
dst-code	(0-255) (0-255)	Set ICMP code range start and end

9.7.2 Show Commands

`show <ip|ipv6> dp-access-list`

`clear <ip|ipv6> dp-access-list`

Display all created dp-access-lists, their description and the number of matches for each rule.

`show <ip|ipv6> dp-access-list NAME`

`clear <ip|ipv6> dp-access-list NAME`

Display information for the specified ACL list.

9.8 Logging

Sometimes, in order to meet the requirements of the government agencies, it is necessary to monitor all subscriber connections passing through NAT. For these purposes, it is possible to configure connections logging on external servers. This section describes various logging parameters and commands available to configure them.

9.8.1 Log Types

There are several types of messages that could be sent to external servers:

Address mapping (`address-map`)

Log creation and deletion events of the mappings between private and public IP addresses. Ports and destination addresses are not logged. This type of message is suitable if you're using *NAT type* of pools and you do not need to store information about destination addresses.

Port mapping (`port-map`)

Log creation and deletion events of the mappings between private and public IP addresses and ports. This type of message is suitable if you're using *NAPT type* of pools and you do not need to store information about destination addresses.

Session (`session`)

Log the subscriber session start and end events. Private addresses, public addresses, and ports, as well as destination addresses are logged. This type of message is the most detailed.

Port blocks (`port-block`)

Log the port blocks allocation and release events. Destination addresses are not logged. This type of messages is available only when using pools of the *Port Block Allocation type*.

The specific message format depends on the logging protocol and will be described below.

9.8.2 Logging Protocols

9.8.2.1 Syslog

When using this protocol, messages are sent to an external Syslog server in accordance with **RFC 5424**.

Packet header

The header format is described in **RFC 5424#section-6**.

Header fields are usually automatically processed by the Syslog server and provided in a readable format.

`PRI`
: This field combines the `Facility` and `Severity` parameters of the message. By default, messages are sent with `Facility = LOCAL0` and `Severity = Informational`. It is possible to change both parameters depending on the server and the logging type.

`VERSION`
: In this field, `1` is always sent.

`TIMESTAMP`
: By default, messages are sent without timestamps, meaning that the `-` symbol is sent in this field. There is an option to enable timestamps.

`HOSTNAME`
: By default, messages are sent without the system name, meaning that the `-` symbol is sent in this field. There is an option to enable system name.

`APP-NAME`
: `NAT` is always sent in this field.

`PROCID`
: `-` is always sent in this field.

`MSGID`
: `-` is always sent in this field.

`STRUCTURED-DATA`
: `-` is always sent in this field.

Message format

Messages are sent in the `MSG` field. The Syslog server provides these messages “as is”. The message format depends on the logging type used.

Address mapping display format:

```
EVENT VRF ID INT IP EXT IP
```

`EVENT` - `A` when created, `D` when deleted.

`VRF ID` - the `ID` field specifies the ID of the VRF that processes the subscriber’s traffic.

`INT IP` - the `IP` field specifies subscriber’s private IP address.

`EXT IP` - the `IP` field specifies subscriber’s public IP address.

Example:

```
A VRF 0 INT 10.0.0.1 EXT 100.64.0.1
```

Port mapping display format:

```
EVENT VRF ID PROTO INT IP:PORT EXT IP:PORT
```

`EVENT` - `A` when created, `D` when deleted.

`VRF ID` - the `ID` field specifies the ID of the VRF that processes the subscriber's traffic.

`PROTO` - the protocol number in accordance with the IANA registry.

`INT IP:PORT` - the `IP:PORT` field specifies subscriber's private IP address and port.

`EXT IP:PORT` - the `IP:PORT` field specifies subscriber's public IP address and port.

Example:

```
A VRF 0 6 INT 10.0.0.1:57938 EXT 100.64.0.1:28475
```

Sessions format:

```
EVENT VRF ID PROTO INT IP:PORT EXT IP:PORT DST IP:PORT DIR TYPE
```

`EVENT` - `A` when created, `D` when deleted.

`VRF ID` - the `ID` field specifies the ID of the VRF that processes the subscriber's traffic.

`PROTO` - the protocol number in accordance with the IANA registry.

`INT IP:PORT` - the `IP:PORT` field specifies the subscriber's private IP address and port.

`EXT IP:PORT` - the `IP:PORT` field specifies the subscriber's public IP address and port.

`DST IP:PORT` - the `IP:PORT` field` specifies the destination IP address and port.

`DIR TYPE` - the `TYPE` field specifies `OUT` for outbound sessions and `IN` for inbound sessions.

Example:

```
A VRF 0 6 INT 10.0.0.1:57938 EXT 100.64.0.1:28475 DST 185.165.123.206:443 DIR␣
↪OUT
```

Port blocks format:

```
EVENT VRF ID INT IP EXT IP:START-END
```

`EVENT` - `A` when created, `D` when deleted.

`VRF ID` - the `ID` field specifies the ID of the VRF that processes the subscriber's traffic.

`INT IP` - the `IP` field specifies subscriber's private IP address.

`EXT IP:START-END` - the `IP:START-END` field specifies subscriber's public IP address and the port block allocated to it.

Example:

```
A VRF 0 INT 10.0.0.1 EXT 100.64.0.1:1024-1535
```

9.8.2.2 NetFlow

When using this protocol, messages are sent to an external NetFlow collector in accordance with **RFC 3954**.

Packet header

The header format is described in **RFC 3954#section-5**.

All fields except `Source ID` are standard. The `Source ID` field is used to separate different message flows from the same source. Each processor core sends its own message flow, so the `Source ID` field specifies the number of the core that sent a certain packet, so that the receiver can divide the received packets into different flows.

Message format

The message format depends on the logging type used.

Address mapping display format:

Field Name	Size (bits)	IANA ID	Description
timeStamp	64	323	System Time when the event occurred
sourceIPv4Address	32	8	Source IPv4 Address
postNATSourceIPv4Address	32	225	Translated Source IPv4 Address
ingressVRFID	32	234	VRF ID in case of overlapping networks
natEvent	8	230	Type of Event

Port mapping display format:

Field Name	Size (bits)	IANA ID	Description
timeStamp	64	323	System Time when the event occurred
sourceIPv4Address	32	8	Source IPv4 Address
postNATSourceIPv4Address	32	225	Translated Source IPv4 Address
protocolIdentifier	8	4	Transport protocol
sourceTransportPort	16	7	Source Port
postNAPTSourceTransportPort	16	227	Translated Source port
ingressVRFID	32	234	VRF ID in case of overlapping networks
natEvent	8	230	Type of Event

Sessions format:

Field Name	Size (bits)	IANA ID	Description
timeStamp	64	323	System Time when the event occurred
sourceIPv4Address	32	8	Source IPv4 Address
postNATSourceIPv4Address	32	225	Translated Source IPv4 Address
protocolIdentifier	8	4	Transport protocol
sourceTransportPort	16	7	Source Port
postNAPTSourceTransportPort	16	227	Translated Source port
destinationIPv4Address	32	12	Destination IPv4 Address
postNATDestinationIPv4Address	32	226	Translated IPv4 destination address
destinationTransportPort	16	11	Destination port
postNAPTDestinationTransportPort	16	228	Translated Destination port
natOriginatingAddressRealm	8	229	Address Realm
ingressVRFID	32	234	VRF ID in case of overlapping networks
natEvent	8	230	Type of Event

Port blocks format:

Field Name	Size (bits)	IANA ID	Description
timeStamp	64	323	System Time when the event occurred
sourceIPv4Address	32	8	Source IPv4 Address
postNATSourceIPv4Address	32	225	Translated Source IPv4 Address
portRangeStart	16	361	Allocated Port Block start
portRangeEnd	16	362	Allocated Port Block end
ingressVRFID	32	234	VRF ID in case of overlapping networks
natEvent	8	230	Type of Event

9.8.2.3 IPFIX

When using this protocol, messages are sent to an external IPFIX collector in accordance with **RFC 7011**.

Packet header

The header format is described in **RFC 7011#section-3**.

All fields except `Observation Domain ID` are standard. The `Observation Domain ID` field is used to separate different message flows from the same source. Each processor core sends its own message flow, so the `Observation Domain ID` field specifies the number of the core that sent a certain packet, so that the receiver can divide the received packets into different flows.

Message format

The message format is identical to the *NetFlow* message format.

9.8.2.4 RADIUS

When using this protocol, messages are sent to an external RADIUS server in accordance with **RFC 2865** and **RFC 2866**.

Packet header

The header format is described in **RFC 2865#section-3**.

To send information about the subscriber sessions, `Accounting-Request` messages are used, so the `Code` header field is set to the value `4`. The rest of the header fields are standard.

> **Warning:** This is not a full-fledged implementation of the RADIUS protocol. NAT only uses the `Accounting-Request` message format to send messages to an external server. NAT does not expect to receive `Accounting-Response` from the RADIUS server.

Message format

To correctly process messages on the RADIUS server, you need to download the `RADIUS dictionary`. The following description uses the internal identifiers described in this dictionary.

Address mapping display format:
Not supported.

Port mapping display format:

Type	Length (bytes)	Description
Acct-Status-Type	4	1 for Start, 2 for Stop
Acct-Session-Id	4	Hash value for all message fields
NAS-Identifier	6	“vCGNAT”
Event-Timestamp	4	System Time when the event occurred
NFWare-vCGNAT-Protocol	4	Transport protocol
NFWare-vCGNAT-Action	4	3 for Add, 4 for Delete
NFWare-vCGNAT-Inside-Addr	4	Source IPv4 Address
NFWare-vCGNAT-Inside-Port	2	Source Port
NFWare-vCGNAT-Nat-Addr	4	Translated Source IPv4 Address
NFWare-vCGNAT-Nat-Port	2	Translated Source port
NFWare-vCGNAT-VRF	4	VRF ID in case of overlapping networks
NFWare-vCGNAT-Direction	4	Direction

Sessions format:

Type	Length (bytes)	Description
Acct-Status-Type	4	1 for Start, 2 for Stop
Acct-Session-Id	4	Hash value for all message fields
NAS-Identifier	6	“vCGNAT”
Event-Timestamp	4	System Time when the event occurred
NFWare-vCGNAT-Protocol	4	Transport protocol
NFWare-vCGNAT-Action	4	3 for Add, 4 for Delete
NFWare-vCGNAT-Inside-Addr	4	Source IPv4 Address
NFWare-vCGNAT-Inside-Port	2	Source Port
NFWare-vCGNAT-Nat-Addr	4	Translated Source IPv4 Address
NFWare-vCGNAT-Nat-Port	2	Translated Source port
NFWare-vCGNAT-Dest-Addr	4	Destination IPv4 Address
NFWare-vCGNAT-Dest-Port	2	Destination port
NFWare-vCGNAT-VRF	4	VRF ID in case of overlapping networks
NFWare-vCGNAT-Direction	4	Direction

Port blocks format:

Type	Length (bytes)	Description
Acct-Status-Type	4	1 for Start, 2 for Stop
Acct-Session-Id	4	Hash value for all message fields
NAS-Identifier	6	“vCGNAT”
Event-Timestamp	4	System Time when the event occurred
NFWare-vCGNAT-Protocol	4	Transport protocol
NFWare-vCGNAT-Action	4	3 for Add, 4 for Delete
NFWare-vCGNAT-Inside-Addr	4	Source IPv4 Address
NFWare-vCGNAT-Nat-Addr	4	Translated Source IPv4 Address
NFWare-vCGNAT-NAT-Port-Start	2	Allocated Port Block start
NFWare-vCGNAT-NAT-Port-End	2	Allocated Port Block end
NFWare-vCGNAT-VRF	4	VRF ID in case of overlapping networks
NFWare-vCGNAT-Direction	4	Direction

9.8.3 Configuration

To begin logging connections information to external servers, it is necessary to enable the required types of logs, configure at least one server, and enable logging globally.

You can configure several logging servers at the same time, they can use the same protocol or different ones. Messages will be duplicated to all configured servers.

When adding a server, it is possible to configure the sender’s IP address and a VRF that will be used to route logs packets. By default, packets are routed within the default VRF, and the sender address is the address of the interface from which the packets are sent.

```
nat log type <address-map|port-map|session|port-block> enable
```

Enable the appropriate logging type.

```
no nat log type <address-map|port-map|session|port-block> enable
```

Disable the appropriate logging type.

nat log server (0-62) type <syslog|netflow|ipfix|radius> ip A.B.C.D port (1-65535) [{vrf NAME|source-ip A.B.C.D}]

Add a logging server. If you already have a server configured with a specific identifier (parameter `(0-62)`), then if you re-use the same identifier, the previous settings will be overwritten.

no nat log server (0-62)

Delete a log server.

nat log server (0-62) type syslog hostname NAME

Add the system name when sending messages via the Syslog protocol.

nat log server (0-62) type syslog timestamp enable

Add a timestamp when sending messages via the Syslog protocol.

nat log server (0-62) type syslog log-type <address-map|port-map|session|port-block> facility FACILITY

Set the `Facility` parameter of the PRI field when sending messages via the Syslog protocol.

nat log server (0-62) type syslog log-type <address-map|port-map|session|port-block> level LEVEL

Set the `Severity` parameter of the PRI field when sending messages via the Syslog protocol.

nat log server (0-62) type <netflow|ipfix> template-resend-pkts (1-1440)

When sending logs using the *NetFlow* and *IPFIX* protocols, it is necessary to periodically refresh the template with the format of the messages being sent. This command allows you to set the frequency of the template refresh in the number of packets. By default, the template is refreshed in every 20th packet.

nat log server (0-62) type <netflow|ipfix> template-resend-timeout (1-1440)

When sending logs using the *NetFlow* or *IPFIX* protocols, it is necessary to periodically refresh the template with the format of the messages being sent. This command allows you to set the frequency of the template refresh in minutes. By default, the template is refreshed every 10 minutes.

nat log server (0-62) type radius secret SECRET

Set the `Shared Secret` when sending messages via the RADIUS protocol.

nat log enable

Enable logging.

<<<<<<< HEAD

9.8.4 Show Commands

show nat log counters

clear nat log counters

Display the NAT log counters.

Counter	Description
Generat	The number of the generated log messages
Sent	The number of the log messages that have been already sent (if there are several log servers, then sent = generated * nb_log_servers)
No Memory	The number of the log messages that were not queued for processing
No Server	The number of the log messages that could not be sent because there is no configured log server
No VRF	The number of the log messages could not be sent because the vrf, to which the log server should be bound, is missing
No Packet	The number of the log messages could not be sent because of failure to allocate a packet from the packet pool to write nat log messages to it

show nat log queues

Display NAT log queues. The output is as follows:

```
nfware# sh nat log queues
NAT-log task id 0 (socket 0):
NAT-log ring-size: 16384
            Free-space          Processed      Loading
    Ring id 0 : 16384               0             0  %
    Ring id 1 : 16384               0             0  %
```

Name	Description
NAT-log task id	NAT-log task id is bound to cores, and the core is bound to a socket. The number of tasks is configured in the platform settings
NAT-log ring-size: 16384	The maximum size of the messages queue that is waiting to be sent to the log server
Ring id 0	Control plane queue
Ring id 1	Data plane queue
Processed	The total number of the processed log messages

show nat log servers

Display all log servers and their configuration. There are three states the server could have:

State	Description
Active	The log server is configured, and nat log messages are sent to it
Inactive	If you configured the log server with VRF and there is no such VRF
Not Configure	IP address and port are not configured

9.8.4.1 NAT Log Balancing

The NAT log balancing mechanism divides the entire logging server set into groups. For each NAT entry (for example, creating or deleting a session), exactly one server is allocated from each group, and the entry is duplicated on all selected servers. The choice of one or another server in the group depends on the IP source address. All actions of one user are logged on the same subset of servers.

Note: Groups consisting of a single server are not explicitly identified as a group. They are single servers and are configured in the same way as mentioned above. However, it should be considered that the balancing mechanism is not applied to them, and the logs of all users will be duplicated on them.

`nat log group N`

Create a group of servers where N is the group number. Up to eight server groups are allowed, numbered 1, 2, and 3. Each group can have up to 24 servers, while the numbering of servers in different groups is independent. The numbering of single servers is independent of each of the groups. The total number of servers, including single servers and servers in all groups, can be at most 63 because of the synchronization mechanism between control and worker threads. Inside the group, you can add servers and set their parameters using any of the existing commands of the form.

`no nat group N`

Delete an entire server group. If you delete all servers in a group, the group itself will also be deleted.

9.9 Timeouts

To ensure the correct operation, NAT monitors subscribers' sessions.

For the TCP and UDP protocols, the session is defined by these five fields in the packet:

- Source IP Address,
- Destination IP Address,
- Protocol,
- Source port,
- Destination port.

For the ICMP protocol, the session is defined by these four fields:

- Source IP address,
- Destination IP address,
- Protocol,
- Query ID.

For GRE and ESP protocols, the session is defined by these three fields:

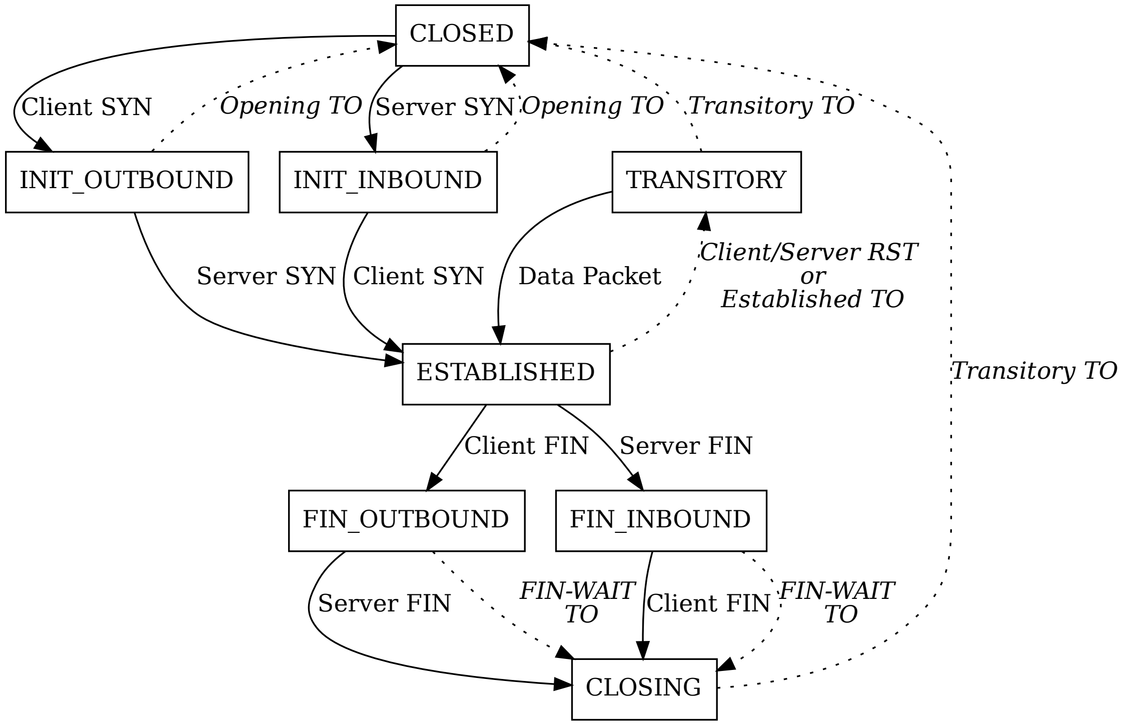
- Source IP address,
- Destination IP address,
- Protocol.

When the NAT receives a packet, it checks whether it has an existing record in the session table in accordance with the fields specified above. If the session exist, NAT updates its state. If there is no record for the session yet, NAT creates it. To prevent the session table from overflowing, NAT monitors session activity. If no packets had passed through the session for a certain time, the session is considered outdated and is deleted from the table.

This section describes how timeouts work and how to configure them for various protocols.

9.9.1 TCP

NAT monitors the state of TCP sessions in accordance with **RFC 5382#section-5** and **RFC 7857#section-2** using the following finite-state machine:



CLOSED

A pseudo-state that reflects a non-existent session. After NAT receives a `SYN` packet from the subscriber or server, it creates a session and puts it in the `INIT_OUTBOUND` or `INIT_INBOUND` state, respectively.

INIT_OUTBOUND and INIT_INBOUND

In this state, NAT expects to receive a response `SYN` packet from the server or subscriber, respectively. If it receives such a packet during the `Opening Timeout`, the session goes in the `ESTABLISHED` state. If the packet has not been received, NAT closes the session.

ESTABLISHED

The main state of the TCP session. If NAT receives a `FIN` packet from the subscriber or server, it puts the session in the `FIN_OUTBOUND` or `FIN_INBOUND` state, respectively. If the NAT receives an `RST` packet from the subscriber or server, or does not receive any packets during the `Established Timeout`, it puts the session in the `TRANSITORY` state.

TRANSITORY

If NAT receives any packet for the session in this state, it transfers the session back to the `ESTABLISHED` state. If NAT does not receive any packets during the `Transitory Timeout`, it closes the session.

FIN_OUTBOUND and FIN_INBOUND

In this state, NAT expects to receive a response `FIN` packet from the server or subscriber, respectively. If it receives such a packet or does not receive during `FIN-WAIT Timeout`, it puts the session in the `CLOSING` state. If NAT does not receive any packets during the `Established Timeout`, it closes the session.

CLOSING

The session stays in this state until `Transitory Timeout` expires, then it's closed.

To configure TCP session timeouts, the following commands are available:

`<nat|nat64> timeout tcp opening (1-604800) [vrf NAME]`

Set `Opening Timeout` in seconds. By default, it is 240 seconds (4 minutes).

`<nat|nat64> timeout tcp established (1-604800) [vrf NAME]`

Set `Established Timeout` in seconds. By default, it is 7440 seconds (2 hours and 4 minutes).

`<nat|nat64> timeout tcp transitory (1-604800) [vrf NAME]`

Set `Transitory Timeout` in seconds. By default, it is 240 seconds (4 minutes).

`<nat|nat64> timeout tcp fin-wait (1-604800) [vrf NAME]`

Set `FIN-WAIT Timeout` in seconds. By default, it is 7440 seconds (2 hours and 4 minutes).

9.9.2 UDP

NAT monitors the state of UDP sessions in accordance with **RFC 4787#section-4.3**. If there are no packets through the created UDP session during the specified timeout, the session is closed.

`<nat|nat64> timeout udp (1-604800) [vrf NAME]`

Set the timeout for UDP sessions in seconds. By default, it is 300 seconds (5 minutes).

9.9.3 ICMP

NAT monitors the state of ICMP sessions in accordance with **RFC 5508#section-3.2**. If there are no packets through the created ICMP session during the specified timeout, the session is closed.

`<nat|nat64> timeout icmp (1-604800) [vrf NAME]`

Set the timeout for ICMP sessions in seconds. By default, it is 60 seconds (1 minute).

9.9.4 GRE

If there are no packets through the created GRE session during the specified timeout, the session is closed.

`<nat|nat64> timeout gre (1-604800) [vrf NAME]`

Set the timeout for GRE sessions in seconds. By default, it is 1800 seconds (30 minutes).

9.9.5 ESP

If there are no packets through the created ESP session during the specified timeout, the session is closed.

`<nat|nat64> timeout esp (1-604800) [vrf NAME]`

Set the timeout for ESP sessions in seconds. By default, it is 1800 seconds (30 minutes).

9.9.6 Timeout Update

By default, NAT updates session timeouts for packets in both directions - from subscribers to servers (outbound), and from servers to subscribers (inbound). For additional protection against external attacks, it is possible to disable the timeouts updates for the inbound packets.

`<nat|nat64> service inbound-refresh enable [vrf NAME]`

Enable timeouts updates for the inbound packets. This is the default behavior. To disable it, use this command with the `no` prefix.

9.9.7 Show Commands

`show <nat|nat64> timeout [vrf NAME]`

Display the current timeout values for ESP, GRE, ICMP, UDP and TCP protocols.

9.10 Port Control Protocol

PCP allows the subscriber to request NAT to open external ports without initializing connection. This can be useful if the subscriber wants to provide access to an internal resource, but there is no way to provide him with a permanent external IP address.

PCP is implemented in accordance with **RFC 6887**.

9.10.1 Configuration

`<nat|nat64> pcp enable [vrf NAME]`

Enable PCP. By default, PCP is disabled.

`<nat|nat64> pcp map disable [vrf NAME]`

Disable MAP messages processing. By default, MAP messages processing is enabled.

`<nat|nat64> pcp peer disable [vrf NAME]`

Disable PEER messages processing. By default, PEER messages processing is enabled.

`<nat|nat64> pcp third-party enable [vrf NAME]`

Enable `THIRD_PARTY` option support. By default, this option is disabled.

9.11 Port Block Allocation

This mode allows you to significantly reduce the number of logs about subscriber connections passing through NAT.

In standard mode NAT allocates a new external port for each new connection. In this case, either the fact of allocating a port or the fact of establishing a connection is logged, depending on the *type of logging* used.

In the Port Block Allocation mode, a whole block of external ports is immediately allocated to the subscriber, then only the ports from the allocated block are used for each new connection. When the subscriber closes all of its connections, the port block allocated to it is released. In this case, the events of allocating and releasing a block of ports are logged, which reduces the number of logs by the factor of hundreds. The log message format is described in detail in the *Logging* section.

The port blocks are protocol-independent. For example, if a port block `12032-12287` is allocated to the subscriber, then both of its TCP and UDP connections will use this block.

Note: When using this mode, the destination address is not logged. The logs contain only the private address of the subscriber, its public IP address, and a port block. If you need to store information about destination addresses, then this mode of operation will not suit you.

9.11.1 Configuration

For a minimal configuration, you need to perform the following steps.

1. Configure `port-block` type logging, for example, using the IPFIX protocol:

   ```
   nat log server 0 type ipfix ip 192.168.1.200 port 4739
   nat log type port-block enable
   nat log enable
   ```

   These commands are described in detail in the *Logging* section.

2. Create `port-block-allocation` type pool:

   ```
   nat pool pba-pool
    range 203.0.113.1 203.0.113.5
    type port-block-allocation block-size 256
    enable
   ```

   In this example, blocks of 256 ports are used. If the client needs more than 256 ports, additional blocks will be allocated to it. These commands are described in detail in the *Pools* section.

3. Create a subscriber group and configure it to use this pool:

   ```
   nat subscriber-group pba-group
    pool pba-pool
    limits port-block-entries 3
   ```

   In this example, a limit of 3 port blocks per subscriber is set. This means that with a block size of 256 ports, a maximum of 768 ports will be available to one subscriber. These commands are described in detail in the *Subscriber Groups* section.

4. Set the subscriber subnet that will use this group:

   ```
   nat rule subnet 100.64.0.0/16 subscriber-group pba-group
   ```

   This command is described in detail in the *Rules* section.

9.12 Deterministic NAT

This mode of operation is implemented in accordance with **RFC 7422** and eliminates the need for logging subscriber connections.

As in the *Port Block Allocation* mode, in this mode, NAT allocates port blocks for the subscribers. The difference is that the port blocks allocation is done algorithmically during the configuration process (instead of being allocated randomly when necessary). Therefore, you need to think in advance about the correspondence between the subscriber addresses and the external port blocks.

NAT supports `Sequential` port block allocation mode:

- reserved ports 1-1023 are not used,
- the remaining ports (1024-65535) are divided into blocks according to the size specified in the pool configuration,
- the first block is assigned to the first subscriber, the second block is assigned to the second one, etc.

The network address and the broadcast address are also considered subscribers, so their own blocks of addresses are allocated to them.

Note: When using this mode, you do not have the information about the destination addresses of subscriber connections. If you need to store information about destination addresses, then this mode of operation will not suit you.

9.12.1 Block Allocation

Let's say you have 14 subscribers in the network `100.64.0.0/28`. Taking into account the network address and the broadcast address, you need 16 blocks.

In the pool, you have two external IP addresses `203.0.113.1 - 203.0.113.2`. Minus the reserved ones, 64512 ports are available to each IP address.

Thus, you can allocate 8064 ports to each subscriber:

Inside Address	Outside Address & Port
100.64.0.0	203.0.113.1:1024-9087
100.64.0.1	203.0.113.1:9088-17151
100.64.0.2	203.0.113.1:17152-25215
100.64.0.3	203.0.113.1:25216-33279
100.64.0.4	203.0.113.1:33280-41343
100.64.0.5	203.0.113.1:41344-49407
100.64.0.6	203.0.113.1:49408-57471
100.64.0.7	203.0.113.1:57472-65535
100.64.0.8	203.0.113.2:1024-9087
100.64.0.9	203.0.113.2:9088-17151
100.64.0.10	203.0.113.2:17152-25215
100.64.0.11	203.0.113.2:25216-33279
100.64.0.12	203.0.113.2:33280-41343
100.64.0.13	203.0.113.2:41344-49407
100.64.0.14	203.0.113.2:49408-57471
100.64.0.15	203.0.113.2:57472-65535

9.12.2 Configuration

To configure the NAT according to the example above, do the following:

1. Create a `deterministic` pool with a block size of 8064:

   ```
   nat pool deterministic-pool
    range 203.0.113.1 203.0.113.2
    type deterministic block-size 8064
    enable
   ```

   These commands are described in detail in the *Pools* section.

2. Create a subscriber group and configure it to use this pool:

   ```
   nat subscriber-group deterministic-group
    pool deterministic-pool
   ```

   These commands are described in detail in the *Subscriber Groups* section .

3. Create a NAT rule for a subnet `100.64.0.0/28`:

```
nat rule subnet 100.64.0.0/28 subscriber-group deterministic-group
```

This command is described in detail in the *Rules* section.

9.12.3 Check

To check the resulting correspondence table, use the following command:

show nat rule subnet A.B.C.D/M [vrf NAME]

show nat64 rule subnet X:X::X:X/M [vrf NAME]

For example, for the above configuration the result would be:

```
nfware# show nat rule subnet 100.64.0.0/28
----------------------------------------
Subscriber          Block
----------------------------------------
100.64.0.0          203.0.113.1:1024-9087
100.64.0.1          203.0.113.1:9088-17151
100.64.0.2          203.0.113.1:17152-25215
100.64.0.3          203.0.113.1:25216-33279
100.64.0.4          203.0.113.1:33280-41343
100.64.0.5          203.0.113.1:41344-49407
100.64.0.6          203.0.113.1:49408-57471
100.64.0.7          203.0.113.1:57472-65535
100.64.0.8          203.0.113.2:1024-9087
100.64.0.9          203.0.113.2:9088-17151
100.64.0.10         203.0.113.2:17152-25215
100.64.0.11         203.0.113.2:25216-33279
100.64.0.12         203.0.113.2:33280-41343
100.64.0.13         203.0.113.2:41344-49407
100.64.0.14         203.0.113.2:49408-57471
100.64.0.15         203.0.113.2:57472-65535
----------------------------------------
```

9.13 Application Layer Gateway

ALG is a feature that allows several applications to work correctly when they pass through the NAT. When an application client sends a private IP address and port in its message, ALG allocates a public IP address and port and translates them in the message. Simply put, ALG does the same thing with application messages as NAT does with the regular IP header. This translation is necessary so that the application server can send a response to a correct public IP address and port.

NAT supports ALG for FTP, TFTP, PPTP, SIP, RTSP, and DNS.

9.13.1 FTP ALG

When using NAT44, the subscriber can use the `passive` FTP mode to work through the NAT with ALG disabled. Otherwise, if the subscriber uses the `active` FTP mode, ALG needs to be enabled. In this case, ALG translates the IP address and port in the `PORT` message.

When using NAT64, ALG must be enabled to allow subscribers to use FTP. In this case, ALG translates the IP address and port in the following messages:

- `EPRT`. In addition to address and port translation, the command itself is changed to `PORT`.
- `EPSV`. The command is changed to `PASV`.
- `227` (response to `PASV`). The command is changed to `229` (response to `EPSV`).

Commands

`<nat|nat64> inspection ftp enable [{control-port (1-65535)|vrf NAME}]`

FTP ALG is disabled by default. This command enables it.

`show <nat|nat64> counters [vrf NAME] alg ftp`

Display FTP ALG counters information.

Counter	Description
`FTP translations`	Translation of internal ip:port to external ip:port and vice versa
`FTP packets dropped`	The number of FTP packets that were dropped
`FTP session entries`	The number of the sessions established at the moment
`FTP session creations`	The number of the sessions established over a whole period of the operation

`clear <nat|nat64> counters [vrf NAME] alg ftp`

Clear FTP ALG counters.

9.13.2 TFTP ALG

TFTP does not send IP addresses in its messages, but it is incompatible with *Address-and-Port-Dependent Filtering* behavior. If this mode is used, TFTP ALG must be enabled to allow users to use TFTP.

Commands

`<nat|nat64> inspection tftp enable [{control-port (1-65535)|vrf NAME}]`

TFTP ALG is disabled by default. This command enables it.

`show <nat|nat64> counters [vrf NAME] alg tftp`

Display TFTP ALG counters information.

Counter	Description
`TFTP translations`	Translation of internal ip:port to external ip:port and vice versa
`TFTP session entries`	The number of the sessions established at the moment
`TFTP session creations`	The number of the sessions established over a whole period of the operation

`clear <nat|nat64> counters [vrf NAME] alg tftp`

Clear TFTP ALG counters.

9.13.3 PPTP ALG

For both NAT44 and NAT64, PPTP ALG must be enabled to allow subscribers to use PPTP. PPTP ALG translates IP address and port in the following messages:

- Outgoing-Call-Request
- Outgoing-Call-Reply
- Call-Clear-Request
- Call-Disconnect-Notify
- WAN-Error-Notify
- Set-Link-Info

Commands

```
<nat|nat64> inspection pptp enable [{control-port (1-65535)|vrf NAME}]
```

PPTP ALG is disabled by default. This command enables it.

```
show <nat|nat64> counters [vrf NAME] alg pptp
```

Display counters for PPTP ALG.

Counter	Description
PPTP translations	Translation of internal ip:port to external ip:port and vice versa
PPTP packets dropped	The number of PPTP packets that were dropped
PPTP outgoing call requests	These requests are PPTP control messages sent by the PNS (refers to the remote client) to the PAC (refers to the server) to indicate that an outbound call from the PAC is to be established. See **RFC 2637#section-2.7**
PPTP call clear requests	Control message indicates that a particular call is to be disconnected. See **RFC 2637#section-2.12** for reference
PPTP outgoing call replies	Control messages from the PAC to the PNS in response to a received Outgoing-Call-Request message. See **RFC 2637#section-2.8** for reference
PPTP call disconnect notifies	Control message from the PAC to the PNS is issued whenever a call is disconnected. See **RFC 2637#section-2.13** for reference
PPTP session entries	The number of the sessions established at the moment
PPTP session creations	The number of the sessions established over a whole period of the operation

```
clear <nat|nat64> counters [vrf NAME] alg pptp
```

Clear PPTP ALG counters.

9.13.4 SIP ALG

Warning: The vast majority of SIP clients support NAT-traversal techniques described in **RFC 6314**, so SIP ALG is not necessary for them. Furthermore, you SHOULD NOT enable SIP ALG unless you have a specific reason to do that because SIP ALG may interfere with NAT traversal techniques.

For both NAT44 and NAT64, SIP ALG translates IP address and port in the following messages:

- REGISTER
- INVITE
- UPDATE
- ACK
- PRACK
- BYE

Commands

```
<nat|nat64> inspection sip enable [{control-port (1-65535)|vrf NAME}]
```

SIP ALG is disabled by default. This command enables it.

```
show <nat|nat64> counters [vrf NAME] alg sip
```

Display SIP ALG counters information.

Counter	Description
`SIP translations`	Translation of internal ip:port to external ip:port and vice versa
`SIP packets dropped`	The number of SIP packets that were dropped
`SIP session entries`	The number of the sessions established at the moment
`SIP session creations`	The number of the sessions established over a whole period of the operation

```
clear <nat|nat64> counters [vrf NAME] alg sip
```

Clear SIP ALG counters.

9.13.5 RTSP ALG

Warning: The vast majority of RTSP clients support NAT-traversal techniques described in **RFC 7604** and **RFC 7825**, so RTSP ALG is not necessary for them. Furthermore, you SHOULD NOT enable RTSP ALG unless you have a specific reason to do that because RTSP ALG may interfere with NAT traversal techniques.

For both NAT44 and NAT64, RTSP ALG translates the IP address and port in `SETUP` messages.

Commands

```
<nat|nat64> inspection rtsp enable [{control-port (1-65535)|vrf NAME}]
```

RTSP ALG is disabled by default. This command enables it.

`show <nat|nat64> counters [vrf NAME] alg rtsp`

Display RTSP ALG counters information.

Counter	Description
`RTSP translations`	Translation of internal ip:port to external ip:port and vice versa
`RTSP packets dropped`	The number of RTSP packets that were dropped
`RTSP setup messages`	These messages are used to specify the transport mechanism for the streamed media. See **RFC 2326#section-10.4** for reference
`RTSP session entries`	The number of the sessions established at the moment
`RTSP session creations`	The number of the sessions established over a whole period of the operation

`clear <nat|nat64> counters [vrf NAME] alg rtsp`

Clear RTSP ALG counters.

9.13.6 DNS ALG

When using NAT44, DNS ALG is not necessary for the correct work of DNS protocol because it does not use private IP addresses in its messages. However, when you enable DNS ALG, it tracks DNS requests sent by subscribers and immediately deletes the session when the corresponding DNS response is received. This allows to significantly reduce the number of concurrent sessions in the NAT session table.

When using NAT64, DNS ALG is necessary to process DNS requests sent by subscribers through the NAT. It translates `AAAA` requests into `A` requests and `A` responses into `AAAA` responses.

> **Warning:** The correct *network architecture for NAT64* involves using a separate DNS64 network element that processes all DNS requests from subscribers. In this case, no DNS requests pass through the NAT, and DNS ALG is not necessary.

Commands

`<nat|nat64> inspection dns enable [{control-port (1-65535)|vrf NAME}]`

DNS ALG is disabled by default. This command enables it.

`show <nat|nat64> counters [vrf NAME] alg dns`

Display DNS ALG counters information.

Counter	Description
`DNS translations`	Translation of internal ip:port to external ip:port and vice versa
`DNS reply packets`	Display the number of the reply packets
`DNS oversized packets`	DNS packets consider oversized when the TC flag (1 bit) is set in the DNS header. This flag is set in the reply packet if the server could not put all the necessary information in the packet because of restrictions
`DNS amplificatic packets`	Shows how many requests related to DNS amplification were dropped
`DNS invalid packets`	This counter will increment when the security appliance detects an invalid DNS packet. For example, a DNS packet with no DNS header, the number of DNS resource records not matching the counter in the header, etc.
`DNS session entries`	The number of the sessions established at the moment
`DNS session creations`	The number of the sessions established over a whole period of the operation

```
clear <nat|nat64> counters [vrf NAME] alg dns
```

Clear DNS ALG counters.

9.13.7 Additional Considerations

Subscribers behind the NAT may experience issues with their VPN connections when using IPsec. It happens because IPsec uses ESP (Encapsulating Security Payloads) as an underlying protocol, and its payload is encrypted, so it is not possible to implement an ALG that would translate IP/TCP headers inside the ESP header.

To solve this problem, subscribers should enable NAT-traversal in their IPsec VPN clients. The vast majority of them support this functionality as described in **RFC 3715** and **RFC 3947**.

9.14 Static Mappings

Static mapping provides access from the Internet to hosts or services in the internal network. There are two types of static mappings available:

IP-IP mapping

This mapping makes the internal resource fully accessible by all protocols and ports. It may be used when the resource runs applications that use ephemeral ports which are not known in advance.

IP:port-IP:port mapping

This mapping makes the internal resource accessible only by specific protocol and port. It should be used when you know exactly which port is used by an application running on the internal resource.

Commands

```
nat static int-ip A.B.C.D ext-ip A.B.C.D
```

Create a static IP-IP mapping. External IP address must already exist as part of any pool of type "NAT".

`nat static proto <tcp|udp> int-ip A.B.C.D int-port 1-65535 ext-ip A.B.C.D ext-port 1-65535`

Create a static IP:Port-IP:Port mapping. External IP address must already exist as part of any pool of type “NAPT”.

`show <nat|nat64> static`

Display the configured static mappings.

9.14.1 Usage Example

1. Create a NAT pool (see *Pools* section) and assign a range of the external IP addresses:

```
nat pool test
 range 203.0.113.1 203.0.113.127
 type nat
 enable
```

2. Create a subscriber group and configure it to use this pool:

```
nat subscriber-group default-group
 pool test
```

3. Create a NAT rule for a subnet 192.168.3.0/28:

```
nat rule subnet 192.168.3.0/28 subscriber-group default-group
```

4. Set a static IP-IP mapping:

```
nat static int-ip 192.168.3.3 ext-ip 203.0.113.1
```

5. Or set a static IP:Port-IP:Port mapping. The configuration for NAPT will be the same. The NAT type line will not be displayed because it is used by default:

```
nat static proto tcp int-ip 192.168.3.3 int-port 3 ext-ip 203.0.113.1 ext-port 171
```

9.15 Show and Clear Commands

Below you see a list of `show` and `clear` commands broken down into blocks. There is a description for each show command. `Clear` commands with the meaning obvious from the command itself are listed without a description.

`show <nat|nat64> counters [vrf NAME]`

Display the information about the current system load (Counter, Current value, Limit, and Load as a percentage). If you have several NAT instances configured in different VRFs, you can view the counters in each VRF separately by specifying its NAME. The counters are:

Counter	Description
`Active Subscribe`	Active internal IP addresses
`Address Map Entries`	Each entry has one unique internal IP address matching the unique external one
`Port Map Entries`	The internal IP:Port mappings to external IP:Port. Mappings is used in NAT sessions creation process
`Session Entries`	A session is an internal data structure which is needed for implementation of the mapping from an internal IP:Port to an external IP:Port. The session contains the following information: int_ip:port ext_ip:port dst_ip:port, the time it was last used
`Pending Fragments`	The IP fragments for all chains that are awaiting assembly

show <nat|nat64> counters [vrf NAME] overall

clear <nat|nat64> counters [vrf NAME] overall

Display the information for all NAT|NAT64 counters. These counters are accumulative and do not show the current load of the system. Below are some counters and their brief description:

Counter	Description
`Inbound and Outbound Translations`	Translations r
`Inbound and Outbound Bytes`	All traffic thr
`Inbound and Outbound Fragment Translations`	All IP fragme
`Inbound and Outbound Fragment Bytes`	The same as a
`Subscriber Creations, Address Map Creations, Port Map Creations and Session Creations`	These counte
`Hairpinning Sessions`	See **RFC 512**
`Filtering Policy Drops`	Filtering mod
`Hairpinning Loop Drops`	See **RFC 614**
`Hairpinning Drops`	The hairpinni
`No Pool Drops`	Packet drops
`No Portless Mapping Drops`	A packet cam
`ACL Drops`	A packet drop
`Inbound Refresh Drops`	A packet cam
`Unsupported L4 Protocol Drops`	A packet with
`No NAT Rule Drops`	A packet cam
`No Mapping Drops`	A packet cam
`No RSS Drops`	RSS was not
`Fragment Timeout Drops`	Until the first
`Fragment Duplicated Drops`	A fragment d
`Fragment Overlap Drops`	A fragment o
`Fragment With Zero Size Drops`	The IP fragm
`Fragment Control Queue Too Short Drops`	There is an a
`TCP No SYN Drops`	The TCP pac
`TCP NULL Flags Drops`	The TCP pac
`TCP SYN & FIN Drops`	The packets w
`TCP XMAS Drops`	The TCP pac
`TCP SYN Fragments Drops`	The TCP SY
`TCP/UDP Port Zero Drops`	The TCP/UD

Counter	Description
`ICMP Query ID Zero Drops`	The ICMP pa
`ICMP Unsupported Proto Drops`	The ICMP er
`ICMP Unknown Type Drops`	The ICMP pa
`ICMP Error Drops`	No matching
`GRE Unknown Version Drops`	The GRE pac

Limit drops counters mean that there was an attempt to exceed the specified limit. In this case we drop the entry/packet/etc., and add 1 to the counter. The limits are:

Limit Drop Counters	Description
`Address Map Entry Limit Drops`	IP-IP NAT mappings
`Session Entry Limit Drops`	NAT sessions (here entry includes Internal, External and Remote IP:Port)
`Subscriber Limit Drops`	Internal IP (subscribers) for NAT
`Pending Fragments Limit Drops`	Fragments currently being processed for the NAT
`Address Map Failure Drops`	See RFC 7659#section-3.1.4 for reference: • there are no free IP addresses in the pool • the system has received a request to allocate the IP address which is already busy
`Port Map Failure Drops:`	This counter includes counters connected to Port Limit Drops:
• `Port Map Entry Limit Drops`	IP:Port - IP:Port NAT mappings
• `Subscriber Port Map Limit Drops`	The limit on the number of ports allocated to one user has been exceeded. The limit is set via `limits port-map-entries` in the subscriber-group section
• `Subscriber Port Block Limit Drops`	The limit on the number of port blocks allocated to one user has been exceeded. The limit is set via `limits port-block-entries` in the subscriber-group section.
• `No Free Port Drops`	No free ports on one of the external IP addresses in one of the pools (see the output `show nat pool POOL_NAME ip`)
• `No Free Block Drops`	No free port blocks on one of the external IP addresses in one of the pools (see the column `Port Blocks` in the output `show nat pool POOL_NAME ip`)
`Subscriber Session Limit Drops`	Sessions per subscriber
`Subscriber Session Rate Limit Drops`	New sessions per second for a subscriber
`Fragment Chain Limit Drops`	The limit for fragment chain is exceeded. The default value is 24

`show <nat|nat64> counters [vrf NAME] protocols`

`clear <nat|nat64> counters [vrf NAME] protocols`

Display the counter information for each protocol, for example:

```
nfware# show nat counters protocols
------------------------------------------------------------------------------------
↪---------------------
Counter                 ICMP                TCP                 UDP            GRE ␣
↪                ESP
------------------------------------------------------------------------------------
↪---------------------
Port Map Entries        0                   0                   0               0  ␣
↪                 0
Translations Outbound   385                 0                   0               0  ␣
↪                 0
Translations Inbound    385                 0                   0               0  ␣
↪                 0
Port Map Creations      1                   0                   0               0  ␣
↪                 0
Port Map Failure Drops  0                   0                   0               0  ␣
↪                 0
------------------------------------------------------------------------------------
↪---------------------
```

show <nat|nat64> sessions [FILTER]

clear <nat|nat64> sessions [FILTER]

Display the output of the whole translation table with the capability to filter this output by following fields: internal/external/remote IP addresses, protocols, ports, pool, or VRF

Filter	Keys
`proto`	• `icmp` • `tcp` Here you can type tcp key or point one or several additional keys like: – `syn-received` – `established` – `fin-received` – `closing` – `transitory` • `udp` • `gre` • `esp`
`int-ip`	• For NAT type `A.B.C.D` internal IP address • For NAT64 type `X:X::X:X` internal IP address
`int-port`	Choose internal port from range `1-65535`
`ext-ip`	Specify external IP address like `A.B.C.D`
`ext-port`	Choose external port from range `1-65535`
`rem-ip`	Specify remote IP address like `A.B.C.D`
`rem-port`	Choose remote port from range `1-65535`
`pool`	Specify pool `NAME`
`vrf`	Specify vrf `NAME`

Full format of the commands:

```
show nat64 sessions [{proto <icmp|tcp [{syn-received|established|fin-received|closing|transitory}]|
B.C.D|ext-port (1-65535)|rem-ip A.B.C.D|rem-port (1-65535)|pool NAME|vrf NAME}]
```

```
show nat sessions [{proto <icmp|tcp [{syn-received|established|fin-received|closing|transitory}]|ud
B.C.D|int-port (1-65535)|ext-ip A.B.C.D|ext-port (1-65535)|rem-ip A.B.C.
D|rem-port (1-65535)|pool NAME|vrf NAME}]
```

```
clear nat64 sessions [{proto <icmp|tcp [{syn-received|established|fin-received|closing|transitory}]
B.C.D|ext-port (1-65535)|rem-ip A.B.C.D|rem-port (1-65535)|pool NAME|vrf NAME}]
```

```
clear nat sessions [{proto <icmp|tcp [{syn-received|established|fin-received|closing|transitory}]|u
B.C.D|int-port (1-65535)|ext-ip A.B.C.D|ext-port (1-65535)|rem-ip A.B.C.
D|rem-port (1-65535)|pool NAME|vrf NAME}]
```

Examples:

```
nfware# show nat sessions int-port 22
------------------------------------------------------------------------------
Protocol  Internal                External                Remote
------------------------------------------------------------------------------
icmp      192.168.1.2:22        12.13.14.15:603      5.5.5.5:603
------------------------------------------------------------------------------

nfware# show nat sessions proto udp int-ip 192.168.1.24 int-port 58448 rem-ip 6.6.6.
↪6 pool TEST
------------------------------------------------------------------------------
Protocol  Internal                External                Remote
------------------------------------------------------------------------------
udp       192.168.1.24:58448      12.13.14.20:5296      6.6.6.6:29375
------------------------------------------------------------------------------

nfware# show nat sessions proto tcp transitory int-ip 192.168.1.25 rem-ip 6.6.6.6
------------------------------------------------------------------------------
Protocol  Internal                External                Remote
------------------------------------------------------------------------------
tcp (t)   192.168.1.25:58448      12.13.14.18:5296      6.6.6.6:443
------------------------------------------------------------------------------
```

show <nat|nat64> sessions [vrf NAME] STRING...

clear <nat|nat64> sessions [vrf NAME] STRING...

Display the detailed information about the session, specified by the full session key. The key can be taken from the output of the `show nat sessions` command. For example:

```
nfware# show nat sessions icmp      192.168.3.4:5          203.0.113.34:119      ␣
↪172.20.1.2:119
Key: proto icmp int-ip 192.168.3.4 int-port 5 ext-ip 203.0.113.34 ext-port 119 r
em-ip 172.20.1.2 rem-port 119
Direction: outbound
Time to live: 0h 0m 59s
```

show <nat|nat64> counters [vrf NAME] rate

Display counters rate. If you have several NAT instances configured in different VRFs, you can view the counters rate in each VRF separately by specifying its NAME.

show nat64 fragment [{src-ip X:X::X:X | dst-ip X:X::X:X | vrf NAME}]

show nat fragment [{src-ip A.B.C.D | dst-ip A.B.C.D | vrf NAME}]

Display the IP packet fragmentation table for NAT|NAT64. You can specify filters like source/destination IP addresses and VRF. vCGNAT assembles IP packet fragments into a chain until it waits for the first fragment or until the chain lifetime expires (15 seconds).

show <nat|nat64> mappings [FILTER]

Display the internal to external IP:Port mapping table for NAT|NAT64 with the capability to filter the output by the following fields: internal/external IP addresses, protocols, ports or VRF. Also, the type of mapping is shown:

- *d* for dynamic translation,
- *s* for static one,
- *pcp* which means that subscribers ask NAT to open ports themselves via PCP protocol.

For dynamic mapping the entry is presented in the table as long as at least one session is alive. The mapping will be deleted as soon as the last session is closed. Static mappings do not age out. PCP mappings live for the requested time, but it is limited to 1 hour. You cannot open a port for more than 1 hour via PCP protocol.

Filter	Keys
`proto`	• `icmp` • `tcp` • `udp` • `gre` • `esp`
`int-ip`	• for NAT type `A.B.C.D` internal IP address • for NAT type `X:X::X:X` internal IP address
`int-port`	Choose internal port from the range `1-65535`
`rem-ip`	Specify remote IP address like `A.B.C.D`
`rem-port`	Choose remote port from range `1-65535`
`vrf`	Specify vrf `NAME`

Full format of the commands:

show nat64 mappings [{proto <icmp|udp|tcp|gre>|int-ip X:X::X:X|int-port (1-65535)|rem-ip X:X::X:X|r

show nat mappings [{proto <icmp|udp|tcp|gre>|int-ip A.B.C.D|int-port (1-65535)|rem-ip A.B.C.D|rem-port (1-65535)|vrf NAME}]

Examples:

```
nfware# show nat mappings
---------------------------------------------------------------
Type  Protocol  Internal              Remote
---------------------------------------------------------------
d     icmp      192.168.1.2:48386     12.13.14.16:17696
s     tcp       192.168.1.3:1016      12.13.14.17:79
pcp   udp       192.168.1.3:3718      12.13.14.18:39530
```

(continues on next page)

(continued from previous page)

```
nfware# show nat mappings proto icmp int-ip 192.168.1.2 rem-ip 12.13.14.16
--------------------------------------------------------------
Type  Protocol  Internal               Remote
--------------------------------------------------------------
d     icmp      192.168.1.2:48386     12.13.14.16:17696
```

CHAPTER

TEN

MONITORING

10.1 SNMP

10.1.1 Configuration

`service snmp enable`

Enable the SNMP protocol support.

`service snmp community COMMUNITY`

Add a read-only community for SNMP requests.

`service snmp host A.B.C.D COMMUNITY (1-65535)`

Add an SNMP traps recipient.

10.1.2 Supported MIBs

The following standard MIBs are supported:

`IF-MIB` according to the **RFC 2863**.

`NATV2-MIB` according to the **RFC 7659**.

The following NFWare MIBs are supported:

`NFWARE-SMI-MIB` for the structure of management information.

`NFWARE-NATV2-MIB` for extenstions to the standard NATV2-MIB.

`NFWARE-RESOURCES-MIB` for monitoring various system resources like CPU and memory.

`NFWARE-CP-MIB` for monitoring software version number.

10.2 Zabbix

The following template for zabbix version 6.0 is used:

`NFWare vCGNAT Template`

10.3 Memory Monitoring

There are two types of memory:

- System memory used by dynamic routing protocols, SSH, SNMP, etc.
- Hugepages for Data Plane. Most memory from hugepages is allocated statically (Sessions, Mappings, Subscribers, Hash Tables), but there is also memory, which is allocated during usage (when creating pools, VRFs, or ACLs, for example).

To monitor memory correctly, we recommend using the following OIDs:

1. Total System Memory = All Memory - All Hugepages

```
.1.3.6.1.4.1.2021.4.5.0 - Total System Memory
.1.3.6.1.4.1.46576.1.2.0.2.1.2.0 - HugePages Total (NUMA node0)
.1.3.6.1.4.1.46576.1.2.0.2.1.2.1 - HugePages Total (NUMA node1)

.1.3.6.1.4.1.2021.4.5.0 - (.1.3.6.1.4.1.46576.1.2.0.2.1.2.0 + .1.3.6.1.4.1.46576.1.2.0.2.
↪1.2.1)
```

2. Free system memory: Free + Buffered + Cached:

```
.1.3.6.1.4.1.2021.4.11.0 + .1.3.6.1.4.1.2021.4.14.0 + .1.3.6.1.4.1.2021.4.15.0
```

3. Memory in vCGNAT is divided by NUMA nodes. It is better to monitor it separately:

```
Total Hugepages NUMA node0: .1.3.6.1.4.1.46576.1.2.0.2.1.2.0
Total Hugepages NUMA node1: .1.3.6.1.4.1.46576.1.2.0.2.1.2.1
```

4. Used memory:

```
NUMA node0: .1.3.6.1.4.1.46576.1.2.0.2.1.4.0
NUMA node1: .1.3.6.1.4.1.46576.1.2.0.2.1.4.1
```

5. Free memory:

```
NUMA node0: .1.3.6.1.4.1.46576.1.2.0.2.1.3.0
NUMA node1: .1.3.6.1.4.1.46576.1.2.0.2.1.3.1
```

Alternatively, you can use a command via CLI to determine how much memory is used:

`show memory dataplane`

Display Dataplane memory information for all sockets. The output shows the vCGNAT objects and their size in Bytes. Their total size is given at the bottom of the output. The summary statistic gives the total amount of memory allocated by vCGNAT, the memory used, the free memory, and the load percentage.

Another useful command is:

`show memory dataplane pools`

Display memory statistics for memory pools for all sockets. When vCGNAT is started, memory pools are allocated statically. The output shows the total, used, and free number of subscribers, mappings, sessions, etc. This command can be used to monitor memory pool utilization. If vCGNAT cannot, for example, allocate sessions from the memory pool, drops will appear in the corresponding column. The critical load value is 90%. The total number of subscribers, mappings, sessions, etc., can be changed in the platform settings. The output is as follows:

```
vcgnat# show memory dataplane pools
------------------------------------------------------------------------------------
↪---
Statistics for Socket 0
------------------------------------------------------------------------------------
↪---
                       Total         Used          Free          Drops            ␣
↪Load
------------------------------------------------------------------------------------
↪---
Subscribers            1527648       161520        1366128       0               ␣
↪10.6%
Mappings               85027648      1096285       83931363      0                ␣
↪1.3%
Sessions               85027648      1164505       83863143      0                ␣
↪1.4%
Fragments              65536         0             65536         0                ␣
↪0.0%
Pending Fragments      1024          2             1022          0                ␣
↪0.2%
------------------------------------------------------------------------------------
↪---
------------------------------------------------------------------------------------
↪---
Statistics for Socket 1
------------------------------------------------------------------------------------
↪---
                       Total         Used          Free          Drops            ␣
↪Load
------------------------------------------------------------------------------------
↪---
Subscribers            1527648       144786        1382862       0                ␣
↪9.5%
Mappings               85027648      970063        84057585      0                ␣
↪1.1%
Sessions               85027648      1030252       83997396      0                ␣
↪1.2%
Fragments              65536         4             65532         0                ␣
↪0.0%
Pending Fragments      1024          2             1022          0                ␣
↪0.2%
------------------------------------------------------------------------------------
↪---
------------------------------------------------------------------------------------
↪---
                       Total         Used          Free          Drops            ␣
↪Load
------------------------------------------------------------------------------------
↪---
IPv4 neighbors         1024          12            1012          0                ␣
↪1.2%
IPv6 neighbors         1024          0             1024          0                ␣
↪0.0%
```

(continues on next page)

(continued from previous page)

```
ARP wait_ctx 0         69632         15            69617         0                ␣
↪0.0%
ARP wait_ctx 1         69632         15            69617         0                ␣
↪0.0%
------------------------------------------------------------------------------------
↪---
```

CHAPTER

ELEVEN

LLDP

11.1 Configuration

The following commands are available to configure LLDP:

`lldp enable`

Enable the support for the LLDP protocol.

`lldp hostname HOSTNAME...`

Set the system name that is passed to the neighbors. By default, the system name is passed, configured with the *`hostname NAME`* command.

`lldp description DESCRIPTION...`

Set the system description that is passed to the neighbors. By default, the string *NFWare virtual Carrier Grade NAT* is passed.

`lldp portidsubtype <mac|name>`

Set the type of the interface identifier that is passed to the neighbors. By default, the MAC address is used.

`lldp tx-interval (1-3600)`

Set the interval between sending the LLDP PDUs in seconds. By default, the message is sent once every 30 seconds.

`lldp hold-multiplier (2-10)`

Set the LLDP hold miltiplier. This parameter is used to calculate the LLDP TTL. The LLDP TTL is calculated as the product of the `tx-interval` and the `hold-multiplier`. By default, this parameter is set to 4, so the default LLDP TTL is 120 seconds.

`lldp update`

Send the LLDP PDU without the `tx-interval` delay. The command can be used to immediately send updated information about the system to your neighbors after a configuration change.

11.2 Information and Statistics

`show lldp neighbors [{interface IFNAME|view <detailed|summary>}]`

Show information about the LLDP-neighbors.

`interface IFNAME` – show only the information for the `IFNAME` interface.

`view summary` – show a brief summary.

`view detailed` – show detailed information.

show lldp interfaces [{interface IFNAME|view <detailed|summary>}]

Show information about the local interfaces.

`interface IFNAME` – show only the information for the interface `IFNAME`.
`view summary` – show brief information.
`view detailed` – show detailed information.

show lldp statistics [<interface IFNAME|summary>]

Show statistics of sent and received LLDP PDUs.

`interface IFNAME` – show only the information for the interface `IFNAME`.
`summary` – show summary statistics for all interfaces.

CHAPTER

TWELVE

RELEASE NOTES

12.1 Release 4.2

12.1.1 Changes

- We now provide offline licenses. With this, connection to our licensing server is no longer needed.
- There is no longer a separate logging interface. Now you must use data interfaces to send logs. For example, if previously you had four interfaces in the VM: virtual interface for management, virtual interface for logging, and two physical PCI-passthrough interfaces for data, now you will have only three interfaces: one a virtual interface for management and two physical PCI-passthrough interfaces for data and logging.

  The typical way to configure logging without spending the whole physical interface on it is to use separate VLAN and VRF for logging traffic, for example:

  ```
  vrf nat-log
  !
  interface if0.100
  vrf nat-log
  ip address 10.0.0.1/24
  !
  nat log server 0 type netflow ip 10.0.0.2 port 2055 vrf nat-log
  nat log type session enable
  nat log enable
  ```

  With the configuration above, you have a separate virtual interface with VLAN tag 100 for sending logs. This interface is in a separate VRF named “nat-log” to prevent routing collisions with the default VRF where customer traffic is routed.

  Additionally, there is now a possibility to configure and use multiple logging servers simultaneously. Therefore, you need to specify a server ID when configuring it. For example, if you had the configuration line `nat log server type syslog ip 1.1.1.1 port 514` you must change it to `nat log server 0 type syslog ip 1.1.1.1 port 514`. And you may configure additional log servers, even using a different protocol, if you need to, for example, `nat log server 1 type ipfix ip 2.2.2.2 port 4739`.

12.1.2 New Features

- VRF support.
- BFD (Bidirectional Forwarding Detection) protocol support.
- Ability to send logs to multiple servers.
- Blackhole routes.
- Mellanox 100 GbE NICs support.
- On-the-fly reconfiguration of pools (add/remove IPs).
- TCP implementation complied with RFC 7857.
- Added ICMP errors sending (NAT is visible in traceroute now).
- Random public IP selection on session creation.
- Autocomplete and search in CLI.
- Changed logging format (all log types now have VRF field and syslog compatibility with RFC 5424).

12.2 Release 4.3

12.2.1 New Features

- Added support NAT between different VRFs.
- Added support for vmxnet3 driver (ESXi 6.7 is the minimum supported version).

12.2.2 Bug Fixes

- Fixed packets discards statistics on Mellanox NICs.
- Fixed processing of ICMP/ICMPv6 errors with embedded ICMP/ICMPv6 in NAT64.
- Fixed processing of ICMP/ICMPv6 errors with embedded IP/IPV6 fragments.
- Fixed processing of fragmented ICMP/ICMPv6 packets in NAT64.
- Fixed changing of static route type between blackhole and reject.
- Fixed source address for ICMPv6 errors.
- Fixed crash when using aggregate-address with route-map in BGP.
- Fixed crash when removing ISIS router configuration with active neighbors.
- Fixed updating BFD Detection Timeout when neighbor changes its Detection Multiplier.
- Fixed crash when creating an OSPF virtual link in an unknown VRF.
- Various fixes and improvements in CLI.

12.3 Release 4.3.1

12.3.1 Bug Fixes

- Fixed VRF counters for locally generated packets.
- Moved local check to icmp error node.
- Added missing newline in exit message.
- Fixed punting fragmented packets.

12.4 Release 4.3.2

12.4.1 Bug Fixes

- Fixed conversion to kilobytes.
- Fixed object name in control-plane MIB.
- Fixed VRF counters for locally generated packets.
- Added missing newline in exit message.
- Fixed punting fragmented packets.

12.5 Release 5.0.0

12.5.1 New Features

- Improved performance and capacity of ARP subsystem.
- Added VRRP version 3 support for IPv4/IPv6.
- Added Real Time Management (RTM) subsystem.
- Added sessions synchronization.
- Added support for Mellanox ConnectX-6 and Intel E810.
- Added support for TACACS+ and RADIUS in AAA subsystem.

12.6 Release 6.0.0

12.6.1 New Features

- Added pcap dumping of filtered packets and new debug subsystem for packets tracking. Now it is possible to trace packets from/to the subscribers using ACL that will help to see which vCGNAT systems a custom packet has passed through
- Added new statistics for tables: `natv2XPoolTable`, `natv2XProtocolTable` and `natv2XInstanceTable`. New counters for `natv2XInstanceTable` are:
  - `natv2InstanceNoFreePortDrops`

  - natv2InstancePortMapSubscriberLimitDrops
  - natv2InstanceNoFreeBlockDrops
  - natv2InstancePortBlockSubscriberLimitDrops
- Added ability to pass license as base64-encoded string
- Added nat log groups and basic nat log balancing
- Added flow label and dscp filtering in ACL
- Added wait queues for packets and support for RDNSS in RA for Neighbor discovery subsystem
- Allowed transparent forwarding of hairpinning packets
- Splited `Port Map Failure Drops` counter
- Added `Port Blocks` counter for external IP address

12.6.2 Bug Fixes

- Fixed command `no nat log group`
- Fixed changing `netflow/ipfix template-resend-timeout`
- Fixed passthrough action in ACL
- Fixed crash adding deterministic rule
- Fixed crash at pool enable
- Fixed TCP sessions state machine
- Fixed sending sync packets
- Fixed stack smashing in `mlx5_xstats_get`
- Fixed Mellanox NICs configuration
- Fixed error processing on hugepages allocation
- Fixed getting hugepages-2048kB information

12.6.3 Updates

- Updated FRR to 8.4.3 version
- Updated cli to 8.4.3 version. Now it checks if the IP address is used in the interface configuration and prevents configuration of the overlapped IP networks inside VRF

12.7 Release 6.1.0

12.7.1 New Features

- Added feature to receive RADIUS accounting messages and parse RADIUS attributes
- Added commands to configure the format of NAT log messages. Now it is possible to include additional fields in NAT logs that contain the values of RADIUS protocol attributes, for example, MSISDN

12.7.2 Bug Fixes

- Fixed critical bugs in RIB subsystem and in nexthop groups subsystem
- Fixed nat log group configuration
- Fixed deadlock in CLI subsystem
- Fixed deadlock in IP (sync) subsystem

12.7.3 Updates

- Updated FRR to 8.4.4 version

12.8 Release 6.2.0

12.8.1 New Features

- Added stateless mode for NAT pools. In this mode, NAT doesn't track state for TCP sessions. It allows to use two instances in active-active mode with duplicate static NAT rules on both vCGNATs.
- Added TCP FIN-WAIT timeout
- Added search by prefix to "show/clear nat sessions" command
- Added support for displaying SFF-8436 and SFF-8636 transceiver information
- Added AAA advanced local mode

12.8.2 Changes

- Allowed NAT session creation in case of disabled inbound-refresh behavior
- Delete subscribers without timeout after receiving RADIUS Accounting Stop message
- Improve performance in route installation to FIB
- Do not write sensitive input to history file

12.8.3 Bug Fixes

- Fixed resend timeout in NAT log flow
- Fixed reading startup-config with loopbacks
- Fixed NAT log group removal. Now, when a NAT log group is removed, all other settings related to this group will be removed too
- Fixed the inability to create a custom syslog format for NAT logs if RADIUS Accounting messages are disabled, even if no fields from these messages are used
- Fixed `pci_buffer_overflow` counter for VLAN and BOND interfaces
- Fixed sending gratuitous ARP requests to Linux kernel. After processing these packets in our Data Plane, they are sent to the Linux kernel, so it also recognizes a new MAC address
- Fixed deletion of subscriber with activated timer

- Fixed processing empty enable password

12.9 Release 6.3.0

12.9.1 New Features

- Added support of interim NAT logs for PBA pools

12.9.2 Changes

- Decreased default TCP FIN-WAIT timeout to 240 seconds
- Allowed only `-` and `_` special symbols in usernames for local users in AAA advanced mode

12.9.3 Bug Fixes

- Fixed `no enable password` error
- Fixed internal NAT log contexts initialization (memory might be allocated on the wrong NUMA node)

CHAPTER

THIRTEEN

HOW TO

13.1 How To Make vCGNAT Update

1. Copy file to the VM:

```
local# scp vCGNAT_UPDATE.tar.gz admin@vCGNAT_MANAGEMENT_IP:
```

where

- vCGNAT_MANAGEMENT_IP is the IP address assigned to the management interface of the virtual machine.
- vCGNAT_UPDATE.tar.gz is the name of the archive with updates. The archive is provided by us.

2. To install, run the command on the vCGNAT:

```
nfware# unpack /home/admin/vCGNAT_UPDATE.tar.gz
```

3. Restart the vCGNAT:

```
nfware# reboot
```

13.2 How To Renew License

If you want to renew your license due to the end of the expiration date, follow the steps below. Note that you do not need to restart the VM in this case.

1. Obtain the ID of the virtual machine installed, and tell it to your manager:

```
nfware# show license host-id
03196D973B6D24649CB55F18804652B8
```

2. In return, you will receive a license file. Upload it to the VM:

```
local# scp 03196D973B6D24649CB55F18804652B8.license admin@vCGNAT_MANAGEMENT_IP:
```

3. Before using the obtained license file, it is recommended to check its parameters:

```
nfware# show license limits /home/admin/03196D973B6D24649CB55F18804652B8.license
```

4. Apply the license file:

```
nfware# license file /home/admin/03196D973B6D24649CB55F18804652B8.license
```

5. No need to restart

Important: If the license parameters, like the date of expiration or limits, are changed, a reboot of the vCGNAT is required.

13.3 How To Change Platform Limits

1. Check the license limits:

```
nfware# show license limits
```

2. You can check some current platform limits (like nb_cmd, nb_rxtx, nb_work, nb_log, max_subscribers, max_sessions, and etc.) as follows:

```
nfware# show platform settings
```

3. If new limits you want to set are outside of the license limits, then perform the steps above to get new license. After this, go to the step 4. Also, go to the step 4 if new limits are not outside the license ones.

4. Change the vCGNAT limits according to *Platform Configuration* section.

5. Restart the vCGNAT:

```
nfware# reboot
```

13.4 How To Get Debug Report

1. Exclude the command `debug report`:

```
nfware# debug report
Generating file with static information
Generating first file with dynamic information
Waiting 5 seconds
Generating second file with dynamic information
Generating archive with reports
Success! Report file nfware-debug-report-2023-10-17_11-19-28.tar.gz
```

2. It will generate a nfware-debug-report file, which can be downloaded via scp.

```
localhost# scp admin@MANAGEMENT_IP:/home/admin/nfware-debug-report-2023-10-17_11-19-28.
↪tar.gz .
```

CHAPTER

FOURTEEN

TROUBLESHOOTING

14.1 Packet Tracer

Packet tracer allows to see through which vCGNAT subsystems both incoming and outgoing user packets have passed. It is a powerful utility that can help debug non-trivial or difficult-to-analyze issues. Packet tracer can be configured to analyze both user traffic and counter drops.

In common cases, debugging should be performed as follows:

1. Create and enable a debug_pool with one IP.
2. Create a subscriber group and configure it to use this pool.
3. Create a nat rule to process all traffic coming from this subscriber.
4. Create an access list with two precise rules for forward and reverse traffic.

```
ip dp-access-list NAME SEQ ACTION <any|udp|tcp|icmp|gre|esp> src-ip <any|A.B.C.D/M> dst-ip <any|A.B.C.D/M>
```

```
ipv6 dp-access-list NAME SEQ ACTION <any|udp|tcp|icmp|gre|esp> src-ip <any|X:X::X:X/M> dst-ip <any|X:X::X:X/M>
```

Before creating a packet tracer, configure access lists for IPv4 or IPv6 traffic.

```
debug packet-tracer <ip|ipv6> dp-access-list NAME max-packets (2-1048576) [payload-length (16-65535) [<
```

Create a packet tracer with the configured access list.

`max-packets` — the maximum number of packets per *nb_work core* will be stored in the trace. You can write $N^2 - 1$ to the ring queue. For example, if you specify a value of 10, the trace subsystem will take the degree of two from the top $2^4 - 1 = 15$ — that's how many packets will be written to the trace. If 12 nb_work cores are specified in the platform settings, then the total number of packets will be $15 * 12 = 180$. These packets can be viewed using the `show debugging packet-tracer ip traces` command, where the timestamp characterizes the beginning of the packet.

`payload-size` — the amount of data in Bytes saved by packet tracer for each packet

`vrf-all` — packets will be collected over all vrf

Note: Only one packet tracer can be created at any given time.

debug packet-tracer <ip|ipv6> counter <hairpinning_loop_drops|hairpinning_drops| unsupported_l4_proto_di

If it is necessary to analyze a reason of drops in the specific counters (see the output of `show nat counters overall` command), then specify the counter name instead of the access list in the packet tracer. A detailed description of counters can be found *here*.

no debug packet-tracer <ip|ipv6> [<vrf NAME|vrf-all>]

Delete created packet tracer.

debug packet-tracer burst-size (1-65535)

Specify the number of packets that the handler will read at one time from the queue. The default is 512.

debug packet-tracer <ip|ipv6> pcap-dump file FILENAME

Collect traces in a pcap file. When a pcap dump is enabled, a separate thread parses the queue in real-time. A queue here means a buffer - the maximum number of simultaneous packets stored in vCGNAT memory before they hit the disc. In that case, option `max-packets` in the debug packet-tracer command does not limit the number of packets to be collected.

> **Warning: The dump will start writing to the file immediately after the command is executed and will not stop automatically. It must be stopped manually; otherwise, a disc overflow will occur! The access list should be as precise as possible to keep the dump from weighing a lot.**

no debug packet-tracer <ip|ipv6> pcap-dump [file FILENAME]

Stop dump recording.

14.1.1 Show Commands

show debugging packet-tracer <ip|ipv6> traces [clear]

Display traces that were collected by packet tracer. There is such information in every traced packet: ethernet and IP headers, nodes through which packets pass, and where the packets are dropped. If the option `clear` is added, the traces are deleted after viewing. Here is an example of ICMP packet:

```
Timestamp: 3813334005032920
Payload (saved 98 bytes of 98):
 ether src: 0C:72:8B:12:00:00
 ether dst: 0C:5B:1E:11:00:02
 ether type: 800
 ipv4:
  src: 10.0.0.2
  dst: 212.12.12.12
  proto: 1
L3 ip process node:
 next node: NODE_NAT_OUTBOUND_IPv4
 vrf id: 1
 ttl: 63
 hash: 26023cd3
NAT ip outbound node:
 next node: NODE_FORWARD_IPv4
 fragment: 0
 passthrough: 0
 session: 0x16c30b500
  nat type: 1
```

(continues on next page)

(continued from previous page)

```
  translated ip: 192.168.1.1
  translated port: 865
 alg result:
  state: 2
  offset: 0
  expiration tsc: ffffffffffffffff
L3 ip forward node:
 next node: NODE_FINISH_IPv4
 stage: Checks passed
 error: 0
 vrf id: 0
 hash: 26023cd3
 route:
  type: 1
  scope: 0
  mask: 32
  nb nexthops: 1
  ip: 212.12.12.12
  src: 0.0.0.0
  nexthop:
   id: 1048569
   dev: 0
   gateway:
    ip: 10.10.20.2
    neighbor:
     ether: 0C:D4:10:AB:00:01
     state: 2
     last time used: d8c3502c7a7d8
L3 ip finish node:
 next node: NODE_MAX
 error: 0
 vrf id: 0
 mtu: 1500
 hash: 26023cd3
```

`show debugging packet-tracer <ip|ipv6> stats [<vrf NAME|vrf-all>]`

Display statistic for a created packet tracer.

```
Packet Tracer:
  Type: IPv4
  Condition: ACL DUMP
  Max Packets: 128
  Payload-length: 100
  Burst Size: 512
  Traced counters:
    hairpinning_loop_drops: Off
    hairpinning_drops: Off
    unsupported_l4_proto_drops: Off
    no_nat_rule_drops: Off
    no_mapping_drops: Off
    ttl_drops: Off
  Counters:
```

(continues on next page)

(continued from previous page)

```
  Total usage:    19
  No free space:  0
  ACL checks:     19
  Added to trace: 19
  Processed:      0
Pcap dump: disabled
```

INDEX

Symbols