



# Network Address Translation

**Peter J. Welcher**

---

## What is Network Address Translation?

The Network Address Translation (NAT) functionality in the Cisco routers allows privately addressed networks to connect to public networks such as the Internet. When the privately addressed network (the "inside") sends a packet through the NAT router, the addresses are converted to legal (registered) IP addresses before being passed to the "outside".

Many sites have been using pass-through firewall gateways to provide this functionality. Well, it's now available in your Cisco router!

I've been encountering more and more sites that need to connect to others, where both parties are using private addressing. Network 10.0.0.0 is particularly popular (along the "more is better" line of thinking?). Two sites that have both used the same private network number have to either coordinate addressing or re-address to avoid conflicts. Now, with NAT, the same address ranges can be used multiple times.

There are some other sticky network problems that NAT solves.

## Uses of NAT

We've already mentioned that NAT cures duplicate address ranges without readdressing host computers.

The translation done by NAT can be either static or dynamic. Static translation is where we specify a lookup table, and one inside address is turned into one pre-specified outside address. Dynamic is where we tell the NAT router what inside addresses need to be translated, and what pool of addresses may be used for the outside addresses. There can be multiple pools of outside addresses. ICMP host unreachable messages are used when addresses run out.

With NAT, multiple internal hosts can also share a single outside IP address, which conserves address space. This is done by port multiplexing: changing the source port on the outbound packet so that replies can be directed back to the appropriate machine.

You can also do load distribution via NAT: have one external address (perhaps your Web server's name, [www.xyz.com](http://www.xyz.com), maps to this address). Then round-robin between different inside machines, so

that incoming *new* connections are distributed across several machines. (Since each connection may involve state information, a given connection has to remain on one machine.)

Finally, organizations that change service providers are now typically not allowed to take their address with them (because exceptions to CIDR addressing blocks have become a problem). NAT solves this by allowing re-addressing to occur at the gateway, allowing time to convert internal hosts to the new network number.

NAT also enhances security: internal network topology and addresses are hidden from the outside world.

The only thing NAT really can't do much about is sloppily-written applications, with hard-coded raw IP addresses.

Design issues:

Address translation is not practical for large numbers of internal hosts all talking at the same time to the outside world. NAT just won't work well at a large scale.

Performance may be a consideration. Currently, NAT causes process switching on NAT interfaces on a Cisco 7000. You can think of this as: the CPU has to look at every packet, to decide whether or not to translate it, and to alter the IP header, possibly the TCP header. One doubts that this will be easily cache-able.

## Configuring NAT

Let's look at how to translate addresses. For more specifics than follow, I'll have to refer you to the Cisco Command Configuration Guide, which is very lucid. The NAT section of the IP chapter is particularly well-written and clear. It has good illustrations and explains in detail what happens as a packet is forwarded through the NAT router.

Here's a minimal sample configuration for static address translation. We assume Ethernet 0 is "inside" and Serial 0 is "outside". Private network 10.0.0.0 is used inside, and 192.1.1.0 outside. We'll translate 10.1.2.3 to 192.1.1.2 (and vice versa). The words "inside source" emphasize that the inside source address is what's getting changed.

```
ip nat inside source static 10.1.2.3 192.1.1.2

interface ethernet 0

ip address 10.1.2.1 255.255.255.0

ip nat inside

interface serial 0

ip address 192.1.1.1 255.255.255.0

ip nat outside
```

You may add address mappings or inside or outside interfaces as necessary.

Let's look at dynamic (pooled) translation. Same network and addresses as before. We'll set up a pool

of addresses, translating sources in the range 10.1.2.0 through 10.1.2.255 to the range 192.1.1.10 through 20. The access list indicates what source addresses can be translated. The idea of the third line is that inside source addresses matching list 20 get translated to addresses from the pool named LegalPool. It pretty much says that, doesn't it!

```
ip nat pool LegalPool 192.1.1.10 192.1.1.20
access-list 20 permit 10.1.2.0 0.0.0.255
ip nat inside source list 20 pool LegalPool
interface ethernet 0
ip address 10.1.2.1 255.255.255.0
ip nat inside
interface serial 0
ip address 192.1.1.1 255.255.255.0
ip nat outside
```

And then there's address overloading (port multiplexing). To obtain a sample configuration for doing port multiplexing, add the word "overload" at the end of the "ip nat inside source" line. This gives the router permission to start sharing addresses.

```
ip nat inside source list 20 pool LegalPool overload
```

You can configure *outside* source address translation similarly, changing "inside source" to "outside source" in the above examples.

Let's look at how to do static outside address translation, supposing subnet 10.1.5.0 occurs both inside and outside (we're connecting to another company here). We only need to talk to the outside machine 10.1.5.3, and we'll readdress it as private address 192.168.1.1 on the inside (if we use 10.1.5.x, we have more complex routing issues to think about). This might call for something like the following.

```
ip nat outside source static 10.1.5.3 192.168.1.1
interface ethernet 0
ip address 10.1.2.1 255.255.255.0
ip nat inside
interface serial 0
ip address 10.1.3.1 255.255.255.0
ip nat outside
```

I'm ignoring routing issues here (as in, why does the inside net think network 192.168.1.0 is back through this NAT router?)

This sort of thing can also be done dynamically, using a pool of *inside* addresses, and an access list to match the *outside* source address, much like our earlier dynamic example.

You can even translate both ways, changing the source addresses as packets go through the router in *either* direction. Just combine the "inside source" and "outside source" configurations!

Finally, there's TCP Load Distribution. We define a pool of addresses containing the real hosts' addresses, ending with "type rotary". The access list now permits the IP address of the virtual host, i.e. what the outside world thinks is the host address. So the virtual host is 192.1.1.50, with the real hosts being 10.1.2.20 through 30.

```
ip nat pool ServerPool 10.1.2.20 10.1.2.30 type rotary
access-list 30 permit 192.1.1.50 0.0.0.0

ip nat inside destination list 30 pool ServerPool

interface ethernet 0

ip address 10.1.2.1 255.255.255.0

ip nat inside

interface serial 0

ip address 192.1.1.1 255.255.255.0

ip nat outside
```

There are also commands to tune address translation. The dynamic translations time out after non-use; the timeout period is configurable. (Without overloading, the timeout is 24 hours). The number is how many seconds before timeout.

```
ip nat translation timeout 3600
```

With overloading, there is finer control, for UDP, DNS, and TCP, also Finish and Reset packets:

```
ip nat translation udp-timeout 3600
ip nat translation dns-timeout 3600
ip nat translation tcp-timeout 3600
ip nat translation finrst-timeout 3600
```

(Please note I am *not* recommending 3600 seconds as a timeout, just using it to demonstrate the syntax of the command.)

## Commands To Manage NAT

To clear the dynamic translation table before timeout occurs:

```
clear ip nat translation *
```

You can also clear specific entries. To see what's there:

```
show ip nat translations
show ip nat translations verbose
```

There's also:

```
show ip nat statistics
```

There are also NAT debug commands. These include:

```
debug ip nat
```

```
debug ip nat 110
```

```
debug ip nat detailed
```

(Here the 110 is an access list number governing which NAT packets to display, as with other debug commands in recent IOS releases).

Don't use these debug commands in a production router, you'll have more output than you could possibly have wanted. They might be darn handy in a test scenario when you're trying to see how NAT works, or what it does!

---

Dr. Peter J. Welcher (CCIE #1773, CCSI #94014) is a Senior Consultant with Chesapeake NetCraftsmen. NetCraftsmen is a high-end consulting firm and Cisco Premier Partner dedicated to quality consulting and knowledge transfer. NetCraftsmen has nine CCIE's, with expertise including large network high-availability routing/switching and design, VoIP, QoS, MPLS, network management, security, IP multicast, and other areas. See <http://www.netcraftsmen.net> for more information about NetCraftsmen. Pete's links start at <http://www.netcraftsmen.net/welcher> . New articles will be posted under the Articles link. Questions, suggestions for articles, etc. can be sent to [pjw@netcraftsmen.net](mailto:pjw@netcraftsmen.net) .

---

Revised 11/9/98  
Copyright 1998, Peter J. Welcher