Nat in 8.x and 9.x

Junling Yao

June 11 2014

Agenda

- ASA 8.x nat types
- ASA 8.x nat match rules
- The concept of nat control in 8.2x
- ASA 9.x nat types
- ASA 9.x nat match rules
- Migration to 8.3 and later
- Data forwarding in asa
- Common tools used for troubleshooting
- Cases for xlate issue
- Performing Zero Downtime Upgrades for Failover Pairs

NAT In 8.2x

NAT concept

Address translation substitutes the real address in a packet with a mapped address that is routable on the destination network. NAT is composed of two steps: the process by which a real address is translated into a mapped address and the process to undo translation for returning traffic.

The ASA translates an address when a NAT rule matches the traffic. If no NAT rule matches, processing for the packet continues. The exception is when you enable NAT control. NAT control requires that packets traversing from a higher security interface (inside) to a lower security interface (outside) match a NAT rule, or processing for the packet stops.

NAT Types

- Dynamic NAT—Dynamic NAT translates a group of real addresses to a pool of mapped addresses that are routable on the destination network
- •PAT—PAT translates multiple real address to a single mapped IP address
- •Static NAT—Static NAT creates a fixed translation of real addresses to mapped addresses. With dynamic NAT and PAT, each host uses a different address or port for each subsequent translation.
- •Static PAT—Static PAT is the same as static NAT, except that it enables you to specify the protocol and port for the real and mapped addresses.

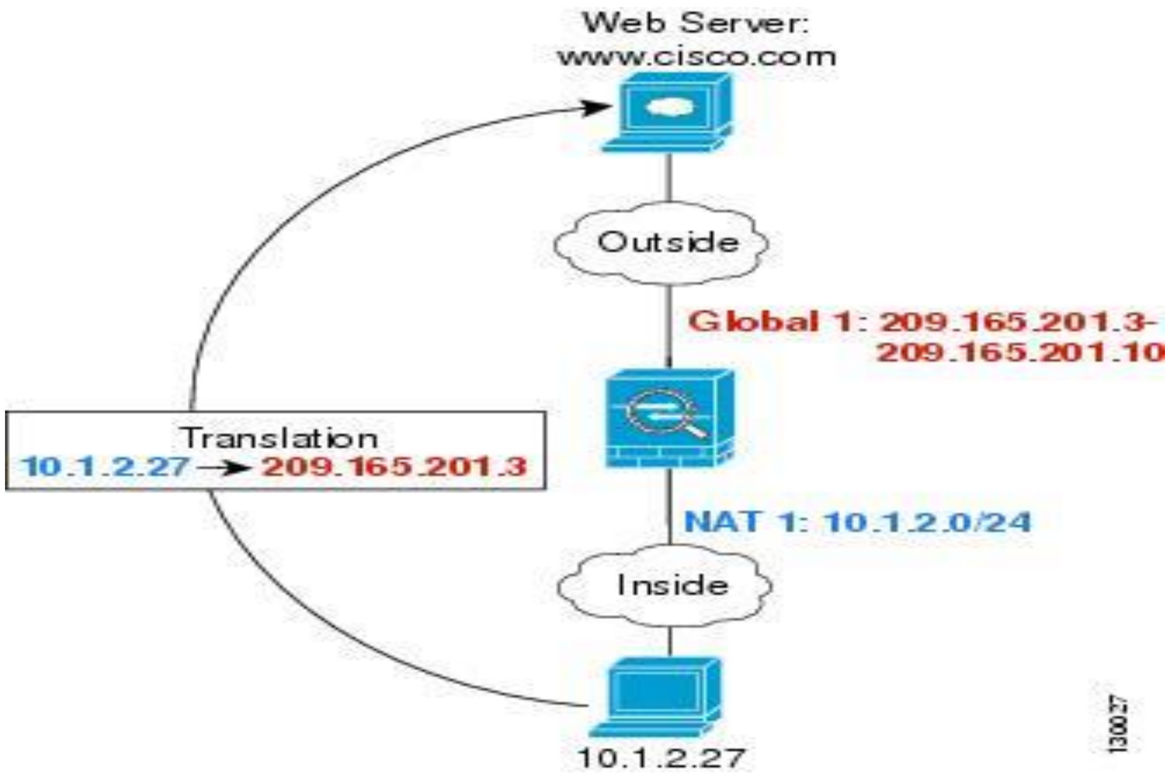
Dynamic NAT

- Dynamic NAT translates a group of real addresses to a pool of mapped addresses that are routable on the destination network. The mapped pool may include fewer addresses than the real group. When a host you want to translate accesses the destination network, the ASA assigns the host an IP address from the mapped pool. The translation is added only when the real host initiates the connection.

Dynamic NAT disadvantages

- •If the mapped pool has fewer addresses than the real group, you could run out of addresses if the amount of traffic is more than expected.
- Use PAT if this event occurs often because PAT provides over 64,000 translations using ports of a single address.
- •You have to use a large number of routable addresses in the mapped pool; if the destination network requires registered addresses, such as the Internet, you might encounter a shortage of usable addresses.

Dynamic NAT sample



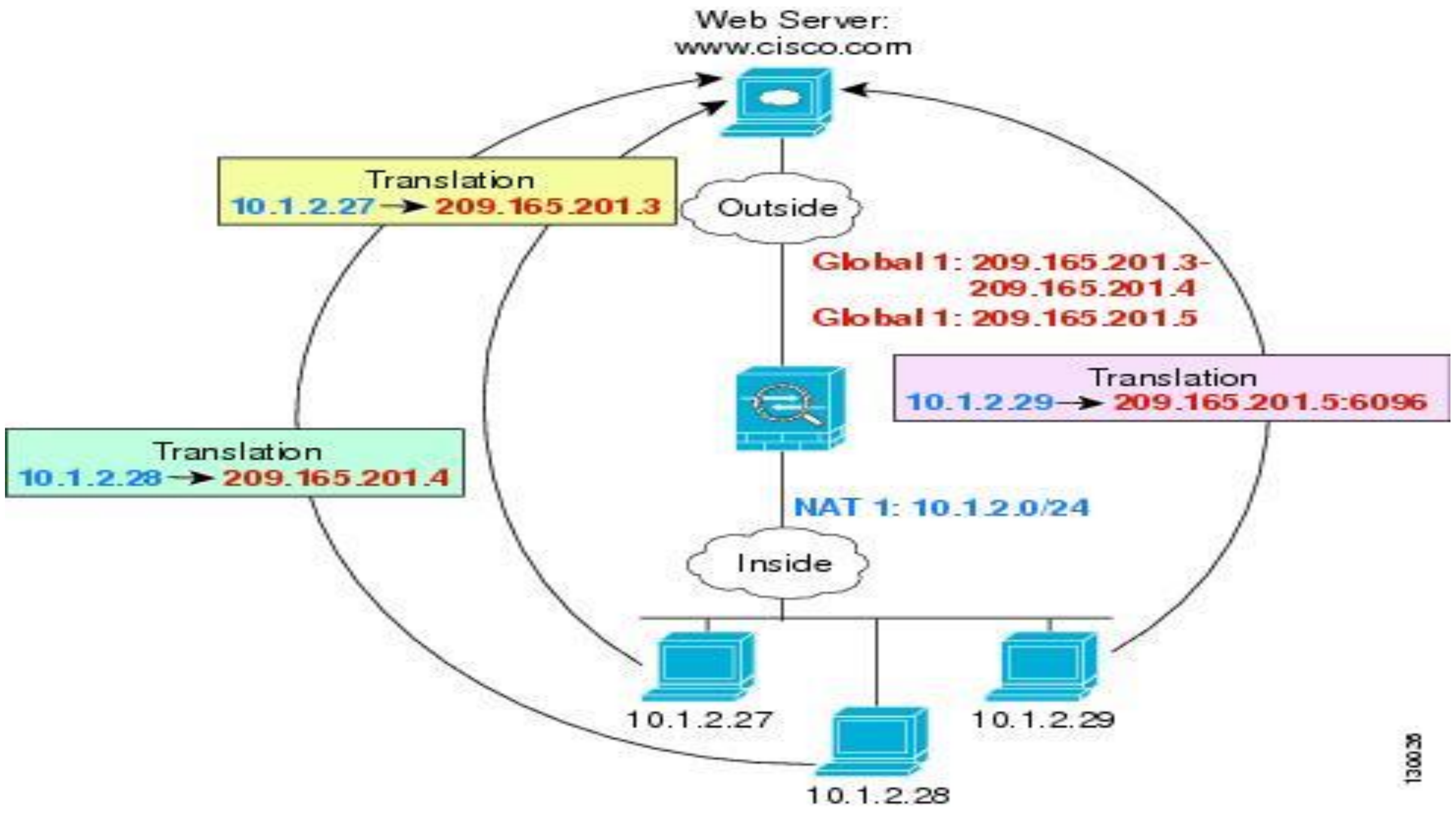
```
hostname(config)# nat (inside) 1 10.1.2.0 255.255.255.0
hostname(config)# global (outside) 1 209.165.201.3-209.165.201.10
```

Dynamic PAT

- PAT translates multiple real addresses to a single mapped IP address by translating the real address and source port to the mapped address and a unique port. If available, the real source port number is used for the mapped port. However, if the real port is *not* available, by default the mapped ports are chosen from the same range of ports as the real port number: 0 to 511, 512 to 1023, and 1024 to 65535. Therefore, ports below 1024 have only a small PAT pool that can be used.
- Each connection requires a separate translation because the source port differs for each connection. For example, 10.1.1.1:1025 requires a separate translation from 10.1.1.1:1026.

- After the connection expires, the port translation also expires after 30 seconds of inactivity. The timeout is not configurable. Users on the destination network cannot reliably initiate a connection to a host that uses PAT (even if the connection is allowed by an access list). Not only can you not predict the real or mapped port number of the host, but the ASA does not create a translation at all unless the translated host is the initiator.
- PAT lets you use a single mapped address, thus conserving routable addresses. You can even use the ASA interface IP address as the PAT address.

NAT and PAT Together

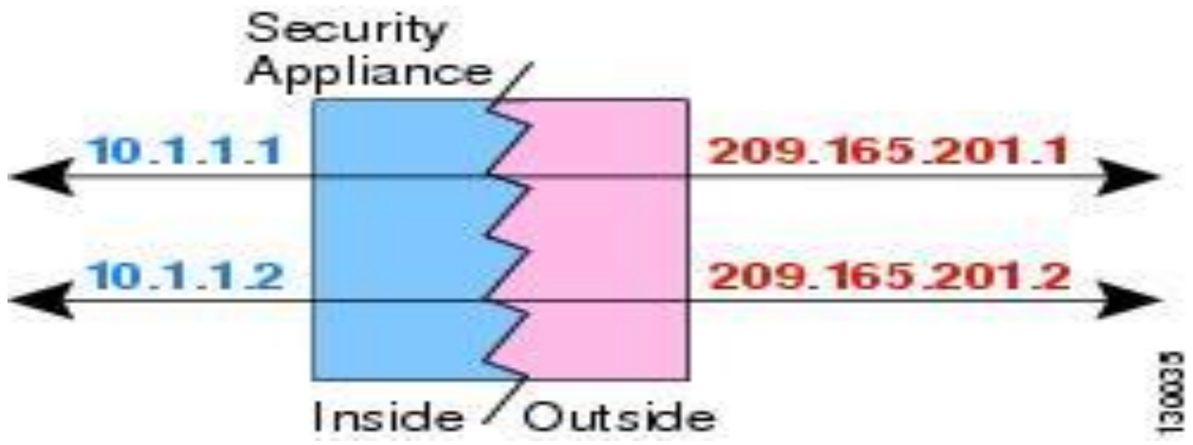


- hostname(config)# **nat (inside) 1 10.1.2.0 255.255.255.0**
- hostname(config)# **global (outside) 1 209.165.201.3-209.165.201.4**
- hostname(config)# **global (outside) 1 209.165.201.5**

Static NAT

- Static NAT creates a fixed translation of real address(es) to mapped address(es).With dynamic NAT and PAT, each host uses a different address or port for each subsequent translation. Because the mapped address is the same for each consecutive connection with static NAT, and a persistent translation rule exists, static NAT allows hosts on the destination network to initiate traffic to a translated host (if an access list exists that allows it).
- You need an equal number of mapped addresses as real addresses with static NAT.
- The translation is always active so both translated and remote hosts can originate connections, and the mapped address is statically assigned by the **static** command.

Static NAT sample

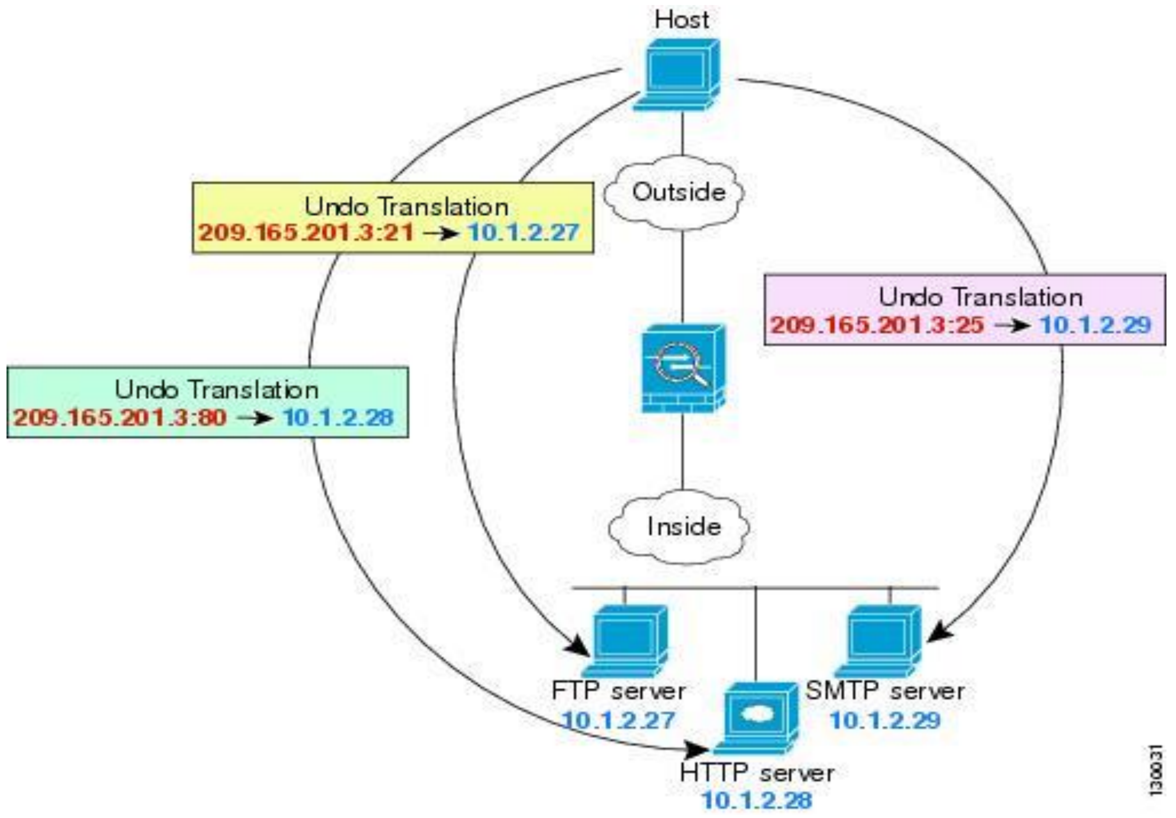


- hostname(config)# **static (inside,outside) 209.165.201.1 10.1.1.1 netmask 255.255.255.255**
- hostname(config)# **static (inside,outside) 209.165.201.2 10.1.1.2 netmask 255.255.255.255**

Static PAT

- Static PAT is the same as static NAT, except that it enables you to specify the protocol (TCP or UDP) and port for the real and mapped addresses. Static PAT enables you to identify the same mapped address across many different static statements, provided that the port is different for each statement.

Static PAT Sample

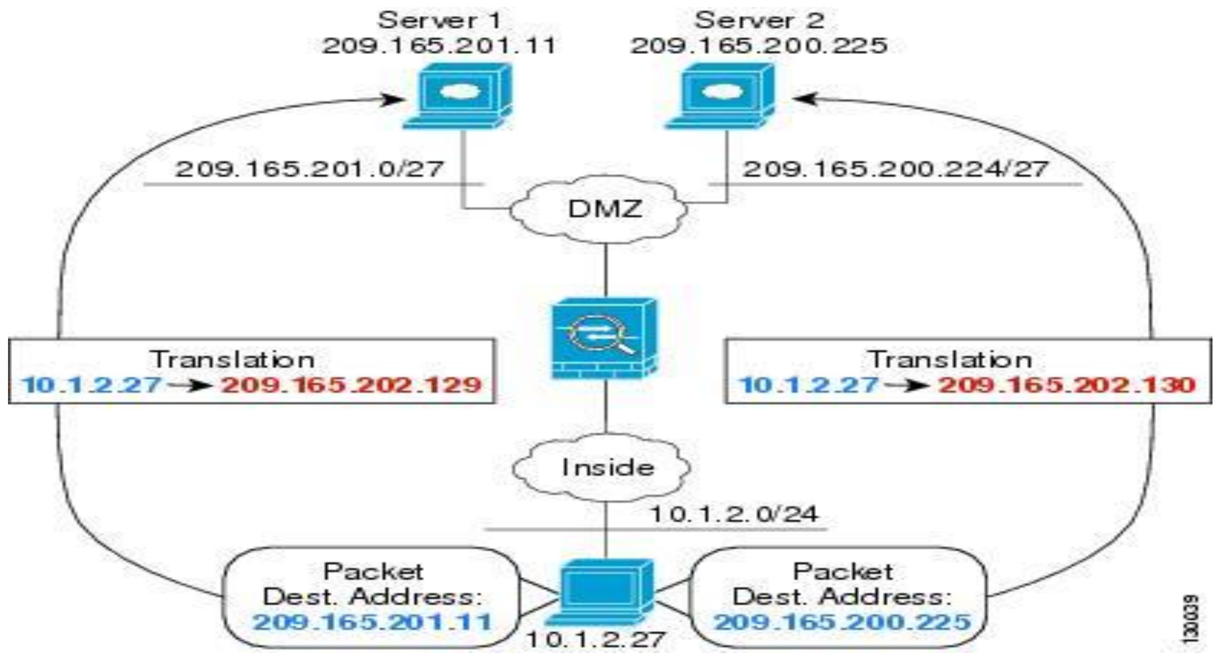


- hostname(config)# **static (inside,outside) tcp 209.165.201.3 ftp 10.1.2.27 ftp netmask 255.255.255.255**
- hostname(config)# **static (inside,outside) tcp 209.165.201.3 http 10.1.2.28 http netmask 255.255.255.255**
- hostname(config)# **static (inside,outside) tcp 209.165.201.3 smtp 10.1.2.29 smtp netmask 255.255.255.255**

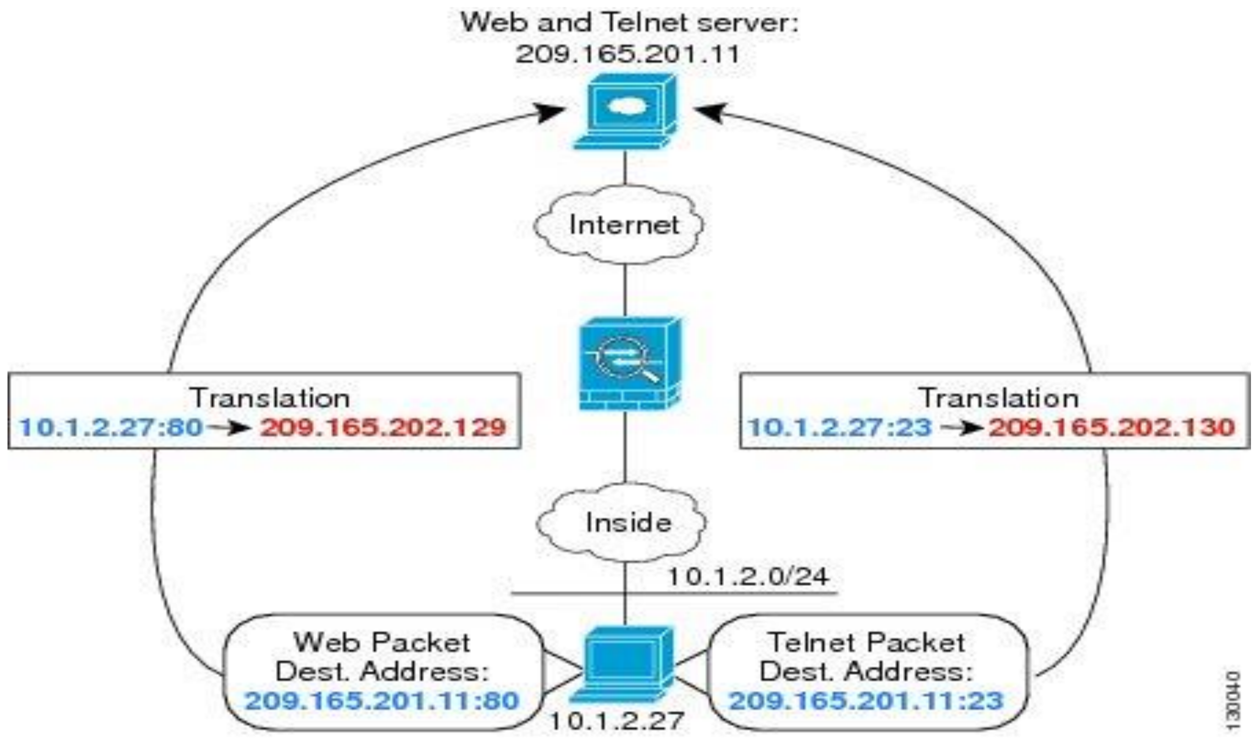
Policy NAT

- Policy NAT lets you identify real addresses for address translation by specifying the source and destination addresses in an extended access list. You can also optionally specify the source and destination ports. Regular NAT can only consider the source addresses, not the destination address . For example, with policy NAT you can translate the real address to mapped address A when it accesses server A, but also translate the real address to mapped address B when it accesses server B.
- All types of NAT support policy NAT, except for NAT exemption. NAT exemption uses an access list to identify the real addresses, but it differs from policy NAT in that the ports are not considered.

Policy NAT Samples



- hostname(config)# **access-list NET1 permit ip 10.1.2.0 255.255.255.0 209.165.201.0 255.255.255.224**
- hostname(config)# **access-list NET2 permit ip 10.1.2.0 255.255.255.0 209.165.200.224 255.255.255.224**
- hostname(config)# **nat (inside) 1 access-list NET1**
- hostname(config)# **global (outside) 1 209.165.202.129**
- hostname(config)# **nat (inside) 2 access-list NET2**
- hostname(config)# **global (outside) 2 209.165.202.130**



- hostname(config)# **access-list WEB permit tcp 10.1.2.0 255.255.255.0 209.165.201.11 255.255.255.255 eq 80**
- hostname(config)# **access-list TELNET permit tcp 10.1.2.0 255.255.255.0 209.165.201.11 255.255.255.255 eq 23**
- hostname(config)# **nat (inside) 1 access-list WEB**
- hostname(config)# **global (outside) 1 209.165.202.129**
- hostname(config)# **nat (inside) 2 access-list TELNET**
- hostname(config)# **global (outside) 2 209.165.202.130**

Static Policy NAT

- hostname(config)# **access-list NET1 permit ip 10.1.2.0 255.255.255.224 209.165.201.0**
- **255.255.255.224**
- hostname(config)# **static (inside,outside) 209.165.202.128 access-list NET1**

Identity NAT

- Identity NAT translates the real IP address to the same IP address. Only "translated" hosts can create NAT translations, and responding traffic is allowed back.
- When you configure identity NAT (which is similar to dynamic NAT), you do not limit translation for a host on specific interfaces; you must use identity NAT for connections through all interfaces. For example, you cannot choose to perform normal translation on real addresses when you access interface A and then use identity NAT when accessing interface B

Identity NAT Samples

- hostname(config)# nat (inside) 0 10.1.1.0 255.255.255.0

- hostname(config)# static (inside,outside) 10.1.1.0 10.1.1.0 255.255.255.0

- hostname(config)# static (outside,inside) 209.165.201.15 209.165.201.15 netmask 255.255.255.255

NAT exemption

- NAT exemption exempts addresses from translation and allows both translated and remote hosts to initiate connections. Like identity NAT, you do not limit translation for a host on specific interfaces; you must use NAT exemption for connections through all interfaces. However, NAT exemption does enable you to specify the real and destination addresses when determining the real addresses to translate (similar to policy NAT), so you have greater control using NAT exemption than identity NAT. However, unlike policy NAT, NAT exemption does not consider the ports in the access list. Use static identity NAT to consider ports in the access list.

NAT exemption Samples

- hostname(config)# access-list EXEMPT permit ip 10.1.2.0 255.255.255.0 any
- hostname(config)# nat (inside) 0 access-list EXEMPT

- hostname(config)# access-list NET1 permit ip 10.1.2.0 255.255.255.0 209.165.201.0 255.255.255.224
- hostname(config)# access-list NET1 permit ip 10.1.2.0 255.255.255.0 209.165.200.224 255.255.255.224
- hostname(config)# nat (inside) 0 access-list NET1

Order of NAT Rules Used to Match Real Addresses

- 1. NAT exemption—In order, **until the first match**.
- 2. Static NAT and Static PAT (regular and policy)—In order, **until the first match**. Static identity NAT is included in this category.
- 3. Policy dynamic NAT—In order**, until the first match**. Overlapping addresses are allowed.
- 4. Regular dynamic NAT—Best match. Regular identity NAT is included in this category. The order of the NAT rules does not matter; the NAT rule that **best matches** the real address is used.

Sample for match rule

Nat (inside) 1 0.0.0.0

Globle (outside) 1 interface

Nat (inside) 1 10.1.1.0

Global (outside) 1 202.2.2.254   // Best match, when 10.1.1.1 initiate the connection, it will be translated to 202.2.2.254 instead of the outside interface ip address.

Same security interface communication

- To permit communication between interfaces with equal security levels, or to allow traffic to enter and exit the same interface, use the **same-security-traffic** command in global configuration mode.
- **same-security-traffic permit** {**inter-interface | intra-interface**}
- **no same-security-traffic permit** {**inter-interface | intra-interface**}

NAT control in 8.2x

- By default, no nat control in 8.2x.
- When enabled, NAT control requires that packets traversing from an inside interface to an outside interface match a NAT rule; for any host on the inside network to access a host on the outside network, you must configure NAT to translate the inside host address.
- NAT control is used for NAT configurations defined with earlier versions of the ASA. The best practice is to use access rules for access control instead of relying on the absence of a NAT rule to prevent traffic through the ASA.
- Interfaces at the same security level are not required to use NAT to communicate.
- hostname(config)# nat-control

NAT In 9.x

- **Network Object NAT**

- **Twice NAT**

Network Object NAT

- All NAT rules that are configured as a parameter of a network object are considered to be *network object NAT* rules. Network object NAT is a quick and easy way to configure NAT for a network object, which can be a single IP address, a range of addresses, or a subnet. After you configure the network object, you can then identify the mapped address for that object.
- When a packet enters the adaptive security appliance, both the source and destination IP addresses are checked against the network object NAT rules. The source and destination address in the packet can be translated by separate rules if separate matches are made. These rules are not tied to each other; different combinations of rules can be used depending on the traffic.
- Because the rules are never paired, you cannot specify that a source address should be translated to A when going to destination X, but be translated to B when going to destination Y. Use twice NAT for that kind of functionality (twice NAT lets you identify the source and destination address in a single rule).

Dynamic NAT

- hostname(config)# **object network my-range-obj**
- hostname(config-network-object)# **range 10.2.2.1 10.2.2.10**

- hostname(config)# **object network my-inside-net**
- hostname(config-network-object)# **subnet 192.168.2.0 255.255.255.0**
- hostname(config-network-object)# **nat (inside,outside) dynamic my-range-obj**

Identity NAT

- The following example maps a host address to itself using an inline mapped address:
- hostname(config)# **object network my-host-obj1**
- hostname(config-network-object)# **host 10.1.1.1**
- hostname(config-network-object)# **nat (inside,outside) static 10.1.1.1**

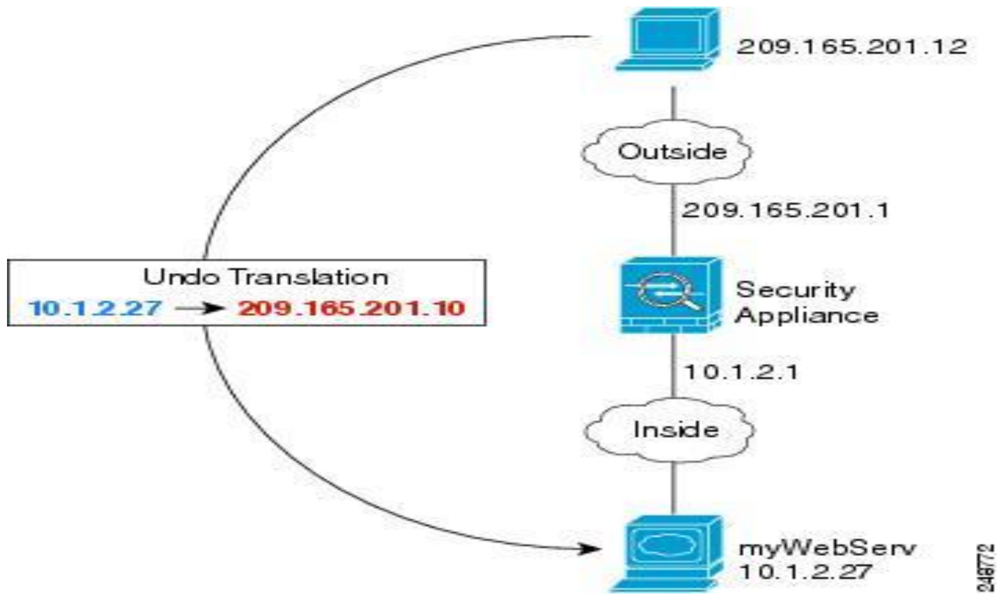
- The following example maps a host address to itself using a network object:
- hostname(config)# **object network my-host-obj1-identity**
- hostname(config-network-object)# **host 10.1.1.1**
- hostname(config-network-object)# **object network my-host-obj1**
- hostname(config-network-object)# **host 10.1.1.1**
- hostname(config-network-object)# **nat (inside,outside) static my-host-obj1-identity**

Dynamic PAT

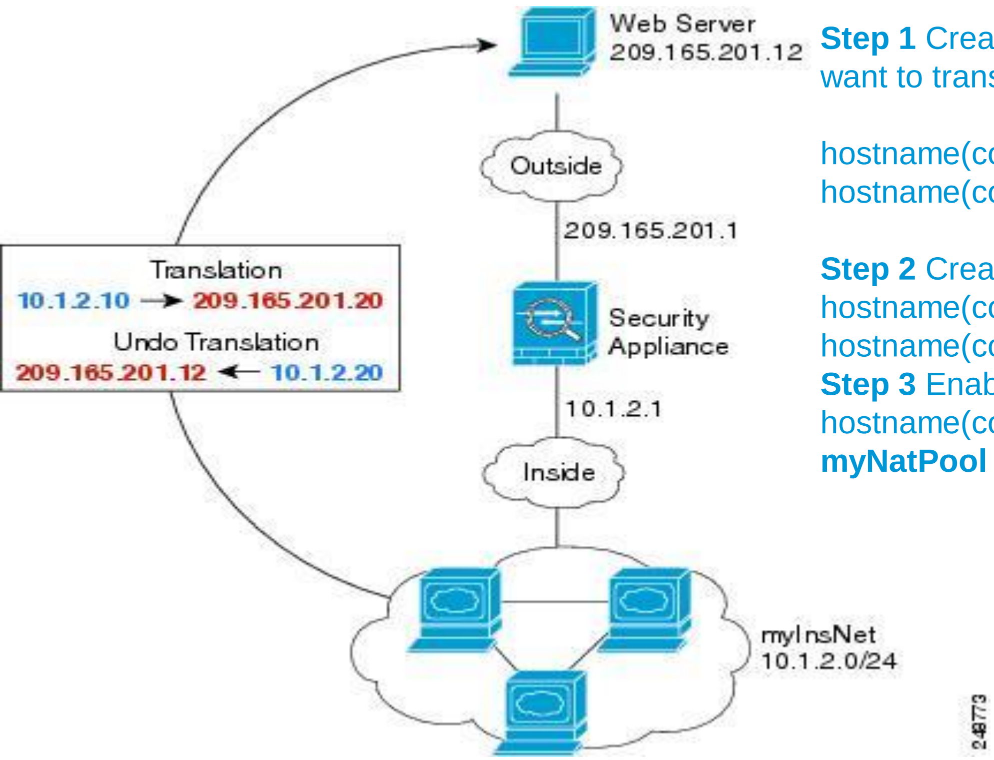
- hostname(config)# **object network my-inside-net**
- hostname(config-network-object)# **subnet 192.168.2.0 255.255.255.0**
- hostname(config-network-object)# **nat (inside,outside) dynamic 10.2.2.2**

- The following example c**onfigures dynamic PAT that hides the 192.168.2.0 network behind the outside interface address:**
- hostname(config)# **object network my-inside-net**
- hostname(config-network-object)# **subnet 192.168.2.0 255.255.255.0**
- hostname(config-network-object)# **nat (inside,outside) dynamic interface**

Object NAT Samples



- **Step 1** Create a network object for the internal web server:
- hostname(config)# **object network myWebServ**
- **Step 2** Define the web server address:
- hostname(config-network-object)# **host 10.1.2.27**
- **Step 3** Configure static NAT for the object:
- hostname(config-network-object)# **nat (inside,outside) static 209.165.201.10**

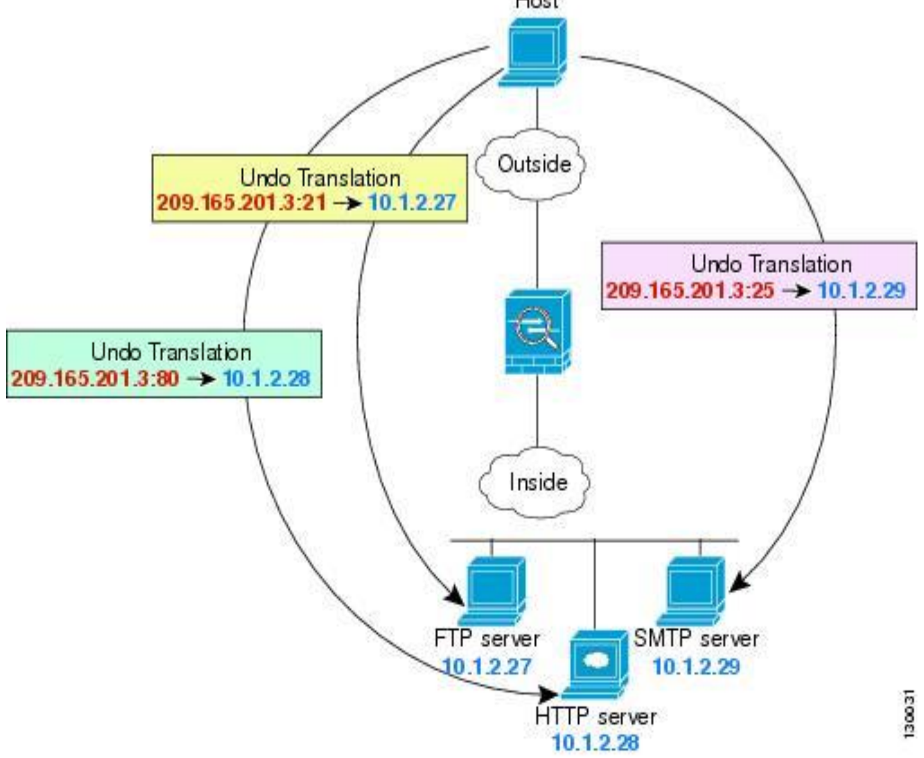


Step 1 Create a network object for the dynamic NAT pool to which you want to translate the inside addresses:

hostname(config)# **object network myNatPool**
hostname(config-network-object)# **range 209.165.201.20 209.165.201.30**

Step 2 Create a network object for the inside network:
hostname(config)# **object network myInsNet**
hostname(config-network-object)# **subnet 10.1.2.0 255.255.255.0**
Step 3 Enable dynamic NAT for the inside network:
hostname(config-network-object)# **nat (inside,outside) dynamic myNatPool**

- **Step 4** Create a network object for the outside web server:
- hostname(config)# **object network myWebServ**
- **Step 5** Define the web server address:
- hostname(config-network-object)# **host 209.165.201.12**
- **Step 6** Configure static NAT for the web server:
- hostname(config-network-object)# **nat (outside,inside) static 10.1.2.20**



Step 1 Create a network object for the FTP server address:
hostname(config)# **object network FTP_SERVER**

Step 2 Define the FTP server address, and configure static NAT with identity port translation for the FTP server:

hostname(config-network-object)# **host 10.1.2.27**
hostname(config-network-object)# **nat (inside,outside) static 209.165.201.3 service tcp ftp ftp**

Step 3 Create a network object for the HTTP server address:
hostname(config)# **object network HTTP_SERVER**

- **Step 4** Define the HTTP server address, and configure static NAT with identity port translation for the HTTP server:
- hostname(config-network-object)# **host 10.1.2.28**
- hostname(config-network-object)# **nat (inside,outside) static 209.165.201.3 service tcp http http**
- **Step 5** Create a network object for the SMTP server address:
- hostname(config)# **object network SMTP_SERVER**
- **Step 6** Define the SMTP server address, and configure static NAT with identity port translation for the SMTP server:
- hostname(config-network-object)# **host 10.1.2.29**
- hostname(config-network-object)# **nat (inside,outside) static 209.165.201.3 service tcp smtp smtp**

Twice NAT

- Twice NAT lets you identify both the source and destination address in a single rule. Specifying both the source and destination addresses lets you specify that a source address should be translated to A when going to destination X, but be translated to B when going to destination Y, for example.
- The destination address is optional. If you specify the destination address, you can either map it to itself (identity NAT), or you can map it to a different address. The destination mapping is always a static mapping.

Dynamic NAT

- **nat** [**(***real_ifc***,***mapped_ifc***)**] [*line* | {**after-auto** [*line*]}] **source dynamic** {*real_obj* | **any**} {*mapped_obj* [**interface**]} [**destination static** {*mapped_obj* | **interface**} *real_obj*] [**service** *mapped_dest_svc_obj real_dest_svc_obj*] [**dns**] [**inactive**] [**description** *desc*]
- Example:
- hostname(config)# object network MyInsNet

  hostname(config-network-object)# subnet 10.1.1.0 255.255.255.0
- hostname(config)# object network NAT_POOL

  hostname(config-network-object)# range 209.165.201.10 209.165.201.20
- hostname(config)# object network Server1 (Optional)

  hostname(config-network-object)# host 209.165.201.8
- hostname(config)# object network Server1_mapped (Optional)

  hostname(config-network-object)# host 10.1.1.67

- hostname(config)# object service REAL_SVC (Optional)

  hostname(config-service-object)# service tcp destination eq 80

- hostname(config)# object service MAPPED_SVC (Optional)

  hostname(config-service-object)# service tcp destination eq 8080

hostname(config)# nat (inside,outside) source dynamic MyInsNet NAT_POOL destination static Server1_mapped Server1 service MAPPED_SVC REAL_SVC

Keyword: Line and after-auto

- Line—By default, the NAT rule is added to the end of section 1 of the NAT table . If you want to add the rule into section 3 instead (after the network object NAT rules), then use the **after-auto** keyword. You can insert a rule anywhere in the applicable section using the *line* argument.

Dynamic PAT

- **nat** [**(***real_ifc***,***mapped_ifc***)**] [*line* | {**after-auto** [*line*]}] **source dynamic** {*real-obj* | **any**} {*mapped_obj* | **interface**} [**destination static** {*mapped_obj* | **interface**} *real_obj*] [**service** *mapped_dest_svc_obj real_dest_svc_obj*] [**dns**] [**inactive**] [**description** *desc*]

- hostname(config)# object network MyInsNet

  hostname(config-network-object)# subnet 10.1.1.0 255.255.255.0

- hostname(config)# object network Server1

  hostname(config-network-object)# host 209.165.201.8

hostname(config)# nat (inside,outside) source dynamic MyInsNet interface destination static Server1 Server1

Static NAT/PAT

- **nat** [**(***real_ifc***,***mapped_ifc***)**] [*line* | {**after-object** [*line*]}] **source static** *real_obj* [*mapped_obj* | **interface**] [**destination static** {*mapped_obj* | **interface**} *real_obj*] [**service** *real_src_mapped_dest_svc_obj mapped_src_real_dest_svc_obj*] [**dns**] [**unidirectional**] [**inactive**] [**description** *desc*]

- hostname(config)# object network MyInsNet

  hostname(config-network-object)# subnet 10.1.1.0 255.255.255.0

- hostname(config)# object network MyInsNet_mapped

  hostname(config-network-object)# subnet 192.168.1.0 255.255.255.0

- hostname(config)# object network Server1 (Optional)

  hostname(config-network-object)# host 209.165.201.8

- hostname(config)# object network Server1_mapped (Optional)

  hostname(config-network-object)# host 10.1.1.67

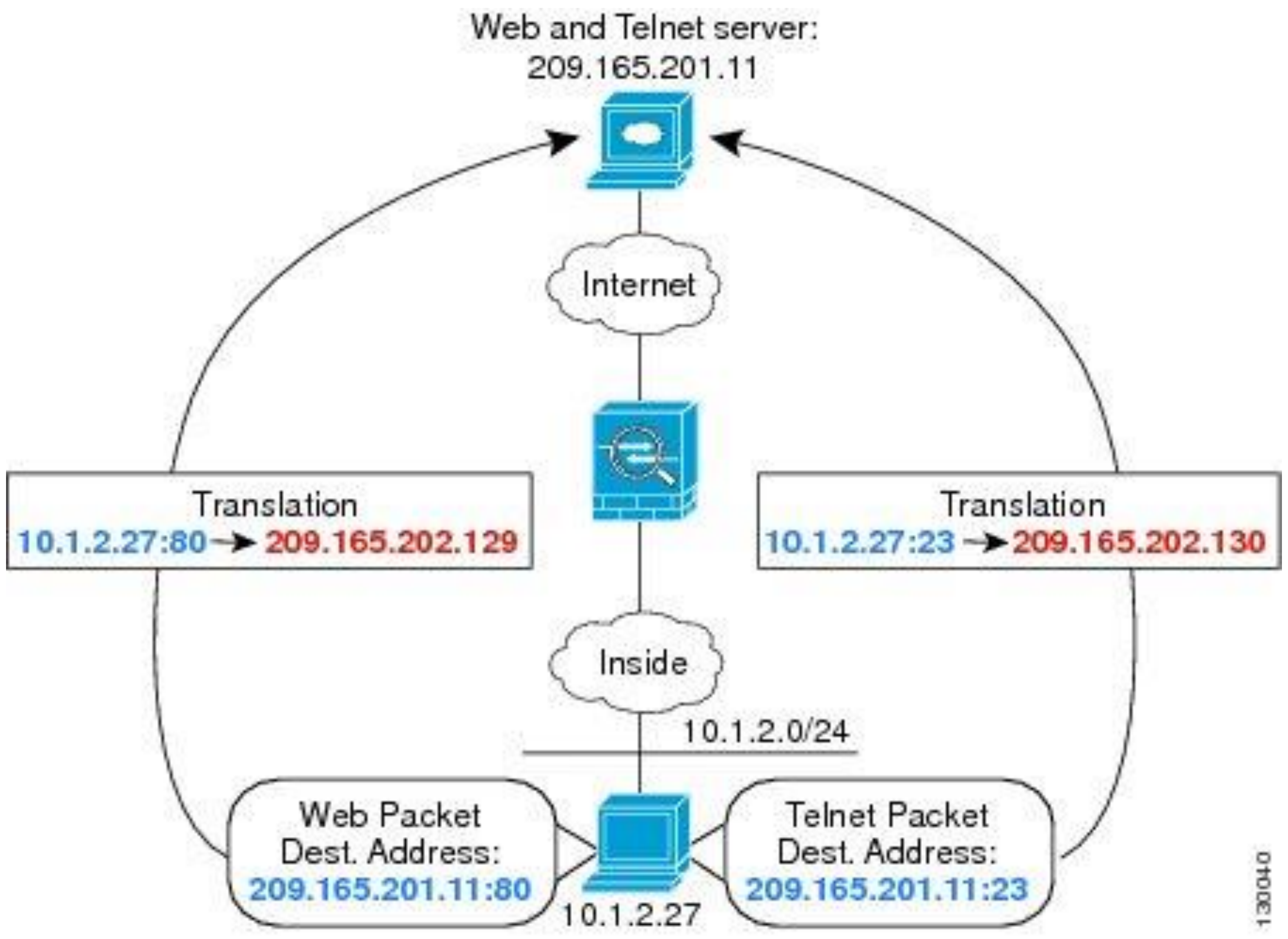
```
hostname(config)# object service REAL_SRC_SVC
hostname(config-service-object)# service tcp source eq 80
hostname(config)# object service MAPPED_SRC_SVC
hostname(config-service-object)# service tcp source eq 8080
```

hostname(config)# nat (inside,dmz) source static MyInsNet MyInsNet_mapped destination static Server1 Server1 service REAL_SRC_SVC MAPPED_SRC_SVC

Identity NAT

- hostname(config)# nat (inside,outside) source static MyInsNet MyInsNet destination static Server1 Server1

Twice NAT with Different Destination Ports Sample



- **Step 1** Add a network object for the inside network:

```
hostname(config)# object network myInsideNetwork
hostname(config-network-object)# subnet 10.1.2.0 255.255.255.0
```

- **Step 2** Add a network object for the Telnet/Web server:

```
hostname(config)# object network TelnetWebServer
hostname(config-network-object)# host 209.165.201.11
```

- **Step 3** Add a network object for the PAT address when using Telnet:

```
hostname(config)# object network PATaddress1
hostname(config-network-object)# host 209.165.202.129
```

- **Step 4** Add a service object for Telnet:

```
hostname(config)# object service TelnetObj
hostname(config-network-object)# service tcp destination eq telnet
```

- **Step 5** Configure the first twice NAT rule:

```
hostname(config)# nat (inside,outside) source dynamic myInsideNetwork PATaddress1 destination
static   TelnetWebServer TelnetWebServer service TelnetObj TelnetObj
```

- **Step 6** Add a network object for the PAT address when using HTTP:

```
hostname(config)# object network PATaddress2
 hostname(config-network-object)# host 209.165.202.130
```

- **Step 7** Add a service object for HTTP:

```
hostname(config)# object service HTTPObj
 hostname(config-network-object)# service tcp destination eq http
```

- **Step 8** Configure the second twice NAT rule:

```
hostname(config)# nat (inside,outside) source dynamic myInsideNetwork PATaddress2 destination static TelnetWebServer TelnetWebServer service HTTPObj HTTPObj
```

NAT Rule Order

- Network object NAT rules and twice NAT rules are stored in a single table that is divided into three sections. Section 1 rules are applied first, then section 2, and finally section 3.
- Section 1: Twice NAT

Applied on a first match basis, in the order they appear in the configuration. By default, twice NAT rules are added to section 1.

Note If you configure VPN, the client dynamically adds invisible NAT rules to the end of this section. Be sure that you do not configure a twice NAT rule in this section that might match your VPN traffic, instead of matching the invisible rule. If VPN does not work due to NAT failure, consider adding twice NAT rules to section 3 instead.

NAT Rule Order (Continued)

- Section 2:Network object NAT

Section 2 rules are applied in the following order, as automatically determined by the adaptive security appliance:

- **1.** Static rules.
- **2.** Dynamic rules.
- Within each rule type, the following ordering guidelines are used:
- **a.** Quantity of real IP addresses—From smallest to largest. For example, an object with one address will be assessed before an object with 10 addresses.
- **b.** For quantities that are the same, then the IP address number is used, from lowest to highest. For example, 10.1.1.0 is assessed before 11.1.1.0.
- **c.** If the same IP address is used, then the name of the network object is used, in alphabetical order. For example, abracadabra is assessed before catwoman.

NAT Rule Order (Continued)

- Section 3: Twice NAT

Section 3 rules are applied on a first match basis, in the order they appear in the configuration. You can specify whether to add a twice NAT rule to section 3 when you add the rule.

The final order determined by asa?

For section 2 rules for example, you have the following IP addresses defined within network objects:

- 192.168.1.0/24 (static)
- 192.168.1.0/24 (dynamic)
- 10.1.1.0/24 (static)
- 192.168.1.1/32 (static)
- 172.16.1.0/24 (dynamic) (object def)
- 172.16.1.0/24 (dynamic) (object abc)

The Answer is:

- The resultant ordering would be:
- 192.168.1.1/32 (static)
- 10.1.1.0/24 (static)
- 192.168.1.0/24 (static)
- 172.16.1.0/24 (dynamic) (object abc)
- 172.16.1.0/24 (dynamic) (object def)
- 192.168.1.0/24 (dynamic)

The major changes in Version 8.3 and later

- Real IP addresses in access lists, where access lists are used in supported features—When using NAT or PAT, you used to have to specify the *mapped* addresses and ports in an access list for all features that use access lists. Now, for several supported features, you must use the real, untranslated IP address and ports. (Other features continue to use the mapped IP address).

- **Features That Use Real IP Addresses**

- The following commands and features now use real IP addresses in the access lists. All of the **access-list** commands used for these features are automatically migrated unless otherwise noted. For access lists that use network object groups (the **object-group network** command), the IP addresses within the object group are migrated to the real IP addresses.

  •**access-group** command

  •Modular Policy Framework **match access-list** command

  •Botnet Traffic Filter **dynamic-filter enable classify-list** command

  •AAA **aaa ... match** commands

  •WCCP **wccp redirect-list group-list** command

access-group change

- **Old Configuration**

static (inside,outside) 172.23.57.1 10.50.50.50 netmask 255.255.255.255

access-list 1 permit ip any host **172.23.57.1**

access-group 1 in interface outside

- **Migrated Configuration**

access-list 1 permit ip any host **10.50.50.50**

access-group 1 in interface outside

Dynamic NAT with AAA

- **Old Configuration**

global (outside) 1 172.23.57.171-172.23.57.172

nat (inside) 1 10.50.50.0 255.255.255.0

nat (dmz) 1 192.168.4.0 255.255.255.0

object-group network mapped_pool

network-object host 172.23.57.171

network-object host 172.23.57.172

access-list 1 permit udp any **object-group mapped_pool**

aaa authentication match 1 outside TEST_SERVER

- **Migrated Configuration**

access-list 1 permit udp any **10.50.50.0 255.255.255.0**

access-list 1 permit udp any **192.168.4.0 255.255.255.0**

Service policy

- **Old Configuration**

```
 *static (inside,outside) tcp 172.23.57.170 6021 10.50.50.50 21*
 access-list 1 permit tcp any host 172.23.57.170 eq 6021
  class-map ftpclass
       match access-list 1
 policy-map ftp_pol
   class ftpclass
    inspect ftp
service-policy ftp_pol interface outside
```

- **Migrated Configuration**

```
access-list 1 permit tcp any host **10.50.50.50 eq ftp**
class-map ftpclass
  match access-list 1
policy-map ftp_pol
  class ftpclass
  inspect ftp
  service-policy ftp_pol interface outside
```

8.2x and 8.3x and later nat statements migration

Regular Static NAT

- **Old Configuration**

```
 static (inside,outside) 209.165.201.15 10.1.1.6 netmask
255.255.255.255
```

- **Migrated Configuration**

```
object network obj-10.1.1.6
   host 10.1.1.6
   nat (inside,outside) static 209.165.201.15
```

8.2x nat statements compared with 8.3x and later

- **Regular Static PAT**

- Old Configuration

  static (inside,outside) tcp 10.1.2.45 80 10.1.1.16 8080 netmask 255.255.255.255

- **Migrated Configuration**

  object network obj-10.1.1.16

  host 10.1.1.16

- nat (inside,outside) static 10.1.2.45 service tcp 8080 www

- **Static Policy NAT**
- **Old Configuration**

```
access-list NET1 permit ip host 10.1.2.27 10.76.5.0 255.255.255.224
static (inside,outside) 209.165.202.129 access-list NET1
```

- **Migrated Configuration**

```
object network obj-10.1.2.27
 host 10.1.2.27
object network obj-209.165.202.129
 host 209.165.202.129
object network obj-10.76.5.0
subnet 10.76.5.0 255.255.255.224
 nat (inside,outside) source static obj-10.1.2.27 obj-209.165.202.129 destination static obj-10.76.5.0 obj-10.76.5.0
```

- **Regular Dynamic PAT**
- **Old Configuration**

```
nat (inside) 1 192.168.1.0 255.255.255.0
nat (dmz) 1 10.1.1.0 255.255.255.0
 global (outside) 1 209.165.201.3
```

- **Migrated Configuration**

```
  object network obj-192.168.1.0
     subnet 192.168.1.0 255.255.255.0
     nat (inside,outside) dynamic 209.165.201.3
object network obj-10.1.1.0
  subnet 10.1.1.0 255.255.255.0
  nat (dmz,outside) dynamic 209.165.201.3
```

Features That Continue to Use Mapped IP Addresses

- The following features use access lists, but these access lists will continue to use the mapped values as seen on an interface:

  •IPSec access lists

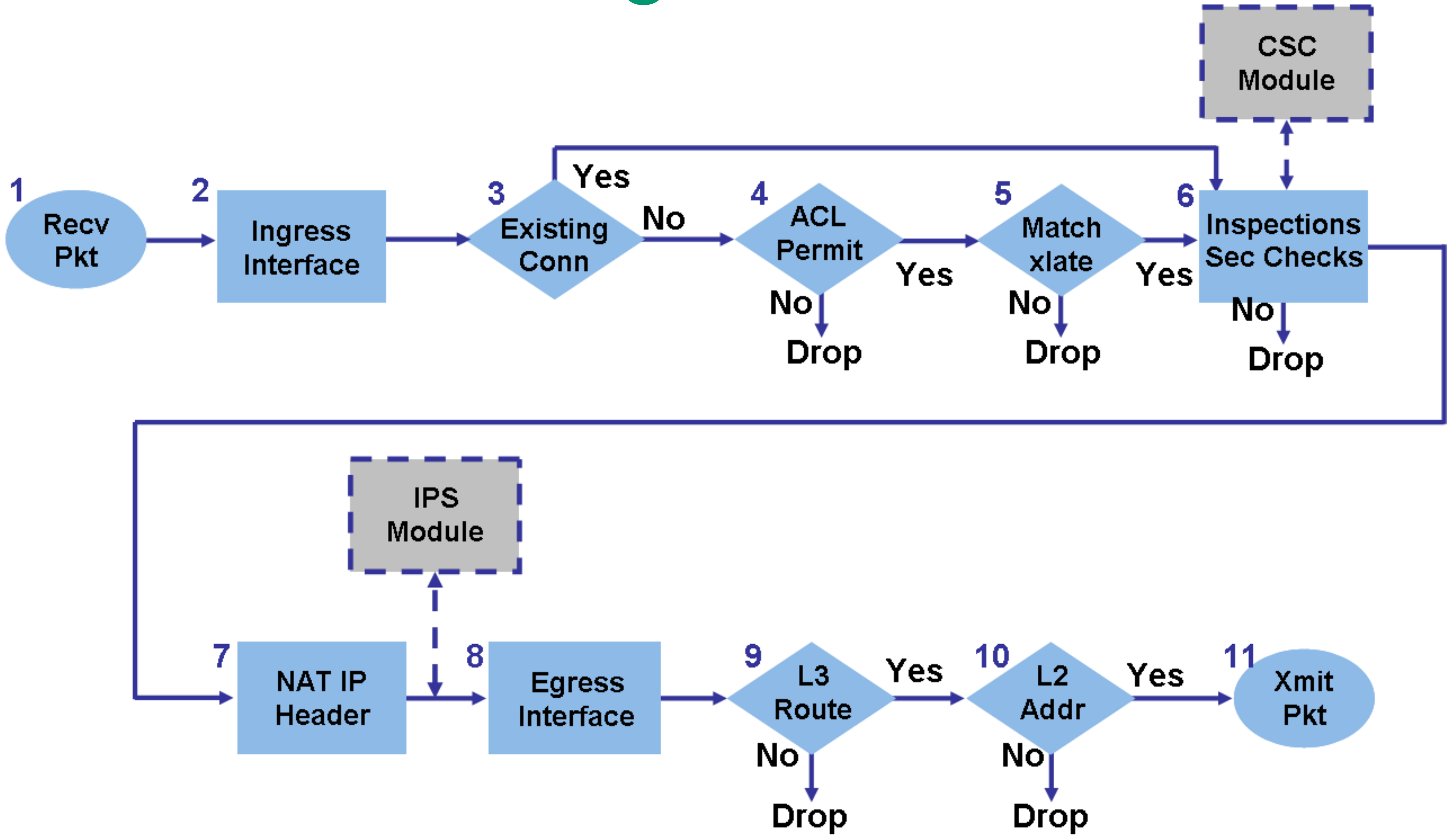
  •**capture** command access lists

  •Per-user access lists

  •Routing protocol access lists

  •All other feature access lists...

Data forwarding in asa



- Packet is reached at the ingress interface.
- Once the packet reaches the internal buffer of the interface, the input counter of the interface is incremented by one.
- Cisco ASA will first verify if this is an existing connection by looking at its internal connection table details. If the packet flow matches an existing connection, then the access-control list (ACL) check is bypassed, and the packet is moved forward.
- If packet flow does not match an existing connection, then TCP state is verified. If it is a SYN packet or UDP packet, then the connection counter is incremented by one and the packet is sent for an ACL check. If it is not a SYN packet, the packet is dropped and the event is logged.
- The packet is verified for the translation rules. If a packet passes through this check, then a connection entry is created for this flow, and the packet moves forward. Otherwise, the packet is dropped and the information is logged.
- The packet is subjected to an Inspection Check. This inspection verifies whether or not this specific packet flow is in compliance with the protocol. Cisco ASA has a built-in inspection engine that inspects each connection as per its pre-defined set of application-level functionalities. If it passed the inspection, it is moved forward. Otherwise, the packet is dropped and the information is logged.

Additional Security-Checks will be implemented if a CSC module is involved.

- The IP header information is translated as per the NAT/PAT rule and checksums are updated accordingly. The packet is forwarded to AIP-SSM for IPS related security checks, when the AIP module is involved.
- The packet is forwarded to the egress interface based on the translation rules. If no egress interface is specified in the translation rule, then the destination interface is decided based on global route lookup.
- On the egress interface, the interface route lookup is performed. Remember, the egress interface is determined by the translation rule that will take the priority.
- Once a Layer 3 route has been found and the next hop identified, Layer 2 resolution is performed. Layer 2 rewrite of MAC header happens at this stage.
- The packet is transmitted on wire, and Interface counters increment on the egress interface.

During the steps, the syslog msg you may see

- %ASA-6-106015: Deny TCP (no connection) from 192.168.30.253/34820 to 192.168.20.253/23 flags ACK on interface inside1
- %ASA-3-305005: No translation group found for tcp src inside1:192.168.30.253/53254 dst dmz:192.168.20.253/23
- %ASA-4-106023: Deny tcp src dmz:192.168.20.253/40621 dst inside1:192.168.30.253/23 by access-group “outside" [0xd4abca4, 0x0]
- %ASA-6-110003: Routing failed to locate next hop for TCP from inside:192.168.10.253/54790 to outside:9.9.9.9/23
- 34 2014/05/21 08:51:36.653 CST   192.168.255.130   May 20 2014 20:51:36: %ASA-6-302014: Teardown TCP connection 3662656 for DMZ:192.168.10.56/3389 to INSIDE:192.168.2.245/23389 duration 0:13:30 bytes 100201 Flow closed by inspection

The common reason for no connection

- **End host sends unlegal packets, for example, the first packet is Ack packet instead of the syn packet.**
- **For bug issue, asa closes connection, but the end host doesn't know it, the subsequent packets are denied by asa as "no connection".**

CSCuj54806/CSCui77398 ICMP inspection closes TCP conns with "Flow closed   by inspection"

Match condition: "inspect icmp" is configured .

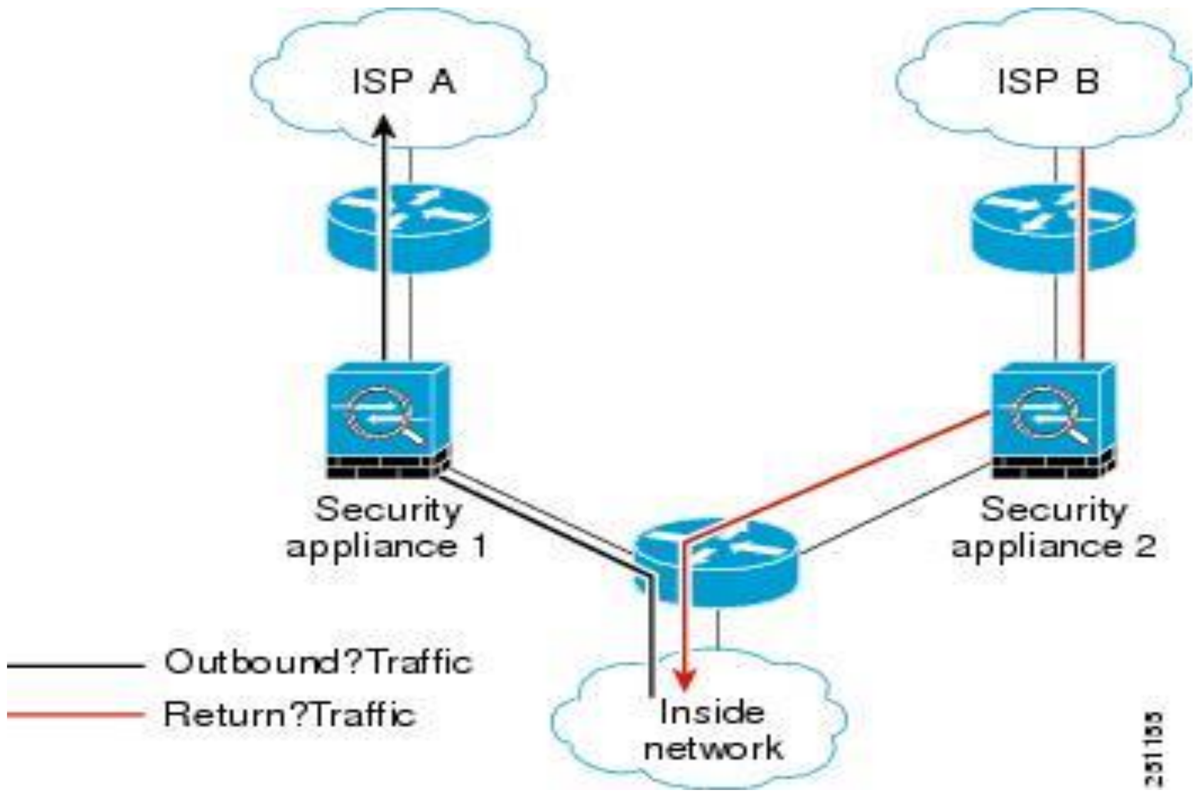
Match phenomenon: tcp application will disconnect randomly, for example, RDP and ftp transmission.

Resolved version:

009.000(003.005) 009.001(003) 008.004(007.001) 008.007(001.008)   009.000(003.100)
100.010(001.021) 009.000(004) 009.001(004) 009.002(000.099) 009.002(001

The common reason for no connection(continue)

- Asymmetric Routing



- If you have asymmetric routing configured on upstream routers, and traffic alternates between two ASAs, then you can configure TCP state bypass for specific traffic. TCP state bypass alters the way sessions are established in the fast path and disables the fast path checks. This feature treats TCP traffic much as it treats a UDP connection: when a non-SYN packet matching the specified networks enters the ASA, and there is not an fast path entry, then the packet goes through the session management path to establish the connection in the fast path. Once in the fast path, the traffic bypasses the fast path checks.

- Configuration example:

```
access-list  test  perm tcp host  202.1.1.1 host  10.1.1.1

 class-map test

    match access-list   test



 policy-map global_policy

    class test

         set connection advanced-options tcp-state-bypass
```

Common tools used for troubleshooting

- Enable syslog server for asa or enable buffer logging for asa

```
logging enable
logging timestamp
logging trap informational
logging host inside 10.1.1.1
```

If you don’t have a syslog server setup, pls try to enable the buffer logging as follows:

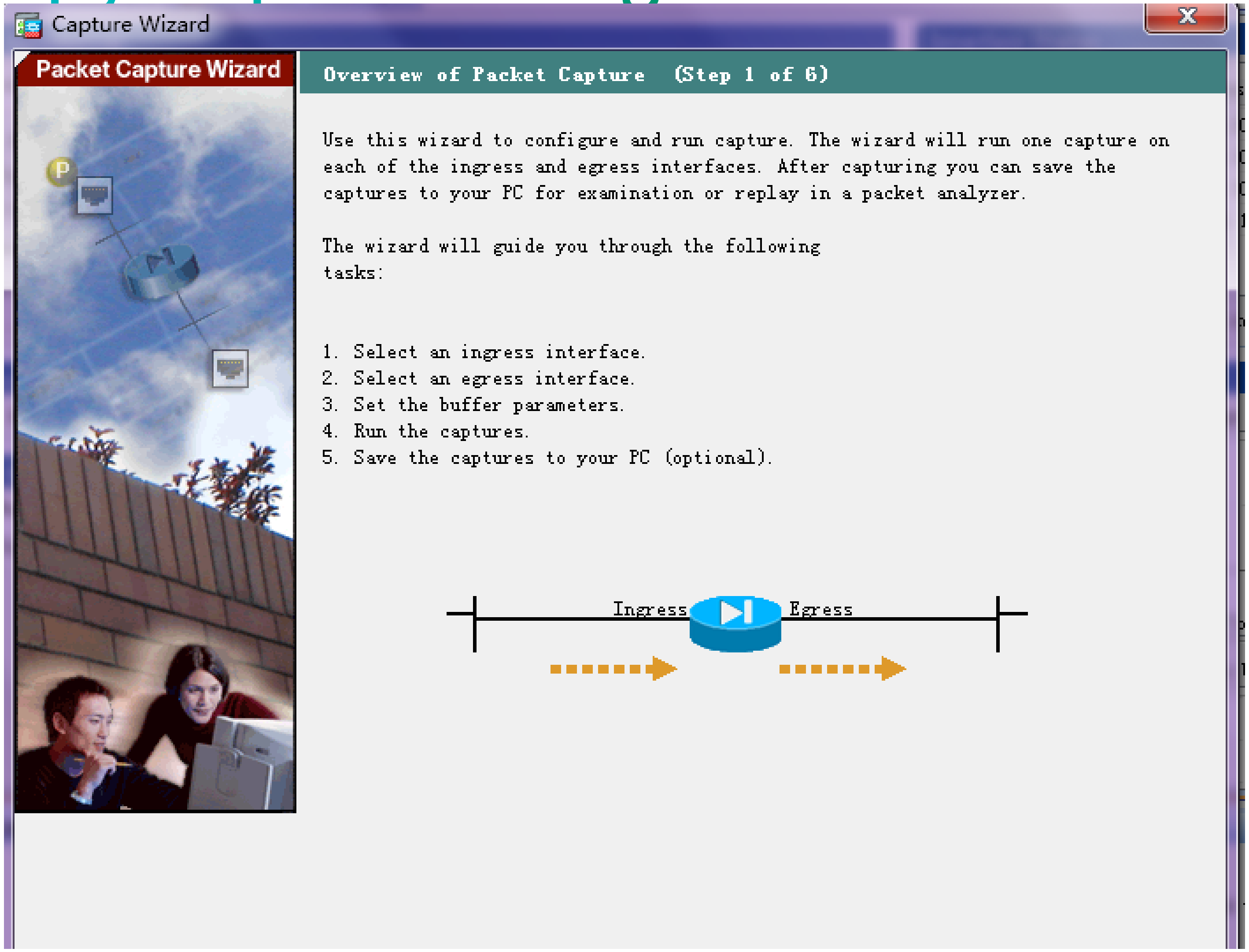
```
logging enable
logging timestamp
logging buffered informational
logging buffer-size 1048576
```

Common tools used for troubleshooting (continued)

- Packet capture

Capturing packets is useful when you troubleshoot connectivity problems or monitor suspicious activity. You can create multiple captures. In order to enable packet capture capabilities for packet sniffing and network fault isolation, use the **capture** command in privileged EXEC mode. In order to disable packet capture capabilities, use the **no form** of this command. In order to view the packet capture, use the show **capture** name command. In order to save the capture to a file, use the **copy capture** command.

Copy capture using ASDM



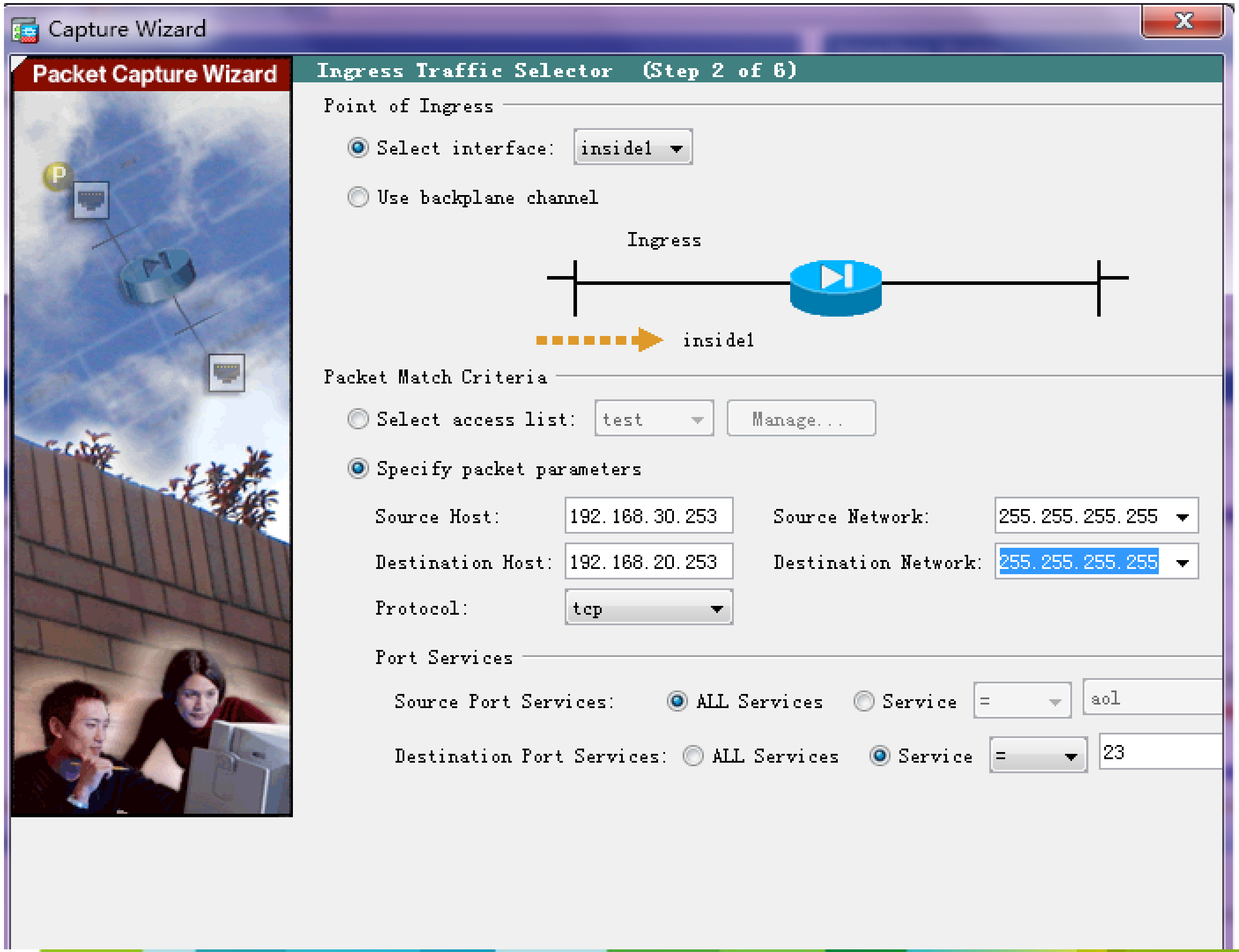
Overview of Packet Capture (Step 1 of 6)

Use this wizard to configure and run capture. The wizard will run one capture on each of the ingress and egress interfaces. After capturing you can save the captures to your PC for examination or replay in a packet analyzer.

The wizard will guide you through the following tasks:

1. Select an ingress interface.
2. Select an egress interface.
3. Set the buffer parameters.
4. Run the captures.
5. Save the captures to your PC (optional).

Copy capture using ASDM



Capture Wizard

Packet Capture Wizard

Ingress Traffic Selector (Step 2 of 6)

Point of Ingress

[✓ Select interface: inside1]

Use backplane channel

Packet Match Criteria

Select access list: test

Manage...

[✓ Specify packet parameters]

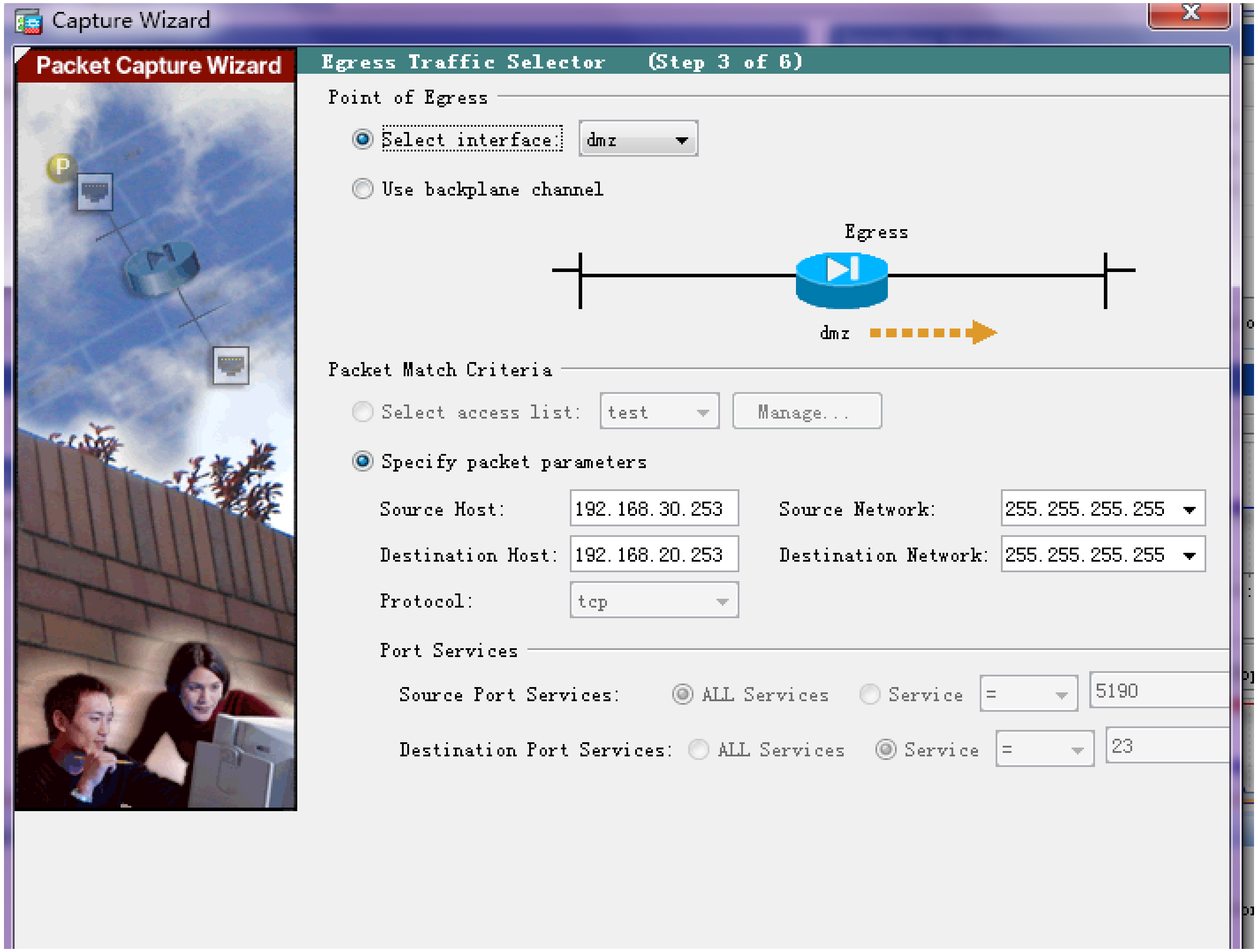
Source Host:	192.168.30.253	Source Network:	255.255.255.255
Destination Host:	192.168.20.253	Destination Network:	255.255.255.255
Protocol:	tcp		

Port Services

Source Port Services: [✓ ALL Services] Service = aol

Destination Port Services: ALL Services [✓ Service] = 23

Copy capture using ASDM



Capture Wizard

Packet Capture Wizard

Egress Traffic Selector (Step 3 of 6)

Point of Egress

[✓ Select interface:] dmz

Use backplane channel

Packet Match Criteria

Select access list: test   Manage...

[✓ Specify packet parameters]

Source Host: 192.168.30.253   Source Network: 255.255.255.255

Destination Host: 192.168.20.253   Destination Network: 255.255.255.255

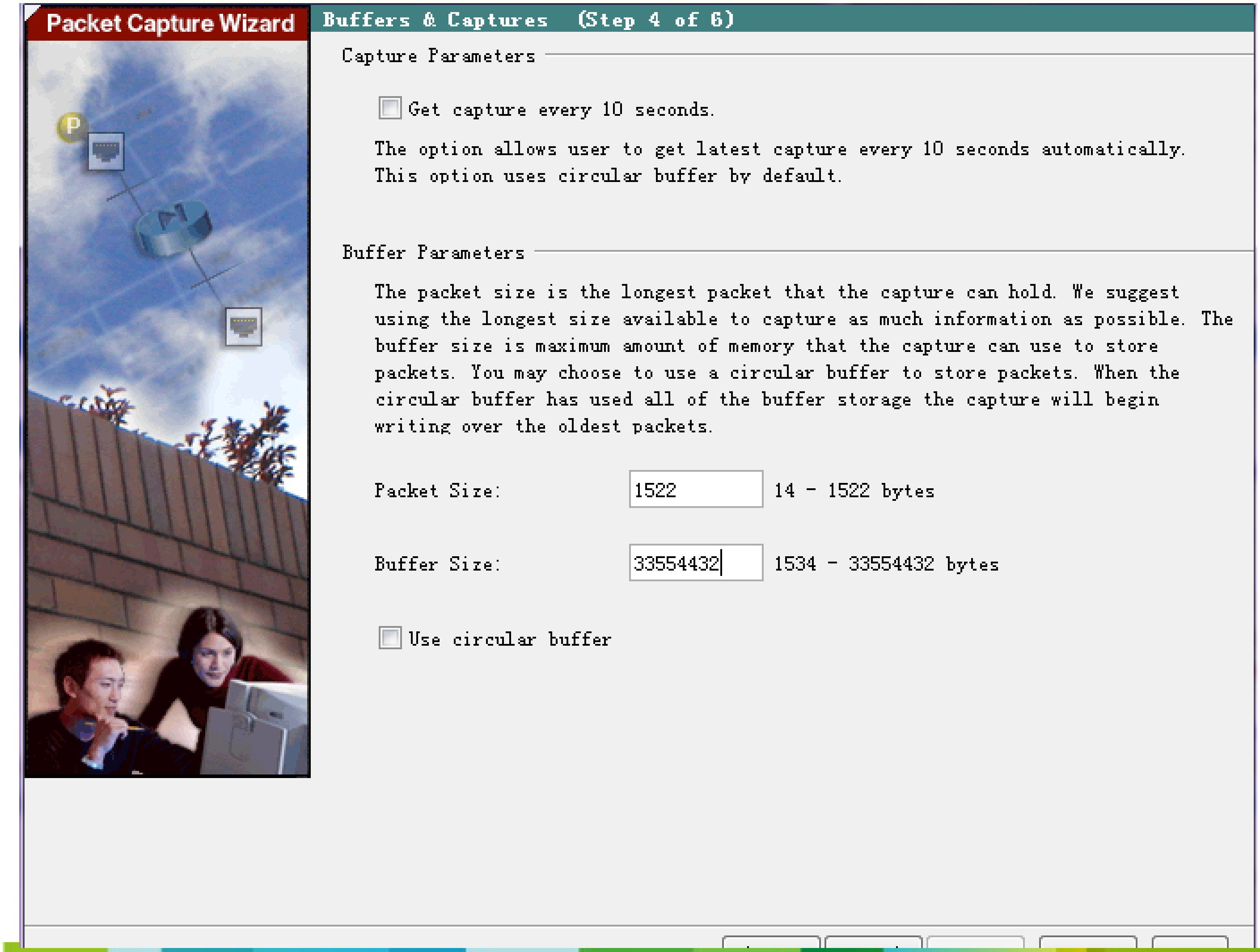
Protocol: tcp

Port Services

Source Port Services: [✓ ALL Services] Service = 5190

Destination Port Services: ALL Services [✓ Service] = 23

Copy capture using ASDM



Buffers & Captures (Step 4 of 6)

Capture Parameters

[ ] Get capture every 10 seconds.

The option allows user to get latest capture every 10 seconds automatically. This option uses circular buffer by default.

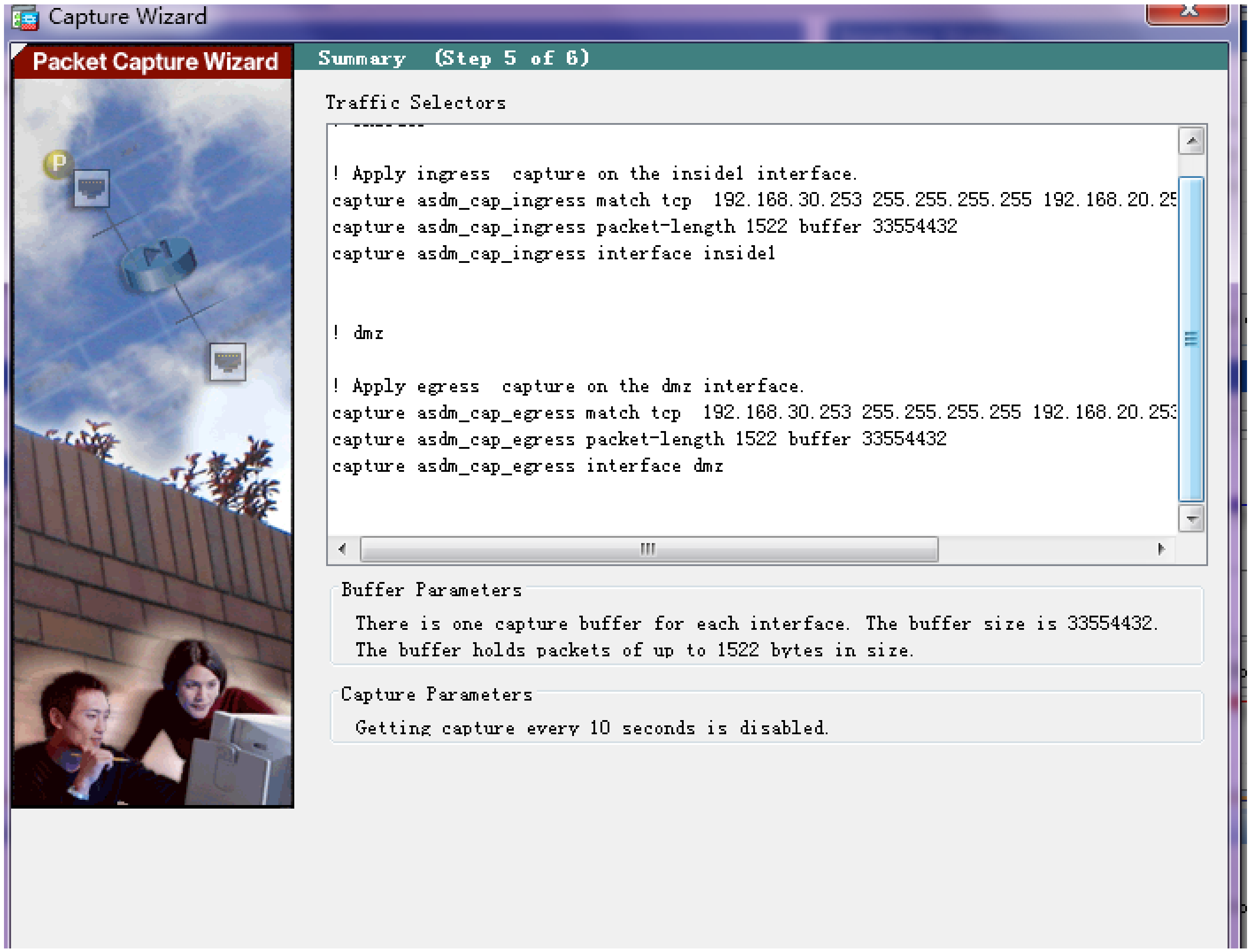
Buffer Parameters

The packet size is the longest packet that the capture can hold. We suggest using the longest size available to capture as much information as possible. The buffer size is maximum amount of memory that the capture can use to store packets. You may choose to use a circular buffer to store packets. When the circular buffer has used all of the buffer storage the capture will begin writing over the oldest packets.

Packet Size:	1522	14 - 1522 bytes
Buffer Size:	33554432	1534 - 33554432 bytes

[ ] Use circular buffer

Copy capture using ASDM



Traffic Selectors

```
! Apply ingress  capture on the inside1 interface.
capture asdm_cap_ingress match tcp  192.168.30.253 255.255.255.255 192.168.20.25
capture asdm_cap_ingress packet-length 1522 buffer 33554432
capture asdm_cap_ingress interface inside1


! dmz

! Apply egress  capture on the dmz interface.
capture asdm_cap_egress match tcp  192.168.30.253 255.255.255.255 192.168.20.253
capture asdm_cap_egress packet-length 1522 buffer 33554432
capture asdm_cap_egress interface dmz
```

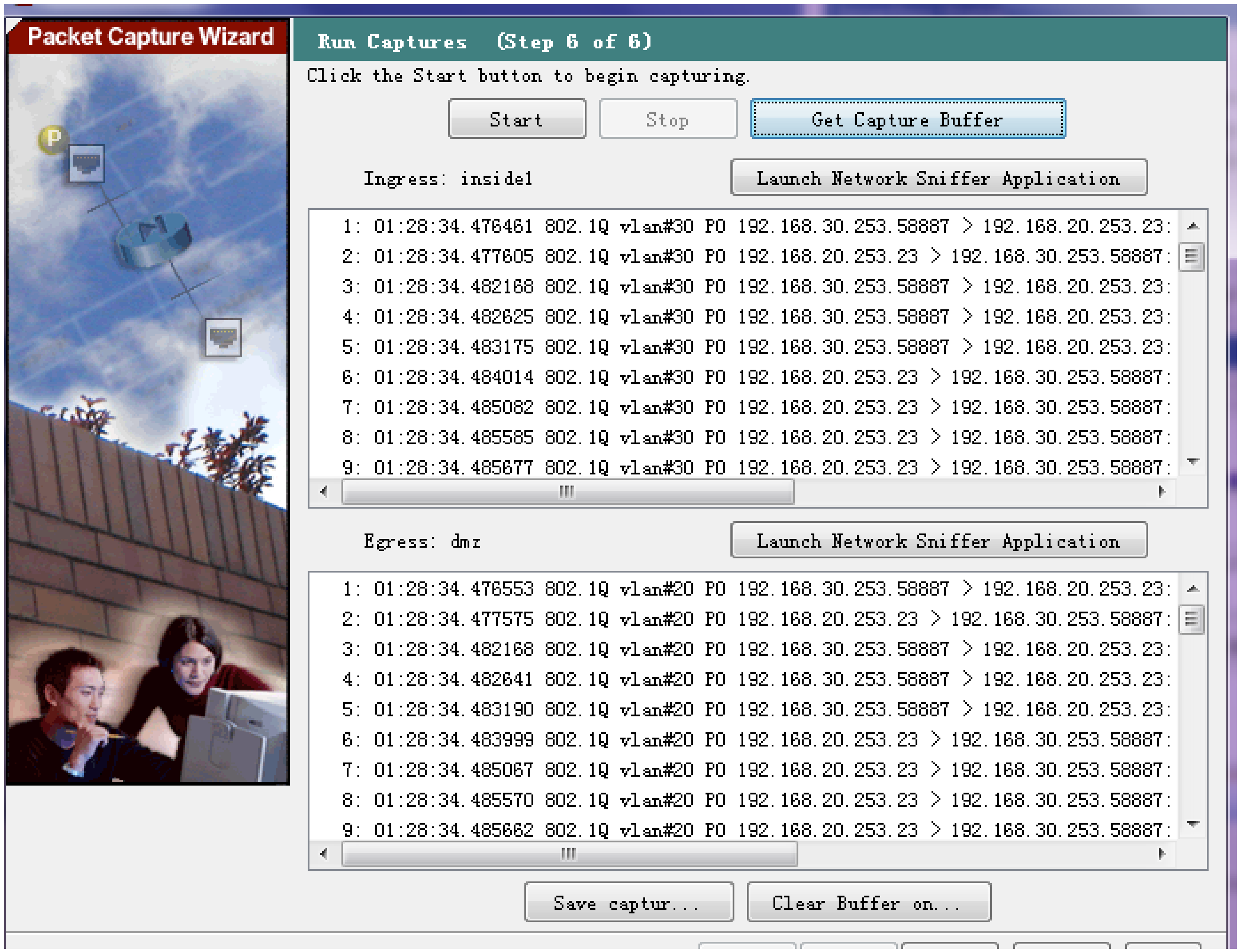
Buffer Parameters

There is one capture buffer for each interface. The buffer size is 33554432. The buffer holds packets of up to 1522 bytes in size.

Capture Parameters

Getting capture every 10 seconds is disabled.

Copy capture using ASDM


Packet Capture Wizard
Run Captures (Step 6 of 6)
Start
Stop
Get Capture Buffer
Launch Network Sniffer Application
Launch Network Sniffer Application
Save captur...
Clear Buffer on...

Click the Start button to begin capturing.

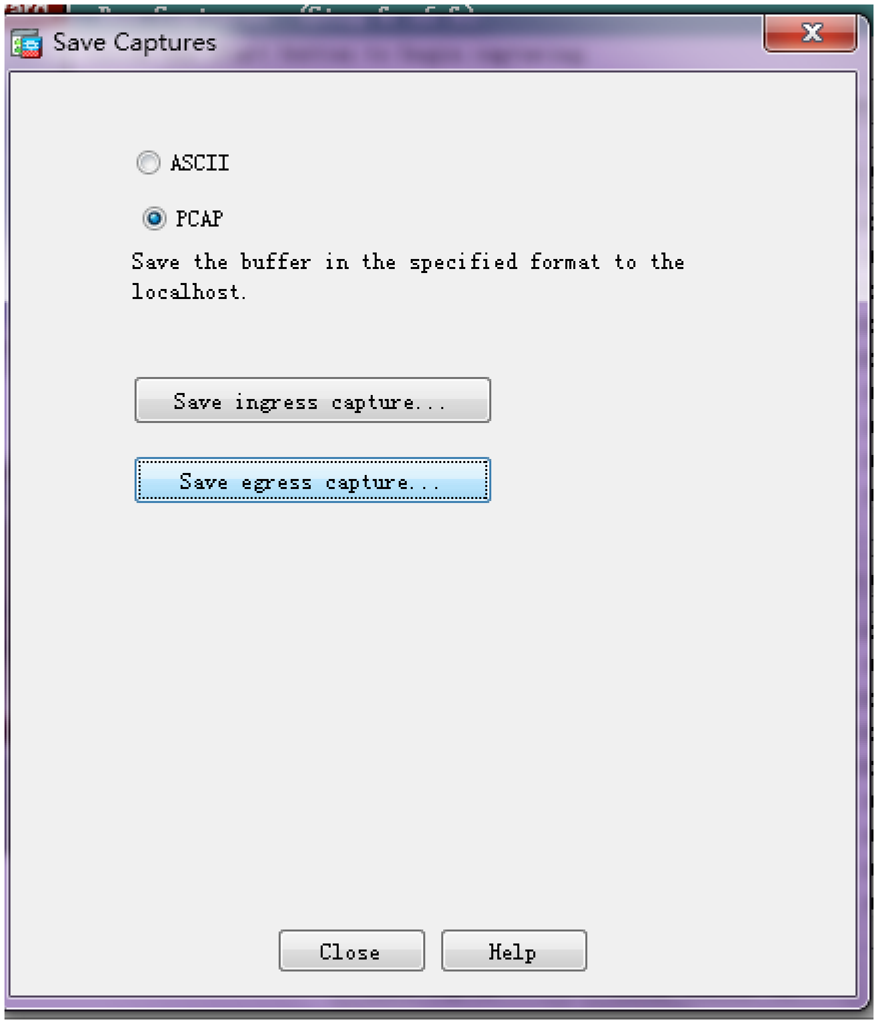
Ingress: inside1

```
1: 01:28:34.476461 802.1Q vlan#30 P0 192.168.30.253.58887 > 192.168.20.253.23:
2: 01:28:34.477605 802.1Q vlan#30 P0 192.168.20.253.23 > 192.168.30.253.58887:
3: 01:28:34.482168 802.1Q vlan#30 P0 192.168.30.253.58887 > 192.168.20.253.23:
4: 01:28:34.482625 802.1Q vlan#30 P0 192.168.30.253.58887 > 192.168.20.253.23:
5: 01:28:34.483175 802.1Q vlan#30 P0 192.168.30.253.58887 > 192.168.20.253.23:
6: 01:28:34.484014 802.1Q vlan#30 P0 192.168.20.253.23 > 192.168.30.253.58887:
7: 01:28:34.485082 802.1Q vlan#30 P0 192.168.20.253.23 > 192.168.30.253.58887:
8: 01:28:34.485585 802.1Q vlan#30 P0 192.168.20.253.23 > 192.168.30.253.58887:
9: 01:28:34.485677 802.1Q vlan#30 P0 192.168.20.253.23 > 192.168.30.253.58887:
```

Egress: dmz

```
1: 01:28:34.476553 802.1Q vlan#20 P0 192.168.30.253.58887 > 192.168.20.253.23:
2: 01:28:34.477575 802.1Q vlan#20 P0 192.168.20.253.23 > 192.168.30.253.58887:
3: 01:28:34.482168 802.1Q vlan#20 P0 192.168.30.253.58887 > 192.168.20.253.23:
4: 01:28:34.482641 802.1Q vlan#20 P0 192.168.30.253.58887 > 192.168.20.253.23:
5: 01:28:34.483190 802.1Q vlan#20 P0 192.168.30.253.58887 > 192.168.20.253.23:
6: 01:28:34.483999 802.1Q vlan#20 P0 192.168.20.253.23 > 192.168.30.253.58887:
7: 01:28:34.485067 802.1Q vlan#20 P0 192.168.20.253.23 > 192.168.30.253.58887:
8: 01:28:34.485570 802.1Q vlan#20 P0 192.168.20.253.23 > 192.168.30.253.58887:
9: 01:28:34.485662 802.1Q vlan#20 P0 192.168.20.253.23 > 192.168.30.253.58887:
```

Copy capture using ASDM

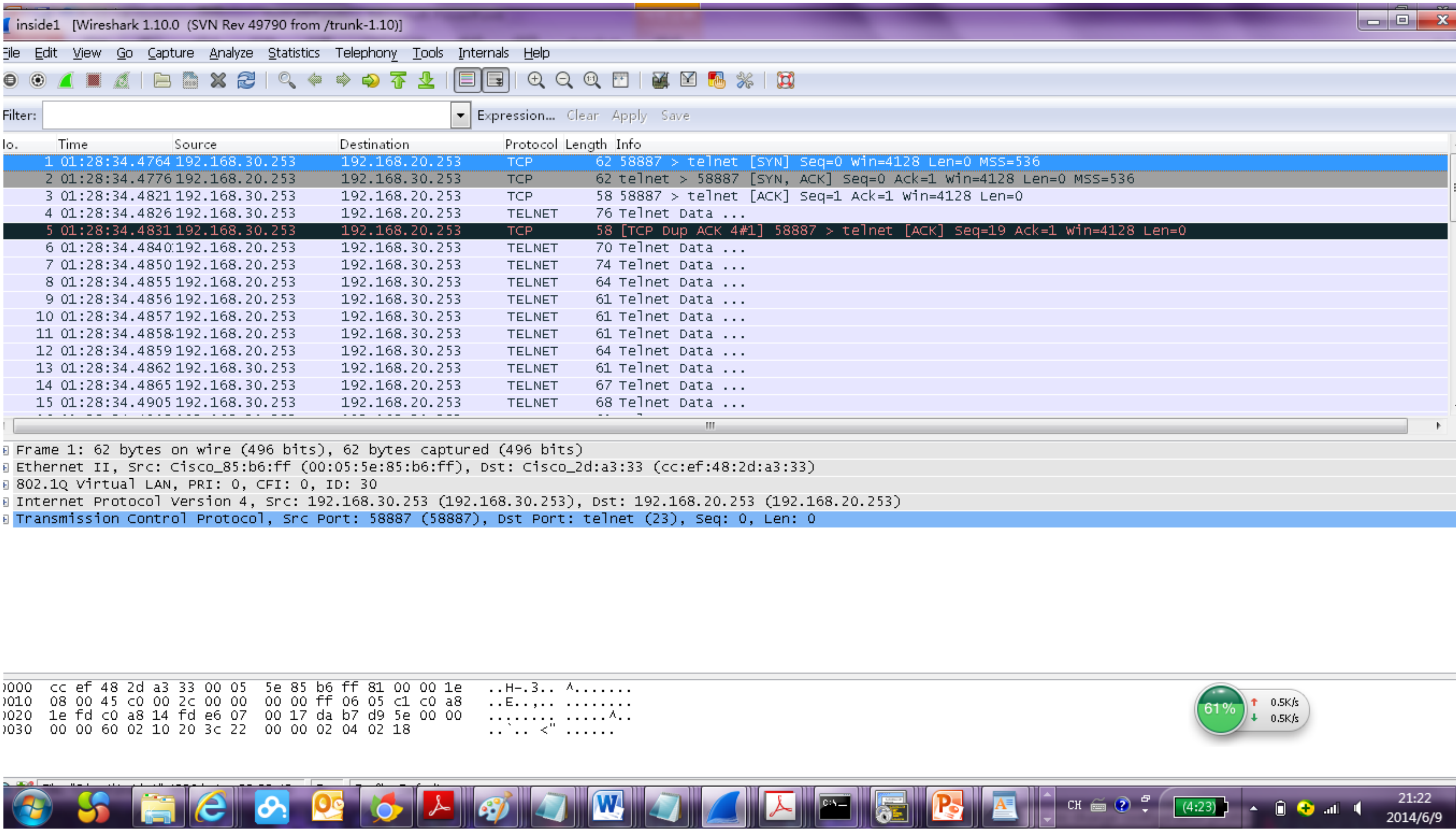


Save Captures

ASCII [✓ PCAP]

Save the buffer in the specified format to the localhost.

Copy capture using ASDM



inside1 [Wireshark 1.10.0 (SVN Rev 49790 from /trunk-1.10)]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save

```
No. Time            Source          Destination     Protocol Length Info
  1 01:28:34.4764   192.168.30.253  192.168.20.253  TCP      62 58887 > telnet [SYN] Seq=0 Win=4128 Len=0 MSS=536
  2 01:28:34.4776   192.168.20.253  192.168.30.253  TCP      62 telnet > 58887 [SYN, ACK] Seq=0 Ack=1 Win=4128 Len=0 MSS=536
  3 01:28:34.4821   192.168.30.253  192.168.20.253  TCP      58 58887 > telnet [ACK] Seq=1 Ack=1 Win=4128 Len=0
  4 01:28:34.4826   192.168.30.253  192.168.20.253  TELNET   76 Telnet Data ...
  5 01:28:34.4831   192.168.30.253  192.168.20.253  TCP      58 [TCP Dup ACK 4#1] 58887 > telnet [ACK] Seq=19 Ack=1 Win=4128 Len=0
  6 01:28:34.4840   192.168.20.253  192.168.30.253  TELNET   70 Telnet Data ...
  7 01:28:34.4850   192.168.20.253  192.168.30.253  TELNET   74 Telnet Data ...
  8 01:28:34.4855   192.168.20.253  192.168.30.253  TELNET   64 Telnet Data ...
  9 01:28:34.4856   192.168.20.253  192.168.30.253  TELNET   61 Telnet Data ...
 10 01:28:34.4857   192.168.20.253  192.168.30.253  TELNET   61 Telnet Data ...
 11 01:28:34.4858   192.168.20.253  192.168.30.253  TELNET   61 Telnet Data ...
 12 01:28:34.4859   192.168.20.253  192.168.30.253  TELNET   64 Telnet Data ...
 13 01:28:34.4862   192.168.30.253  192.168.20.253  TELNET   61 Telnet Data ...
 14 01:28:34.4865   192.168.30.253  192.168.20.253  TELNET   67 Telnet Data ...
 15 01:28:34.4905   192.168.30.253  192.168.20.253  TELNET   68 Telnet Data ...
```

```
Frame 1: 62 bytes on wire (496 bits), 62 bytes captured (496 bits)
Ethernet II, Src: Cisco_85:b6:ff (00:05:5e:85:b6:ff), Dst: Cisco_2d:a3:33 (cc:ef:48:2d:a3:33)
802.1Q Virtual LAN, PRI: 0, CFI: 0, ID: 30
Internet Protocol Version 4, Src: 192.168.30.253 (192.168.30.253), Dst: 192.168.20.253 (192.168.20.253)
Transmission Control Protocol, Src Port: 58887 (58887), Dst Port: telnet (23), Seq: 0, Len: 0
```

```
0000  cc ef 48 2d a3 33 00 05  5e 85 b6 ff 81 00 00 1e   ..H-.3.. ^.......
0010  08 00 45 c0 00 2c 00 00  00 00 ff 06 05 c1 c0 a8   ..E..,.. ........
0020  1e fd c0 a8 14 fd e6 07  00 17 da b7 d9 5e 00 00   ........ .....^..
0030  00 00 60 02 10 20 3c 22  00 00 02 04 02 18         ..`.. <" ......
```

Copy capture using CLI

- access-list inside1 permit tcp host 192.168.30.253 host 192.168.20.253 eq telnet

  access-list inside1 permit tcp host 192.168.20.253 eq telnet host   192.168.30.253

  access-list dmz permit tcp host 4.4.4.4 host 192.168.20.253 eq   telnet

  ccess-list dmz permit tcp host 4.4.4.4 eq telnet host 192.168.30.253

- capture inside1 type raw-data access-list inside1 interface inside1

  capture dmz type raw-data access-list dmz interface dmz
- Initiate actual traffic
- ciscoasa(config)# copy /pcap capture:inside1 tftp:

  Source capture name [inside1]?

  Address or name of remote host []?

Packet trace

```
ciscoasa(config)# packet-tracer input inside1 tcp 192.168.30.253 1025 192.168.20..253 23
detailed

Phase: 1

Type: ACCESS-LIST

Subtype:

Result: ALLOW

Config:

Implicit Rule

Additional Information:

 Forward Flow based lookup yields rule:

 in  id=0x33bca6e0, priority=1, domain=permit, deny=false

        hits=773, user_data=0x0, cs_id=0x0, l3_type=0x8

        src mac=0000.0000.0000, mask=0000.0000.0000

        dst mac=0000.0000.0000, mask=0100.0000.0000
```

Packet trace (continued)

```
Phase: 2

Type: ROUTE-LOOKUP

Subtype: input

Result: ALLOW

Config:

Additional Information:

in   192.168.20.0    255.255.255.0   dmz
```

Packet trace (continued)

```
Phase: 3
Type: ACCESS-LIST
Subtype:
Result: DROP
Config:
Implicit Rule
Additional Information:
 Forward Flow based lookup yields rule:
 in  id=0x33c054e0, priority=11, domain=permit, deny=true
        hits=1, user_data=0x5, cs_id=0x0, flags=0x0, protocol=0
        src ip=0.0.0.0, mask=0.0.0.0, port=0
        dst ip=0.0.0.0, mask=0.0.0.0, port=0, dscp=0x0
```

Packet trace (continued)

```
Result:
input-interface: inside1
input-status: up
input-line-status: up
output-interface: dmz
output-status: up
output-line-status: up
Action: drop
Drop-reason: (acl-drop) Flow is denied by configured rule
```

Cases for xlate issue

ASA version is 8.2x, customer complained the policy static nat never takes effect if the static nat has been configured , both statements configured with same source host.

Lab creating result:

1 When static one-to-one nat and policy nat are written at the same time , but in different order, for example, static one to one nat is configured first, then:

```
static (inside,outside) 10.75.61.176 192.168.30.254 netmask 255.255.255.255
static (inside,outside) 10.75.61.177  access-list  cs
```

output from asa:

```
I%ASA-6-302020: Built outbound ICMP connection for faddr 10.75.61.199/0 gaddr
10.75.61.176/8661 laddr 192.168.30.254/8661
%ASA-6-302020: Built inbound ICMP connection for faddr 10.75.61.199/0 gaddr
10.75.61.176/8661 laddr 192.168.30.254/8661
%ASA-6-302020: Built outbound ICMP connection for faddr 10.75.61.199/0 gaddr
10.75.61.176/8662 laddr 192.168.30.254/8662
```

Cases for xlate issue (continued)

2 When policy nat is configured first , then:

```
static (inside,outside) 10.75.61.177  access-list cs

static (inside,outside) 10.75.61.176 192.168.30.254 netmask 255.255.255.255
```

ASA output as follows:

```
%ASA-6-302020: Built outbound ICMP connection for faddr 10.75.61.199/0 gaddr

10.75.61.177/7356 laddr 192.168.30.254/7356

%ASA-6-302020: Built outbound ICMP connection for faddr 10.75.61.199/0 gaddr

10.75.61.177/7357 laddr 192.168.30.254/7357

%ASA-6-302020: Built inbound ICMP connection for faddr 10.75.61.199/0 gaddr

10.75.61.177/7357 laddr 192.168.30.254/7357
```

3 The nat order is determined by the configuration order when policy nat and static nat is configured at the same time.

Cases for xlate issue (continued)

- In asa 9.x, nat can’t work for 172.0.1.0 and 172.0.2.0 subnets even if the right configuration statement is written.

```
object network YP1_subnet_to_internet
   subnet 172.0.1.0 255.255.255.0
   nat (inside,outside) dynamic 220.250.64.124
object network  YP2_subnet_to_internet
    subnet 172.0.2.0 255.255.255.0
    nat (inside,outside) dynamic 220.250.64.124
```

Troubleshooting:

Show nat from customer’s operation log:

Cases for xlate issue (continued)

- ASASM/YP# sh nat Auto NAT Policies (Section 2)

```
43 (inside) to (outside) source static YPVpn-obj MyVpn-NoNat_Subnet-obj  translate_hits = 4404, untranslate_hits =
0

 44 (inside) to (outside) source dynamic YP1_subnet_to_internet 220.250.64.124 translate_hits = 0, untranslate_hits = 0

 45 (inside) to (outside) source dynamic YP2_subnet_to_internet 220.250.64.124 translate_hits = 0, untranslate_hits = 0
```

Check customer's configuration:

```
 object network YPVpn-obj

    subnet 172.0.2.0 255.255.255.0

object network MyVpn-NoNat_Subnet-obj

    subnet 172.0.2.0 255.255.255.0

object network YPVpn-obj

    nat (inside,outside) static MyVpn-NoNat_Subnet-obj
```

Cases for xlate issue (continued)

Solution:

remove the config from the configuration:

```
object network YPVpn-obj
   nat (inside,outside) static MyVpn-NoNat_Subnet-obj
```

Performing Zero Downtime Upgrades for Failover Pairs

The two units in a failover configuration should have the same major (first number) and minor (second number) software version. However, you do not need to maintain version parity on the units during the upgrade process; you can have different versions on the software running on each unit and still maintain failover support. To ensure long-term compatibility and stability, we recommend upgrading both units to the same version as soon as possible.

Performing Zero Downtime Upgrades for Failover Pairs

- **Maintenance Release**

You can upgrade from any maintenance release to any other maintenance release within a minor release.

For example, you can upgrade from 7.0(1) to 7.0(4) without first installing the maintenance releases in between.

- **Minor Release**

You can upgrade from a minor release to the next minor release. You cannot skip a minor release.

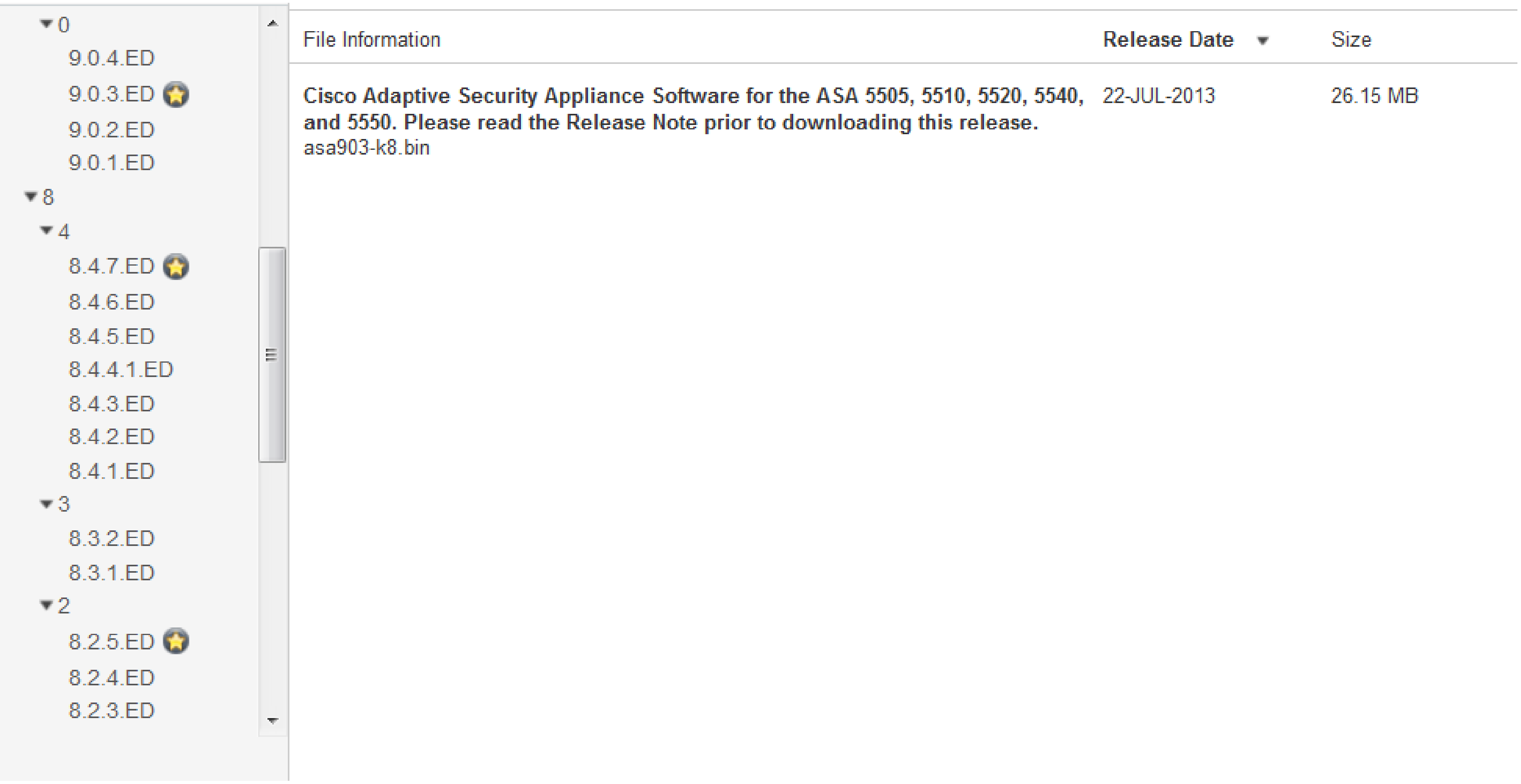
For example, you can upgrade from 7.0 to 7.1. Upgrading from 7.0 directly to 7.2 is not supported for zero-downtime upgrades; you must first upgrade to 7.1.

- **Major Release**

You can upgrade from the last minor release of the previous version to the next major release.

- For example, you can upgrade from 7.9 to 8.0, assuming that 7.9 is the last minor version in the 7.x release.

Performing Zero Downtime Upgrades for Failover Pairs (continue)



- 0
  - 9.0.4.ED
  - 9.0.3.ED
  - 9.0.2.ED
  - 9.0.1.ED
- 8
  - 4
    - 8.4.7.ED
    - 8.4.6.ED
    - 8.4.5.ED
    - 8.4.4.1.ED
    - 8.4.3.ED
    - 8.4.2.ED
    - 8.4.1.ED
  - 3
    - 8.3.2.ED
    - 8.3.1.ED
  - 2
    - 8.2.5.ED
    - 8.2.4.ED
    - 8.2.3.ED

File Information	Release Date	Size
Cisco Adaptive Security Appliance Software for the ASA 5505, 5510, 5520, 5540, and 5550. Please read the Release Note prior to downloading this release. asa903-k8.bin	22-JUL-2013	26.15 MB

Performing Zero Downtime Upgrades for Failover Pairs (continue)

- Pls refer to url below to upgrade asa failover pair:

http://www.cisco.com/c/en/us/td/docs/security/asa/asa92/configuration/general/asa-general-cli/admin-swconfig.html#34831

Thank you.