NAT

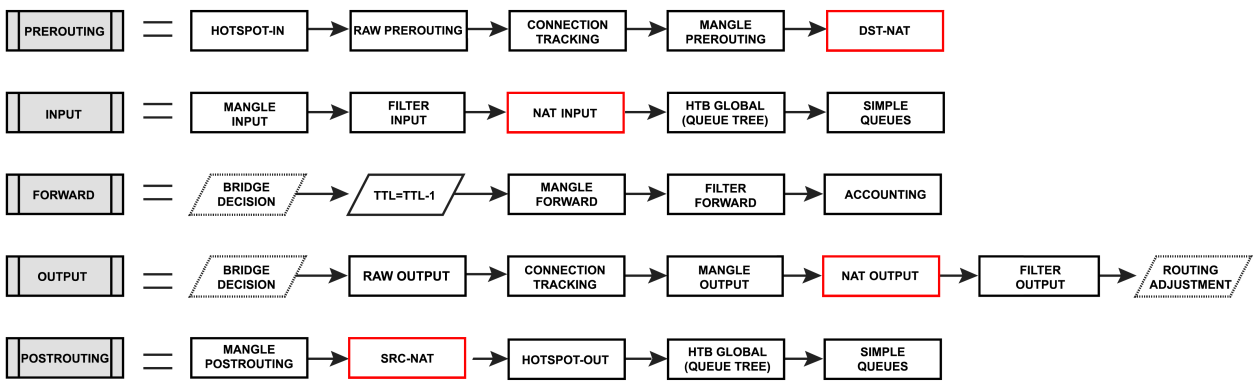
Introduction

Network Address Translation is an Internet standard that allows hosts on local area networks to use one set of IP addresses for internal communications and another set of IP addresses for external communications. A LAN that uses NAT is ascribed as a *natted* network. For NAT to function, there should be a NAT gateway in each *natted* network. The NAT gateway (NAT router) performs IP address rewriting on the way while packets travel from/to LAN. In RouterOS NAT is supported for IPv4. RouterOS does not support NAT64.

Nat matches only the first packet of the connection, connection tracking remembers the action and performs on all other packets belonging to the same connection.

Whenever NAT rules are changed or added, the connection tracking table should be cleared otherwise NAT rules may seem to be not functioning correctly until the connection entry expires.

Types of NAT:



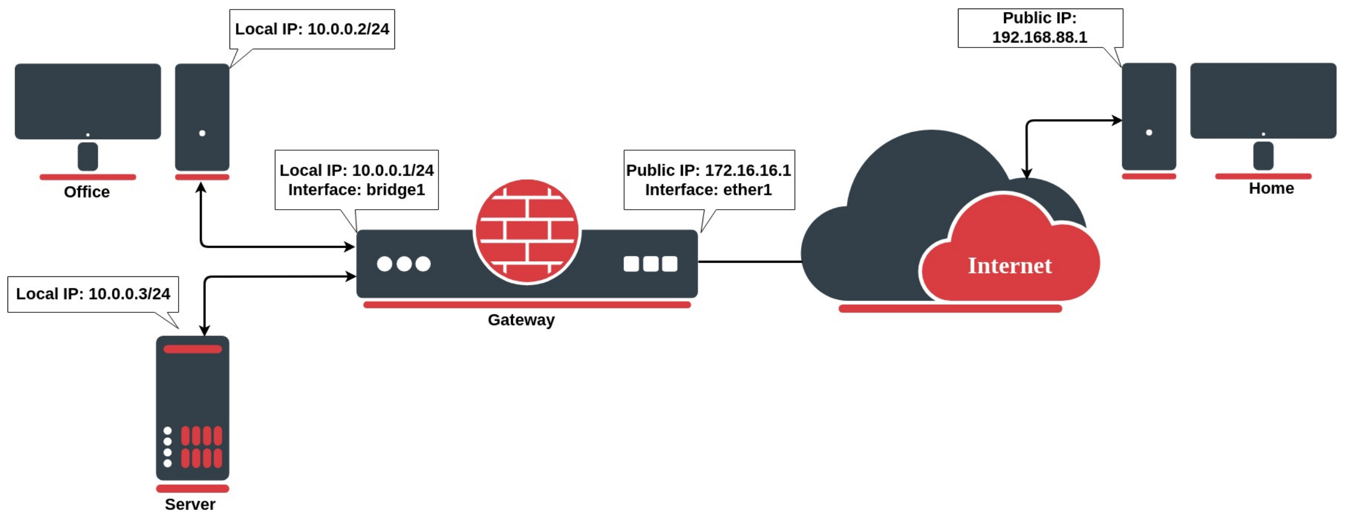
There are two types of NAT:

- **source NAT or srcnat.** This type of NAT is performed on packets that are originated from a natted network. A NAT router replaces the private source address of an IP packet with a new public IP address as it travels through the router. A reverse operation is applied to the reply packets traveling in the other direction.
- **destination NAT or dstnat.** This type of NAT is performed on packets that are destined for the natted network. It is most commonly used to make hosts on a private network to be accessible from the Internet. A NAT router performing dstnat replaces the destination IP address of an IP packet as it travels through the router toward a private network.

Since RouterOS v7 the firewall NAT has two new *INPUT* and *OUTPUT* chains which are traversed for packets delivered to and sent from applications running on the local machine:

- **input** - used to process packets entering the router through one of the interfaces with the destination IP address which is one of the router's addresses. Packets passing through the router are not processed against the rules of the input chain.
- **output** - used to process packets that originated from the router and leave it through one of the interfaces. Packets passing through the router are not processed against the rules of the output chain.

Destination NAT



Network address translation works by modifying network address information in the packet's IP header. Let`s take a look at the common setup where a network administrator wants to access an office server from the internet.

We want to allow connections from the internet to the office server whose local IP is 10.0.0.3. In this case, we have to configure a destination address translation rule on the office gateway router:

```
/ip firewall nat add chain=dstnat action=dst-nat dst-address=172.16.16.1 dst-port=22 to-addresses=10.0.0.3
protocol=tcp
```

The rule above translates: when an incoming connection requests TCP port 22 with destination address 172.16.16.1, use the *dst-nat* action and depart packets to the device with local IP address 10.0.0.3 and port 22.

To allow access only from the PC at home, we can improve our *dst-nat* rule with *"src-address=192.168.88.1"* which is a Home`s PC public (this example) IP address. It is also considered to be more secure!

Source NAT

If you want to hide your local devices behind your public IP address received from the ISP, you should configure the source network address translation (masquerading) feature of the MikroTik router.
Let`s assume you want to hide both the office computer and server behind the public IP 172.16.16.1, the rule will look like the following one:

```
/ip firewall nat add chain=srcnat src-address=10.0.0.0/24 action=src-nat to-addresses=172.16.16.1 out-
interface=WAN
```

Now your ISP will see all the requests coming with IP 172.16.16.1 and they will not see your LAN network IP addresses.

Masquerade

Firewall NAT `action=masquerade` is a unique subversion of `action=srcnat`, it was designed for specific use in situations when public IP can randomly change, for example, DHCP server changes assigned IP or PPPoE tunnel after disconnect gets a different IP, in short - **when public IP is dynamic**

```
/ip firewall nat add chain=srcnat src-address=10.0.0.0/24 action=masquerade out-interface=WAN
```

Every time when interface disconnects and/or its IP address changes, the router will clear all masqueraded connection tracking entries related to the interface, this way improving system recovery time after public IP change. If `srcnat` is used instead of `masquerade`, connection tracking entries remain and connections can simply resume after a link failure.

Unfortunately, this can lead to some issues with unstable links when the connection gets routed over different links after the primary link goes down. In such a scenario following things can happen:

- on disconnect, all related connection tracking entries are purged;
- next packet from every purged (previously masqueraded) connection will come into the firewall as *new*, and, if a primary interface is not back, a packet will be routed out via an alternative route (if you have any) thus creating a new masqueraded connection;
- the primary link comes back, routing is restored over the primary link, so packets that belong to existing connections are sent over the primary interface without being masqueraded, that way leaking local IPs to a public network.

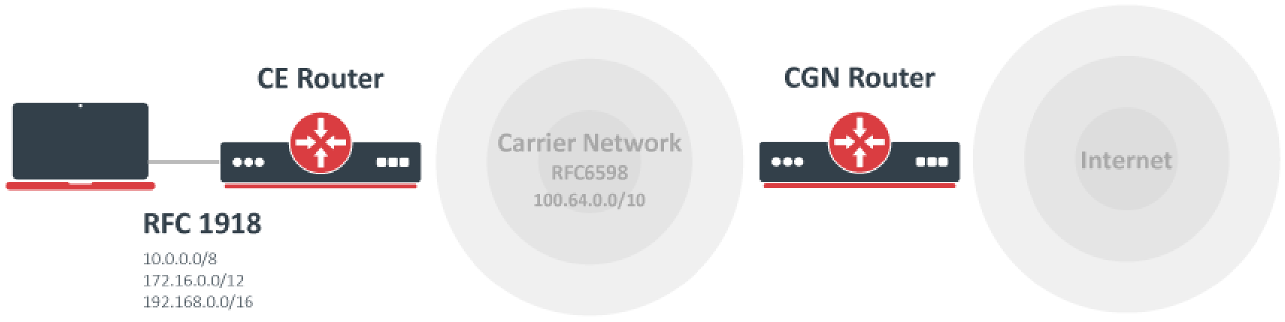
To work around this situation **blackhole** route can be created as an alternative to the route that might disappear on disconnect.

Hosts behind a NAT-enabled router do not have true end-to-end connectivity. Therefore some Internet protocols might not work in scenarios with NAT. Services that require the initiation of TCP connection from outside the private network or stateless protocols such as UDP, can be disrupted.

To overcome these limitations RouterOS includes a number of so-called NAT helpers, that enable NAT traversal for various protocols. When `action=srcnat` is used instead, connection tracking entries remain and connections can simply resume.

Though Source NAT and masquerading perform the same fundamental function: mapping one address space into another one, the details differ slightly. Most noticeably, masquerading chooses the source IP address for the outbound packet from the IP bound to the interface through which the packet will exit.

CGNAT (NAT444)



To combat IPv4 address exhaustion, a new RFC 6598 was deployed. The idea is to use shared 100.64.0.0/10 address space inside the carrier's network and perform NAT on the carrier's edge router to a single public IP or public IP range.

Because of the nature of such a setup, it is also called NAT444, as opposed to a NAT44 network for a 'normal' NAT environment, three different IPv4 address spaces are involved.

CGNAT configuration on RouterOS does not differ from any other regular source NAT configuration:

```
/ip firewall nat
 add chain=src-nat action=srcnat src-address=100.64.0.0/10 to-address=2.2.2.2 out-interface=<public_if>
```

Where:

- 2.2.2.2 - public IP address,
- public_if - interface on provider's edge router connected to the internet

The advantage of NAT444 is obvious, fewer public IPv4 addresses are used. But this technique comes with major drawbacks:

- The service provider router performing CGNAT needs to maintain a state table for all the address translations: this requires a lot of memory and CPU resources.
- Console gaming problems. Some games fail when two subscribers using the same outside public IPv4 address try to connect to each other.
- Tracking users for legal reasons means extra logging, as multiple households go behind one public address.

- Anything requiring incoming connections is broken. While this already was the case with regular NAT, end-users could usually still set up port forwarding on their NAT router. CGNAT makes this impossible. This means no web servers can be hosted here, and IP Phones cannot receive incoming calls by default either.
- Some web servers only allow a maximum number of connections from the same public IP address, as a means to counter DoS attacks like SYN floods. Using CGNAT this limit is reached more often and some services may be of poor quality.
- 6to4 requires globally reachable addresses and will not work in networks that employ addresses with a limited topological span.

Packets with Shared Address Space source or destination addresses MUST NOT be forwarded across Service Provider boundaries. Service Providers MUST filter such packets on ingress links. In RouterOS this can be easily done with firewall filters on edge routers:

```
/ip firewall filter
 add chain=input src-address=100.64.0.0/10 action=drop in-interface=<public_if>
 add chain=output dst-address=100.64.0.0/10 action=drop out-interface=<public_if>
 add chain=forward src-address=100.64.0.0/10 action=drop in-interface=<public_if>
 add chain=forward src-address=100.64.0.0/10 action=drop out-interface=<public_if>
 add chain=forward dst-address=100.64.0.0/10 action=drop out-interface=<public_if>
```

Service providers may be required to log of MAPed addresses, in a large CGN deployed network which may be a problem. Fortunately, RFC 7422 suggests a way to manage CGN translations in such a way as to significantly reduce the amount of logging required while providing traceability for abuse response.

RFC states that instead of logging each connection, CGNs could deterministically map customer private addresses (received on the customer-facing interface of the CGN, a.k.a., internal side) to public addresses extended with port ranges.

That means that separate NAT rules have to be added to achieve individual mappings such as the ones seen in the below example:

Inside IP	Outside IP/Port range
100.64.0.1	2.2.2.2:5000-5199
100.64.0.2	2.2.2.2:5200-5399
100.64.0.3	2.2.2.2:5400-5599
100.64.0.4	2.2.2.2:5600-5799
100.64.0.5	2.2.2.2:5800-5999

Instead of writing the rules by hand, it is suggested to use a script instead. The following example could be adapted to any requirements of your setup.

```
{
######## Adjustable values #########
:local StartingAddress 100.64.0.1
:local ClientCount 5
:local AddressesPerClient 2
:local PublicAddress 2.2.2.2
:local StartingPort 5000
:local PortsPerAddress 200
####################################

# All client chain jump
/ip firewall nat add chain=srcnat action=jump jump-target=clients \
    src-address="$StartingAddress-$($StartingAddress + ($ClientCount * $AddressesPerClient) - 1)"

:local currentPort $StartingPort

:for c from=1 to=$ClientCount do={
    # Specific client chain jumps
    :if ($AddressesPerClient > 1) do={
      /ip firewall nat add chain=clients action=jump jump-target="client-$c" \
      src-address="$($StartingAddress + ($AddressesPerClient * ($c - 1)))-$($StartingAddress +
($AddressesPerClient * $c -1))"
    } else={
      /ip firewall nat add chain=clients action=jump jump-target="client-$c" \
      src-address="$($StartingAddress + ($AddressesPerClient * ($c - 1)))"
    }
    # Translation rules
    :for a from=1 to=$AddressesPerClient do={
      /ip firewall nat add chain="client-$c" action=src-nat protocol=tcp \
      src-address="$($StartingAddress + (($c -1) * $AddressesPerClient) + $a - 1)" to-address=$PublicAddress to-
ports="$currentPort-$($currentPort + $PortsPerAddress - 1)"
      /ip firewall nat add chain="client-$c" action=src-nat protocol=udp \
      src-address="$($StartingAddress + (($c -1) * $AddressesPerClient) + $a - 1)" to-address=$PublicAddress to-
ports="$currentPort-$($currentPort + $PortsPerAddress - 1)"
      :set currentPort ($currentPort + $PortsPerAddress)
    }
}
}
```

The six local values can be adjusted and the script can be either simply pasted in the terminal or it can be stored in the system script section, in case the configuration needs to be regenerated later.

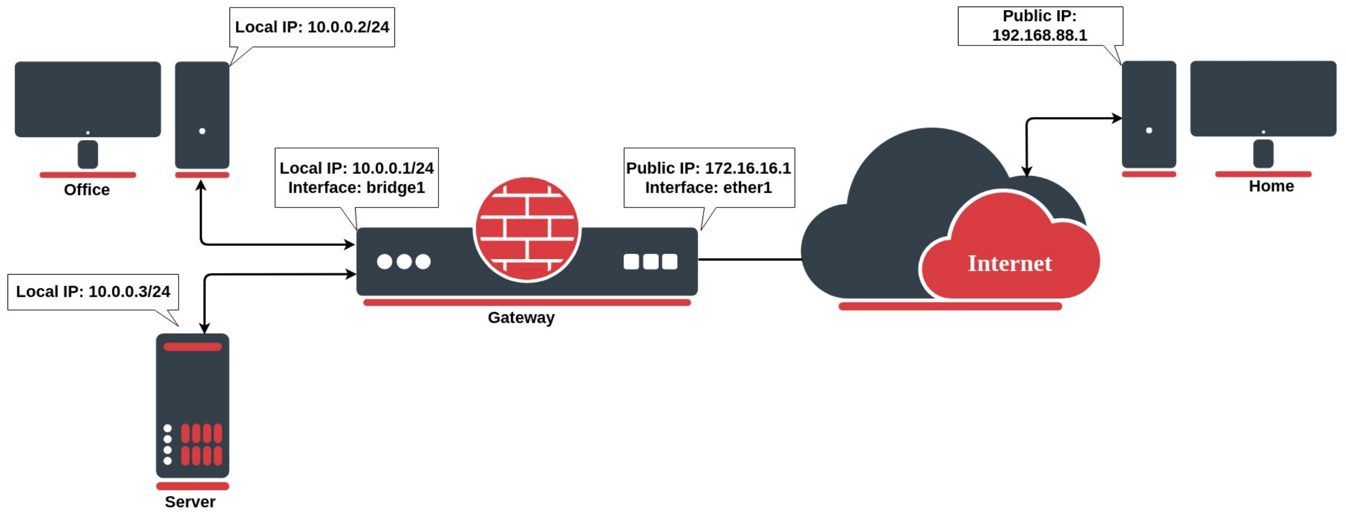
After execution, you should get a set of rules:

```
[admin@MikroTik] > ip firewall nat print
Flags: X - disabled, I - invalid; D - dynamic
 0    chain=srcnat action=jump jump-target=clients
      src-address=100.64.0.1-100.64.0.10

 1    chain=clients action=jump jump-target=client-1
      src-address=100.64.0.1-100.64.0.2

 2    chain=client-1 action=src-nat to-addresses=2.2.2.2 to-ports=5000-5199
      protocol=tcp src-address=100.64.0.1

 3    chain=client-1 action=src-nat to-addresses=2.2.2.2 to-ports=5000-5199
      protocol=udp src-address=100.64.0.1

 4    chain=client-1 action=src-nat to-addresses=2.2.2.2 to-ports=5200-5399
      protocol=tcp src-address=100.64.0.2

 5    chain=client-1 action=src-nat to-addresses=2.2.2.2 to-ports=5200-5399
      protocol=udp src-address=100.64.0.2

 6    chain=clients action=jump jump-target=client-2
      src-address=100.64.0.3-100.64.0.4

 7    chain=client-2 action=src-nat to-addresses=2.2.2.2 to-ports=5400-5599
      protocol=tcp src-address=100.64.0.3

 8    chain=client-2 action=src-nat to-addresses=2.2.2.2 to-ports=5400-5599
      protocol=udp src-address=100.64.0.3

 9    chain=client-2 action=src-nat to-addresses=2.2.2.2 to-ports=5600-5799
      protocol=tcp src-address=100.64.0.4

10    chain=client-2 action=src-nat to-addresses=2.2.2.2 to-ports=5600-5799
      protocol=udp src-address=100.64.0.4

11    chain=clients action=jump jump-target=client-3
      src-address=100.64.0.5-100.64.0.6

12    chain=client-3 action=src-nat to-addresses=2.2.2.2 to-ports=5800-5999
      protocol=tcp src-address=100.64.0.5

13    chain=client-3 action=src-nat to-addresses=2.2.2.2 to-ports=5800-5999
      protocol=udp src-address=100.64.0.5

14    chain=client-3 action=src-nat to-addresses=2.2.2.2 to-ports=6000-6199
      protocol=tcp src-address=100.64.0.6

15    chain=client-3 action=src-nat to-addresses=2.2.2.2 to-ports=6000-6199
```

```
      protocol=udp src-address=100.64.0.6

[...]
```

Hairpin NAT

Hairpin network address translation (*NAT Loopback*) is where the device on the LAN can access another machine on the LAN via the public IP address of the gateway router.

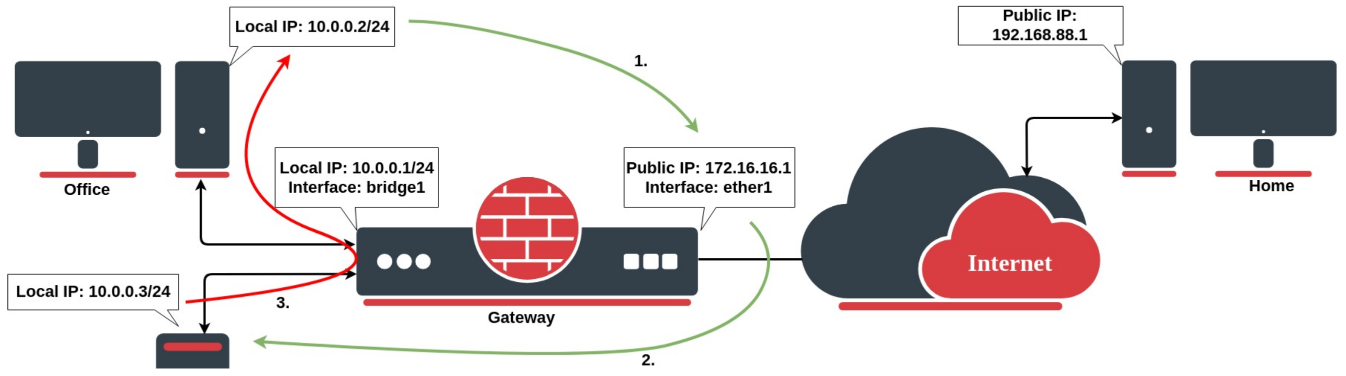


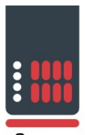
In the above example, the gateway router has the following `dst-nat` configuration rule:

```
/ip firewall nat add chain=dstnat action=dst-nat dst-address=172.16.16.1 dst-port=443 to-addresses=10.0.0.3 to-
ports=443 protocol=tcp
```

When a user from the PC at home establishes a connection to the web server, the router performs DST NAT as configured:

1. the client sends a packet with a source IP address of 192.168.88.1 to a destination IP address of 172.16.16.1 on port 443 to request some web resources;
2. the router destination NAT`s the packet to 10.0.0.3 and replaces the destination IP address in the packet accordingly. The source IP address stays the same: 192.168.88.1;
3. the server replies to the client's request and the reply packet has a source IP address of 10.0.0.3 and a destination IP address of 192.168.88.1.
4. the router determines that the packet is part of a previous connection and undoes the destination NAT, and puts the original destination IP address into the source IP address field. The destination IP address is 192.168.88.1, and the source IP address is 172.16.16.1;
5. The client receives the reply packet it expects, and the connection is established;





But, there will be a **problem**, when a client on the same network as the web server requests a connection to the web server's **public** IP address:

1. the client sends a packet with a source IP address of 10.0.0.2 to a destination IP address of 172.16.16.1 on port 443 to request some web resources;
2. the router destination NATs the packet to 10.0.0.3 and replaces the destination IP address in the packet accordingly. The source IP address stays the same: 10.0.0.2;
3. the server replies to the client's request. However, the source IP address of the request is on the same subnet as the web server. The web server does not send the reply back to the router but sends it back directly to 10.0.0.2 with a source IP address in the reply of 10.0.0.3;
4. The client receives the reply packet, but it discards it because it expects a packet back from 172.16.16.1, and not from 10.0.0.3;

To resolve this issue, we will configure a new *src-nat* rule (the hairpin NAT rule) as follows:

```
/ip firewall nat
add action=masquerade chain=srcnat dst-address=10.0.0.3 out-interface=LAN protocol=tcp src-address=10.0.0.0/24
```

After configuring the rule above:

1. the client sends a packet with a source IP address of 10.0.0.2 to a destination IP address of 172.16.16.1 on port 443 to request some web resources;
2. the router destination NATs the packet to 10.0.0.3 and replaces the destination IP address in the packet accordingly. It also source NATs the packet and replaces the source IP address in the packet with the IP address on its LAN interface. The destination IP address is 10.0.0.3, and the source IP address is 10.0.0.1;
3. the web server replies to the request and sends the reply with a source IP address of 10.0.0.3 back to the router's LAN interface IP address of 10.0.0.1;
4. the router determines that the packet is part of a previous connection and undoes both the source and destination NAT, and puts the original destination IP address of 10.0.0.3 into the source IP address field, and the original source IP address of 172.16.16.1 into the destination IP address field

Endpoint-Independent NAT

Endpoint-independent NAT creates mapping in the source NAT and uses the same mapping for all subsequent packets with the same source IP and port. This mapping is created with the following rule:

```
/ip firewall nat
add action=endpoint-independent-nat chain=srcnat out-interface=WAN protocol=udp
```

This mapping allows running source-independent filtering, which allows forwarding packets from any source from WAN to mapped internal IP and port. The following rule enables filtering:

```
/ip firewall nat
add action=endpoint-independent-nat chain=dstnat in-interface=WAN protocol=udp
```

Endpoint-independent NAT works only with UDP protocol.

Additionally, endpoint-independent-nat can take a few other parameters:

- **`randomize-port`** - randomize to which public port connections will be mapped.

More info https://www.ietf.org/rfc/rfc5128.txt section 2.2.3 and 2.2.5

NAT Helpers

Hosts behind a NAT-enabled router do not have true end-to-end connectivity. Therefore some Internet protocols might not work in scenarios with NAT. To overcome these limitations RouterOS includes a number of NAT helpers, that enable NAT traversal for various protocols.

Nat helpers can be managed from `/ip firewall service-ports` menu.

List of available nat helpers:

Helper	Description
FTP	FTP service helper
H323	H323 service helper
IRC	IRC service helper
PPTP	PPTP (GRE) tunneling helper
UDPLITE	UDP-Lite service helper
DCCP	DCCP service helper
SCTP	SCTP service helper
SIP	SIP helper. Additional options: • **sip-direct-media** allows redirecting the RTP media stream to go directly from the caller to the callee. The default value is *yes*. • **sip-timeout** allows adjusting TTL of SIP UDP connections. Default: 1 hour. In some setups, you have to reduce that.
TFTP	TFTP service helper
RSTP	RTSP service helper

If connection tracking is not enabled then firewall service ports will be shown as inactive

udplite, **dccp**, and **sctp** are built-in services of the connection tracking. Since these are not separately loaded modules, they cannot be disabled separately, they got disabled together with the connection tracking.

NAT Actions

Table lists NAT actions and their associated properties. Other actions are listed here.

Property	Description
action (*action name*; Default: **accept**)	• `dst-nat` - replaces the destination address and/or port of an IP packet with values specified by `to-addresses` a nd `to-ports` parameters • `masquerade` - replaces the source port of an IP packet with one specified by `to-ports` parameter and replace the source address of an IP packet to the IP determined by the routing facility. • `netmap` - creates a static 1:1 mapping of one set of IP addresses to another one. Often used to distribute public IP addresses to hosts on private networks • `redirect` - replaces the destination port of an IP packet with one specified by `to-ports` parameter and destination address to the address of the virtual or physical incoming interface (interface that recieved the packet).

	<ul><li>`same` - gives a particular client the same source/destination IP address from a supplied range for each connection. This is most frequently used for services that expect the same client address for multiple connections from the same client. **IPv4** only</li><li>`src-nat` - replaces the source address of an IP packet with values specified by `to-addresses` and `to-ports` parameters</li><li>`endpoint-independent-nat` - uses endpoint-independent mapping and filtering. Works only with UDP protocol. **IPv4** only.</li><li>`socksify` - routes traffic specified by firewall rules through SOCKS proxy server. Requires `socks5-server` and `socks5-port` parameters or socksify-service parameter.(relevant socksify information)</li></ul>
same-not-by-dst (*yes | no*; Default:)	Specifies whether to take into account or not the destination IP address when selecting a new source IP address. Applicable if `action=same`
to-addresses (*IP address [-IP address]*; Default: **0.0.0.0**)	Replace the original address with the specified one. Applicable if action is `dst-nat`, `netmap`, `same`, `src-nat`
to-ports (*integer[-integer]: 0..65535*; Default:)	Replace the original port with the specified one. Applicable if action is `dst-nat`, `redirect`, `masquerade`, `netmap`, `same`, `src-nat`