

Network Address Translation on a Stick

Document ID: 6505

Introduction

Prerequisites

- Requirements
- Components Used
- Conventions

Background Information

Example 1 Network Diagram and Configuration

- Network Diagram
- Requirements
- NAT Router Configuration

Example 1 show and debug Command Output

- Test One
- Test Two

Example 2 Network Diagram and Configuration

- Network Diagram
- Requirements
- NAT Router Configuration

Example 2 show and debug Command Output

- Test One

Summary

Related Information

Introduction

What do we mean by Network Address Translation (NAT) on a stick? The term "on a stick" usually implies the use of a single physical interface of a router for a task. Just as we can use subinterfaces of the same physical interface to perform Inter-Switch Link (ISL) trunking, we can use a single physical interface on a router in order to accomplish NAT.

Note: The router must process switch every packet due to the loopback interface. This degrades the performance of the router.

Prerequisites

Requirements

There are no specific requirements for this document.

Components Used

This feature requires you to use a version of Cisco IOS[®] Software that supports NAT. Use the Cisco Feature Navigator II (registered customers only) to determine which IOS versions you can use with this feature.

Conventions

For more information on document conventions, refer to Cisco Technical Tips Conventions.

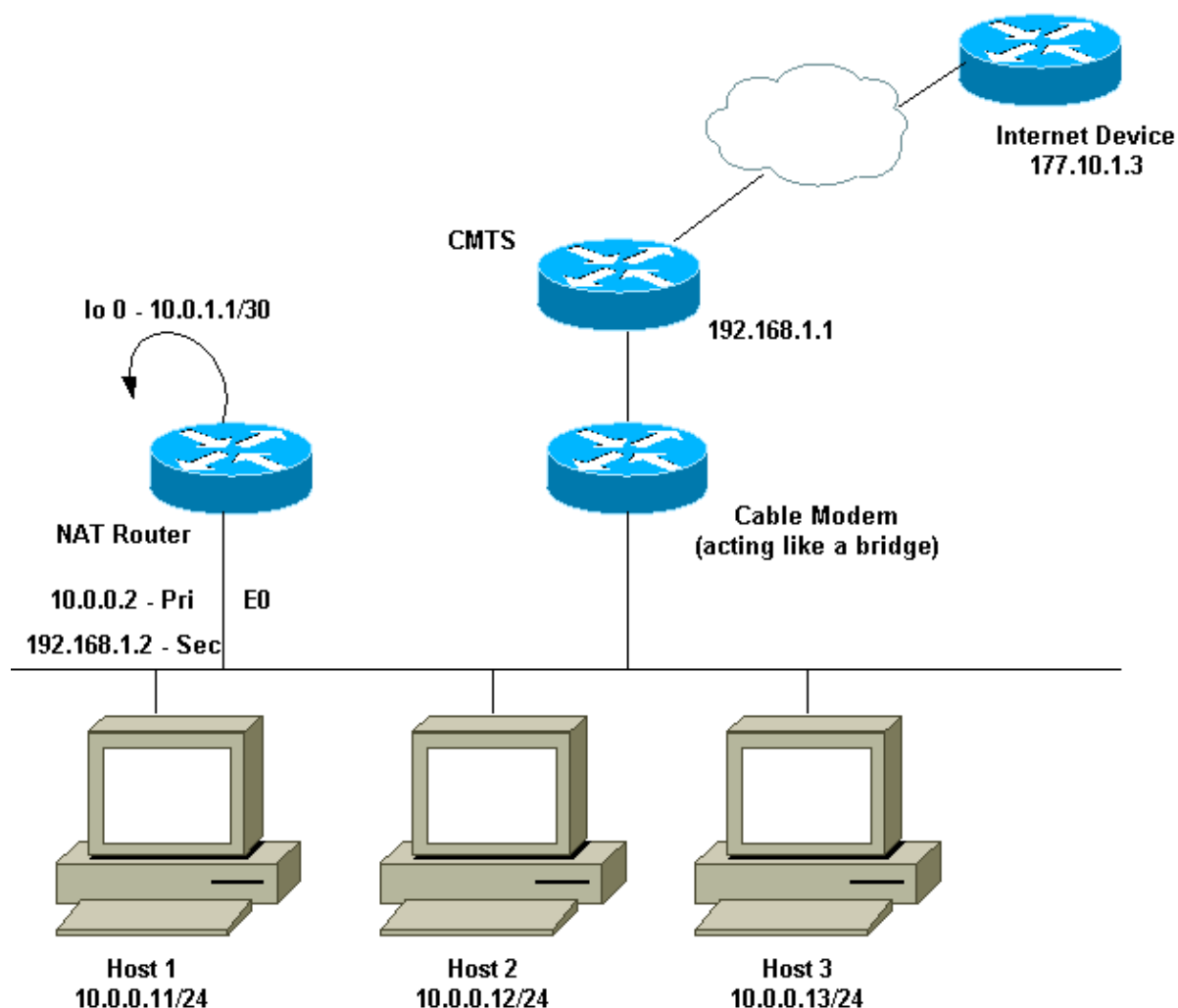
Background Information

In order for NAT to take place, a packet must be switched from a NAT "inside" defined interface to a NAT "outside" defined interface or vice-versa. This requirement for NAT has not changed, but this document demonstrates how you can use a virtual interface, otherwise known as a loopback interface, and policy-based routing to make NAT work on a router with a single physical interface.

The need for NAT on a stick is rare. In fact, the examples in this document may be the only situations in which this configuration is needed. Though other occasions arise where users employ policy routing in conjunction with NAT, we do not consider this to be NAT on a stick because these instances still use more than one physical interface.

Example 1 Network Diagram and Configuration

Network Diagram



The above network diagram is very common in a cable modem set up. The Cable Modem Termination System (CMTS) is a router and the Cable Modem (CM) is a device that acts like a bridge. The problem we face is that our Internet Service Provider (ISP) has not given us enough valid addresses for the number of hosts that need to reach the Internet. The ISP gave us the address 192.168.1.2, which was to be used for a device. Upon further request, we received three more; 192.168.2.1 to 192.168.2.3 into which NAT translates the hosts in the 10.0.0.0/24 range.

Requirements

Our requirements are:

- All hosts on the network must be able to reach the Internet.
- Host 2 must be able to be reached from the Internet with the IP address of 192.168.2.1.
- Because we can have more hosts than legal addresses, we use the 10.0.0.0/24 subnet for our internal addressing.

For the purposes of this document, we only show the configuration of the NAT router. However, we do mention some important configuration notes with respect to the hosts.

NAT Router Configuration

```

NAT Router Configuration

interface Loopback0
 ip address 10.0.1.1 255.255.255.252
 ip nat outside

!--- Creates a virtual interface called Loopback 0 and assigns an
!--- IP address of 10.0.1.1 to it. Defines interface Loopback 0 as
!--- NAT outside.

!
!
interface Ethernet0
 ip address 192.168.1.2 255.255.255.0 secondary
 ip address 10.0.0.2 255.255.255.0
 ip Nat inside

!--- Assigns a primary IP address of 10.0.0.2 and a secondary IP
!--- address of 192.168.1.2 to Ethernet 0. Defines interface Ethernet 0
!--- as NAT inside. The 192.168.1.2 address will be used to communicate
!--- through the CM to the CMTS and the Internet. The 10.0.0.2 address
!--- will be used to communicate with the local hosts.

ip policy route-map Nat-loop

!--- Assigns route-map "Nat-loop" to Ethernet 0 for policy routing.

!
ip Nat pool external 192.168.2.2 192.168.2.3 prefix-length 29
ip Nat inside source list 10 pool external overload
ip Nat inside source static 10.0.0.12 192.168.2.1

!--- NAT is defined: packets that match access-list 10 will be
!--- translated to an address from the pool called "external".
!--- A static NAT translation is defined for 10.0.0.12 to be
!--- translated to 192.168.2.1 (this is for host 2 which needs
!--- to be accessed from the Internet).

ip classless
!
!
ip route 0.0.0.0 0.0.0.0 192.168.1.1
ip route 192.168.2.0 255.255.255.0 Ethernet0

!--- Static default route set as 192.168.1.1, also a static
!--- route for network 192.168.2.0/24 directly attached to
```

```

!--- Ethernet 0
!
!
access-list 10 permit 10.0.0.0 0.0.0.255

!--- Access-list 10 defined for use by NAT statement above.

access-list 102 permit ip any 192.168.2.0 0.0.0.255
access-list 102 permit ip 10.0.0.0 0.0.0.255 any

!--- Access-list 102 defined and used by route-map "Nat-loop"
!--- which is used for policy routing.

!
Access-list 177 permit icmp any any

!--- Access-list 177 used for debug.

!
route-map Nat-loop permit 10
  match ip address 102
  set ip next-hop 10.0.1.2

!--- Creates route-map "Nat-loop" used for policy routing.
!--- Route map states that any packets that match access-list 102 will
!--- have the next hop set to 10.0.1.2 and be routed "out" the
!--- loopback interface. All other packets will be routed normally.
!--- We use 10.0.1.2 because this next-hop is seen as located
!--- on the loopback interface which would result in policy routing to
!--- loopback0. Alternatively, we could have used "set interface
!--- loopback0" which would have done the same thing.

!
end
NAT-router#

```

Note: All the hosts have their default gateway set to 10.0.0.2, which is the NAT router. The ISP as well as the CMTS must have a route to 192.168.2.0/29 which points to the NAT router for the return traffic to work, because the traffic from the inside hosts appears as arriving from this subnet. In this example, the CMTS would route traffic for 192.168.2.0/29 to 192.168.1.2 which is the secondary IP address configured on the NAT router.

Example 1 show and debug Command Output

This section provides information you can use to confirm your configuration works properly.

In order to illustrate that the above configuration works, we have run a few **ping** tests while the **debug** output on the NAT router is monitored. You can see that the **ping** commands are successful and the **debug** output shows exactly what is happening.

Note: Before you use **debug** commands, refer to Important Information on Debug Commands.

Test One

For our first test, we **ping** from a device in our lab-defined Internet to Host 2. Remember that one of the requirements was that devices in the Internet must be able to communicate with Host 2 with the IP address of 192.168.2.1. The following is the **debug** output as seen on the NAT router. The **debug** commands that were running on the NAT router were **debug ip packet 177 detail** which uses the defined **access-list 177**, **debug**

ip Nat, and **debug ip policy** which shows us the policy-routed packets.

This is the output of the **show ip Nat translation** command executed on the NAT router:

```
NAT-router# show ip Nat translation
Pro Inside global      Inside local      Outside local      Outside global
--- 192.168.2.1        10.0.0.12        ---                ---
NAT-router#
```

From a device on the internet, in this case a router, we **ping** 192.168.2.1 which is successful as shown here:

```
Internet-device# ping 192.168.2.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.2.1, timeout is 2 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 92/92/92 ms
Internet-device#
```

To see what happens in the NAT router, refer to this **debug** output and comments:

```
IP: s=177.10.1.3 (Ethernet0), d=192.168.2.1, len 100, policy match
    ICMP type=8, code=0
IP: route map Nat-loop, item 10, permit
IP: s=177.10.1.3 (Ethernet0), d=192.168.2.1 (Loopback0), Len 100, policy routed
    ICMP type=8, code=0

!--- The above debug output shows the packet with source 177.10.1.3 destined
!--- to 192.168.2.1. The packet matches the statements in the "Nat-loop"
!--- policy route map and is permitted and policy-routed. The Internet
!--- Control Message Protocol (ICMP) type 8, code 0 indicates that this
!--- packet is an ICMP echo request packet.

IP: Ethernet0 to Loopback0 10.0.1.2
IP: s=177.10.1.3 (Ethernet0), d=192.168.2.1 (Loopback0), g=10.0.1.2, Len 100,
forward
    ICMP type=8, code=0

!--- The packet now is routed to the new next hop address of 10.0.1.2
!--- as shown above.

IP: NAT enab = 1 trans = 0 flags = 0
NAT: s=177.10.1.3, d=192.168.2.1->10.0.0.12 [52]
IP: s=177.10.1.3 (Loopback0), d=10.0.0.12 (Ethernet0), g=10.0.0.12, Len 100,
forward
    ICMP type=8, code=0
IP: NAT enab = 1 trans = 0 flags = 0

!--- Now that the routing decision has been made, NAT takes place. We can
!--- see above that the address 192.168.2.1 is translated to 10.0.0.12 and
!--- this packet is forwarded out Ethernet 0 to the local host.
!--- Note: When a packet is going from inside to outside, it is routed and
!--- then translated (NAT). In the opposite direction (outside to inside),
!--- NAT takes place first.

IP: s=10.0.0.12 (Ethernet0), d=177.10.1.3, Len 100, policy match
    ICMP type=0, code=0
IP: route map Nat-loop, item 10, permit
IP: s=10.0.0.12 (Ethernet0), d=177.10.1.3 (Loopback0), Len 100, policy routed
    ICMP type=0, code=0
IP: Ethernet0 to Loopback0 10.0.1.2
```

```

!--- Host 2 now sends an ICMP echo response, seen as ICMP type 0, code 0.
!--- This packet also matches the policy routing statements and is
!--- permitted for policy routing.

NAT: s=10.0.0.12->192.168.2.1, d=177.10.1.3 [52]
IP: s=192.168.2.1 (Ethernet0), d=177.10.1.3 (Loopback0), g=10.0.1.2, Len 100,
forward
    ICMP type=0, code=0
IP: s=192.168.2.1 (Loopback0), d=177.10.1.3 (Ethernet0), g=192.168.1.1, Len 100,
forward
    ICMP type=0, code=0
IP: NAT enab = 1 trans = 0 flags = 0

!--- The above output shows the Host 2 IP address is translated to
!--- 192.168.2.1 and the packet that results packet is sent out loopback 0,
!--- because of the policy based routing, and finally forwarded
!--- out Ethernet 0 to the Internet device.

!--- The remainder of the debug output shown is a repeat of the previous
!--- for each of the additional four ICMP packet exchanges (by default,
!--- five ICMP packets are sent when pinging from Cisco routers). We have
!--- omitted most of the output since it is redundant.

IP: s=177.10.1.3 (Ethernet0), d=192.168.2.1, Len 100, policy match
    ICMP type=8, code=0
IP: route map Nat-loop, item 10, permit
IP: s=177.10.1.3 (Ethernet0), d=192.168.2.1 (Loopback0), Len 100, policy routed
    ICMP type=8, code=0
IP: Ethernet0 to Loopback0 10.0.1.2
IP: s=177.10.1.3 (Ethernet0), d=192.168.2.1 (Loopback0), g=10.0.1.2, Len 100,
forward
    ICMP type=8, code=0
IP: NAT enab = 1 trans = 0 flags = 0
NAT: s=177.10.1.3, d=192.168.2.1->10.0.0.12 [53]
IP: s=177.10.1.3 (Loopback0), d=10.0.0.12 (Ethernet0), g=10.0.0.12, Len 100,
forward
    ICMP type=8, code=0
IP: NAT enab = 1 trans = 0 flags = 0
IP: s=10.0.0.12 (Ethernet0), d=177.10.1.3, Len 100, policy match
    ICMP type=0, code=0
IP: route map Nat-loop, item 10, permit
IP: s=10.0.0.12 (Ethernet0), d=177.10.1.3 (Loopback0), Len 100, policy routed
    ICMP type=0, code=0
IP: Ethernet0 to Loopback0 10.0.1.2
NAT: s=10.0.0.12->192.168.2.1, d=177.10.1.3 [53]
IP: s=192.168.2.1 (Ethernet0), d=177.10.1.3 (Loopback0), g=10.0.1.2, Len 100,
forward
    ICMP type=0, code=0
IP: s=192.168.2.1 (Loopback0), d=177.10.1.3 (Ethernet0), g=192.168.1.1, Len 100,
forward
    ICMP type=0, code=0
IP: NAT enab = 1 trans = 0 flags = 0

```

Test Two

Another of our requirements is to allow the hosts the ability to communicate with the Internet. For this test, we **ping** the Internet device from Host 1. The resulting **show** and **debug** commands are below.

Initially the NAT translation table in the NAT router is as follows:

```
NAT-router# show ip Nat translation
Pro Inside global      Inside local      Outside local      Outside global
--- 192.168.2.1        10.0.0.12        ---                ---
NAT-router#
```

Once we issue the **ping** from Host 1, we see:

```
Host-1# ping 177.10.1.3

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 177.10.1.3, timeout is 2 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 92/92/96 ms
Host-1#
```

We see above that the **ping** was successful. The NAT table in the NAT router now looks like:

```
NAT-router# show ip Nat translation
Pro Inside global      Inside local      Outside local      Outside global
icmp 192.168.2.2:434   10.0.0.11:434    177.10.1.3:434    177.10.1.3:434
icmp 192.168.2.2:435   10.0.0.11:435    177.10.1.3:435    177.10.1.3:435
icmp 192.168.2.2:436   10.0.0.11:436    177.10.1.3:436    177.10.1.3:436
icmp 192.168.2.2:437   10.0.0.11:437    177.10.1.3:437    177.10.1.3:437
icmp 192.168.2.2:438   10.0.0.11:438    177.10.1.3:438    177.10.1.3:438
--- 192.168.2.1        10.0.0.12        ---                ---
NAT-router#
```

The NAT translation table above now shows additional translations which are a result of the dynamic NAT configuration (as opposed the static NAT configuration).

The **debug** output below shows what occurs on the NAT router.

```
IP: NAT enab = 1 trans = 0 flags = 0
IP: s=10.0.0.11 (Ethernet0), d=177.10.1.3, Len 100, policy match
    ICMP type=8, code=0
IP: route map Nat-loop, item 10, permit
IP: s=10.0.0.11 (Ethernet0), d=177.10.1.3 (Loopback0), Len 100, policy routed
    ICMP type=8, code=0
IP: Ethernet0 to Loopback0 10.0.1.2

!--- The above output shows the ICMP echo request packet originated by
!--- Host 1 which is policy-routed out the loopback interface.

NAT: s=10.0.0.11->192.168.2.2, d=177.10.1.3 [8]
IP: s=192.168.2.2 (Ethernet0), d=177.10.1.3 (Loopback0), g=10.0.1.2, Len 100,
forward
    ICMP type=8, code=0
IP: s=192.168.2.2 (Loopback0), d=177.10.1.3 (Ethernet0), g=192.168.1.1, Len 100,
forward
    ICMP type=8, code=0
IP: NAT enab = 1 trans = 0 flags = 0

!--- After the routing decision has been made by the policy routing,
!--- translation takes place, which translates the Host 1 IP address of 10.0.0.11
!--- to an address from the "external" pool 192.168.2.2 as shown above.
!--- The packet is then forwarded out loopback 0 and finally out Ethernet 0
!--- to the Internet device.

IP: s=177.10.1.3 (Ethernet0), d=192.168.2.2, Len 100, policy match
    ICMP type=0, code=0
IP: route map Nat-loop, item 10, permit
IP: s=177.10.1.3 (Ethernet0), d=192.168.2.2 (Loopback0), Len 100, policy routed
```

```

    ICMP type=0, code=0
IP: Ethernet0 to Loopback0 10.0.1.2
IP: s=177.10.1.3 (Ethernet0), d=192.168.2.2 (Loopback0), g=10.0.1.2, Len 100,
forward
    ICMP type=0, code=0

!--- The Internet device sends an ICMP echo response which matches our
!--- policy, is policy-routed, and forward out the Loopback 0 interface.

IP: NAT enab = 1 trans = 0 flags = 0
NAT: s=177.10.1.3, d=192.168.2.2->10.0.0.11 [8]
IP: s=177.10.1.3 (Loopback0), d=10.0.0.11 (Ethernet0), g=10.0.0.11, Len 100,
forward
    ICMP type=0, code=0

!--- The packet is looped back into the loopback interface at which point
!--- the destination portion of the address is translated from 192.168.2.2
!--- to 10.0.0.11 and forwarded out the Ethernet 0 interface to the local host.

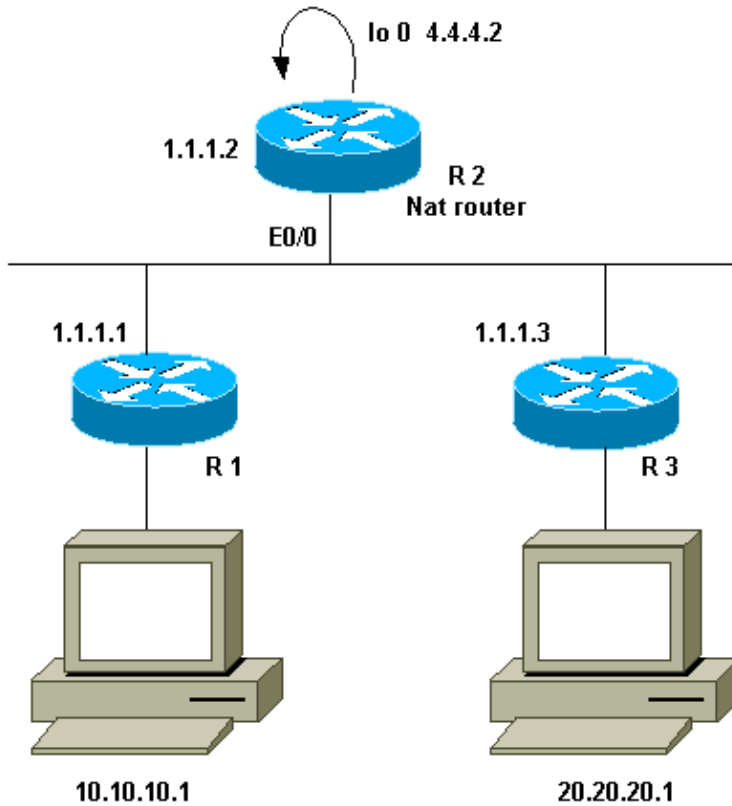
!--- The ICMP exchange is repeated for the rest of the ICMP packets, some of
!--- which are shown below.

IP: NAT enab = 1 trans = 0 flags = 0
IP: s=10.0.0.11 (Ethernet0), d=177.10.1.3, Len 100, policy match
    ICMP type=8, code=0
IP: route map Nat-loop, item 10, permit
IP: s=10.0.0.11 (Ethernet0), d=177.10.1.3 (Loopback0), Len 100, policy routed
    ICMP type=8, code=0
IP: Ethernet0 to Loopback0 10.0.1.2
NAT: s=10.0.0.11->192.168.2.2, d=177.10.1.3 [9]
IP: s=192.168.2.2 (Ethernet0), d=177.10.1.3 (Loopback0), g=10.0.1.2, Len 100,
forward
    ICMP type=8, code=0
IP: s=192.168.2.2 (Loopback0), d=177.10.1.3 (Ethernet0), g=192.168.1.1, Len 100,
forward
    ICMP type=8, code=0
IP: NAT enab = 1 trans = 0 flags = 0
IP: s=177.10.1.3 (Ethernet0), d=192.168.2.2, Len 100, policy match
    ICMP type=0, code=0
IP: route map Nat-loop, item 10, permit
IP: s=177.10.1.3 (Ethernet0), d=192.168.2.2 (Loopback0), Len 100, policy routed
    ICMP type=0, code=0
IP: Ethernet0 to Loopback0 10.0.1.2
IP: s=177.10.1.3 (Ethernet0), d=192.168.2.2 (Loopback0), g=10.0.1.2, Len 100,
forward
    ICMP type=0, code=0
IP: NAT enab = 1 trans = 0 flags = 0
NAT: s=177.10.1.3, d=192.168.2.2->10.0.0.11 [9]
IP: s=177.10.1.3 (Loopback0), d=10.0.0.11 (Ethernet0), g=10.0.0.11, Len 100,
forward
    ICMP type=0, code=0

```

Example 2 Network Diagram and Configuration

Network Diagram



Requirements

We want certain devices behind the two sites (R1 and R3) to communicate. The two sites use non-registered IP addresses, so we must translate the addresses when they communicate with each other. In our case, host 10.10.10.1 is translated to 200.200.200.1 and host 20.20.20.1 will be translated to 100.100.100.1. Therefore, we need translation to occur in both directions. For accounting purposes, traffic between these two sites must pass through R2. To summarize, our requirements are:

- Host 10.10.10.1, behind R1, needs to communicate with Host 20.20.20.1 behind R3 with the use of their global addresses.
- Traffic between these hosts must be sent through R2.
- For our case, we need Static NAT translations as shown in the configuration below.

NAT Router Configuration

NAT Router Configuration
<pre> interface Loopback0 ip address 4.4.4.2 255.255.255.0 ip Nat inside !--- Creates a virtual interface called "loopback 0" and assigns IP address !--- 4.4.4.2 to it. Also defines for it a NAT inside interface. ! Interface Ethernet0/0 ip address 1.1.1.2 255.255.255.0 no ip redirects ip Nat outside ip policy route-map Nat </pre>

```

!--- Assigns IP address 1.1.1.1/24 to e0/0. Disables redirects so that packets
!--- which arrive from R1 destined toward R3 are not redirected to R3 and
!--- visa-versa. Defines the interface as NAT outside interface. Assigns
!--- route-map "Nat" used for policy-based routing.

!
ip Nat inside source static 10.10.10.1 200.200.200.1

!--- Creates a static translation so packets received on the inside interface
!--- with a source address of 10.10.10.1 will have their source address
!--- translated to 200.200.200.1. Note: This implies that the packets received
!--- on the outside interface with a destination address of 200.200.200.1
!--- will have the destination translated to 10.10.10.1.

ip Nat outside source static 20.20.20.1 100.100.100.1

!--- Creates a static translation so packets received on the outside interface
!--- with a source address of 20.20.20.1 will have their source address
!--- translated to 100.100.100.1. Note: This implies that packets received on
!--- the inside interface with a destination address of 100.100.100.1 will
!--- have the destination translated to 20.20.20.1.

ip route 10.10.10.0 255.255.255.0 1.1.1.1
ip route 20.20.20.0 255.255.255.0 1.1.1.3
ip route 100.100.100.0 255.255.255.0 1.1.1.3
!
access-list 101 permit ip host 10.10.10.1 host 100.100.100.1
route-map Nat permit 10
 match ip address 101
 set ip next-hop 4.4.4.2

```

Example 2 show and debug Command Output

Note: Certain show commands are supported by the Output Interpreter tool, which allows you to view an analysis of show command output. Before you use **debug** commands, refer to Important Information on Debug Commands.

Test One

As shown in the configuration above, we have two static NAT translations which can be seen on R2 with the command **show ip Nat translation**.

This is the output of the **show ip Nat translation** command executed on the NAT router:

```

NAT-router# show ip Nat translation
Pro Inside global      Inside local          Outside local         Outside global
--- ---
--- 200.200.200.1      10.10.10.1           ---
R2#

```

For this test, we sourced a **ping** from a device (10.10.10.1) behind R1 destined for the global address of a device (100.100.100.1) behind R3. Running **debug ip Nat** and **debug ip packet** on R2 resulted in this output:

```

IP: NAT enab = 1 trans = 0 flags = 0
IP: s=10.10.10.1 (Ethernet0/0), d=100.100.100.1, Len 100, policy match
   ICMP type=8, code=0
IP: route map Nat, item 10, permit

```

```

IP: s=10.10.10.1 (Ethernet0/0), d=100.100.100.1 (Loopback0), Len 100, policy
routed
    ICMP type=8, code=0
IP: Ethernet0/0 to Loopback0 4.4.4.2

!--- The above output shows the packet source from 10.10.10.1 destined
!--- for 100.100.100.1 arrives on E0/0, which is defined as a NAT
!--- outside interface. There is not any NAT that needs to take place at
!--- this point, however the router also has policy routing enabled for
!--- E0/0. The output shows that the packet matches the policy that is
!--- defined in the policy routing statements.

IP: s=10.10.10.1 (Ethernet0/0), d=100.100.100.1 (Loopback0), g=4.4.4.2, Len 100,
forward
    ICMP type=8, code=0
IP: NAT enab = 1 trans = 0 flags = 0

!--- The above now shows the packet is policy-routed out the loopback0
!--- interface. Remember the loopback is defined as a NAT inside interface.

NAT: s=10.10.10.1->200.200.200.1, d=100.100.100.1 [26]
NAT: s=200.200.200.1, d=100.100.100.1->20.20.20.1 [26]

!--- For the above output, the packet is now arriving on the loopback0
!--- interface. Since this is a NAT inside interface, it is important to
!--- note that before the translation shown above takes place, the router
!--- will look for a route in the routing table to the destination, which
!--- before the translation is still 100.100.100.1. Once this route look up
!--- is complete, the router will continue with translation, as shown above.
!--- The route lookup is not shown in the debug output.

IP: s=200.200.200.1 (Loopback0), d=20.20.20.1 (Ethernet0/0), g=1.1.1.3, Len 100,
forward
    ICMP type=8, code=0
IP: NAT enab = 1 trans = 0 flags = 0

!--- The above output shows the resulting translated packet that results is
!--- forwarded out E0/0.

```

This is the output as a result of the response packet sourced from the device behind router 3 destined for the device behind router 1:

```

NAT: s=20.20.20.1->100.100.100.1, d=200.200.200.1 [26]
NAT: s=100.100.100.1, d=200.200.200.1->10.10.10.1 [26]

!--- The return packet arrives into the e0/0 interface which is a NAT
!--- outside interface. In this direction (outside to inside), translation
!--- occurs before routing. The above output shows the translation takes place.

IP: s=100.100.100.1 (Ethernet0/0), d=10.10.10.1 (Ethernet0/0), Len 100, policy
rejected -- normal forwarding
    ICMP type=0, code=0
IP: s=100.100.100.1 (Ethernet0/0), d=10.10.10.1 (Ethernet0/0), g=1.1.1.1,
Len 100, forward
    ICMP type=0, code=0

!--- The E0/0 interface still has policy routing enabled, so the packet is
!--- check against the policy, as shown above. The packet does not match the
!--- policy and is forwarded normally.

```

Summary

This document has demonstrated how the use of NAT and policy-based routing can be used to create a "NAT on a stick" scenario. It is important to keep in mind that this configuration can reduce the performance on the router running NAT because the packets may be process-switched through the router.

Related Information

- [NAT Support Page](#)
 - [Technical Support – Cisco Systems](#)
-

[Contacts & Feedback](#) | [Help](#) | [Site Map](#)

© 2008 – 2009 Cisco Systems, Inc. All rights reserved. [Terms & Conditions](#) | [Privacy Statement](#) | [Cookie Policy](#) | [Trademarks of Cisco Systems, Inc.](#)

Updated: Dec 15, 2008

Document ID: 6505
