ACCESS CONTROL LIST

Access Control List

- An Access Control List (ACL) is a series of IOS commands that are used to filter packets based on information found in the packet header.
  - By default, a router does not have any ACLs configured.
  - When an ACL is applied to an interface, the router performs the additional task of evaluating all network packets as they pass through the interface to determine if the packet can be forwarded.
- An ACL uses a sequential list of permit or deny statements, known as access control entries (ACEs).
  - **Note:** ACEs are also commonly called ACL statements.
- When network traffic passes through an interface configured with an ACL, the router compares the information within the packet against each ACE, in sequential order, to determine if the packet matches one of the ACEs.
  - This process is called **packet filtering**.

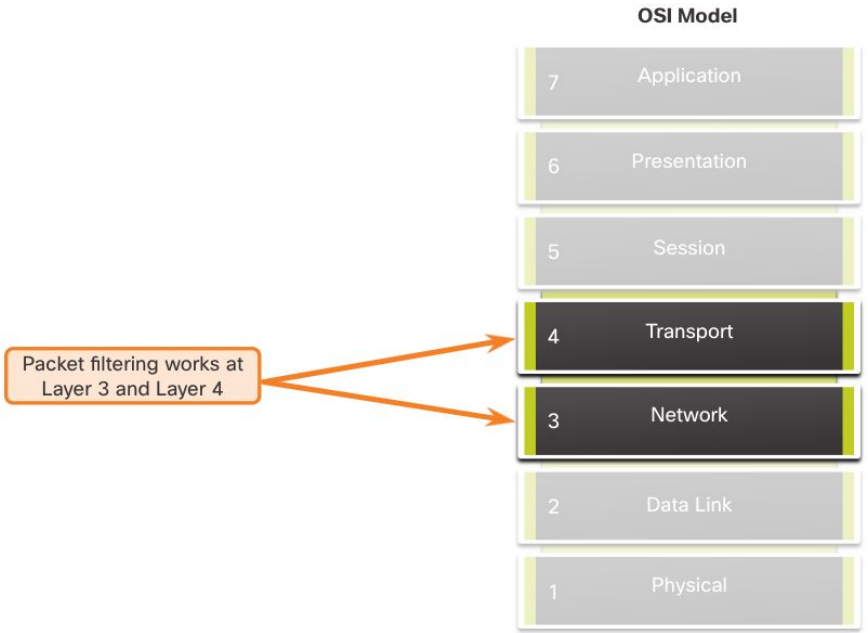
Access Control List

Several tasks performed by routers require the use of ACLs to identify traffic:

- Limit network traffic in order to increase network performance
- Provide traffic flow control
- Provide a basic level of security for network access
- Filter traffic based on traffic type
- Screen hosts to permit or deny access to network services
- Provide priority to certain classes of network traffic

Access Control List: Packet Filtering

- Packet filtering controls access to a network by analyzing the incoming and/or outgoing packets and forwarding them or discarding them based on given criteria.
- Packet filtering can occur at Layer 3 (Network) or Layer 4 (Transport).
- Cisco routers support two types of ACLs:
  - **Standard ACLs** - ACLs only filter at Layer 3 using the source IPv4 address only.
  - **Extended ACLs** - ACLs filter at:
    - Layer 3 using the source and / or destination IPv4 address.
    - Layer 4 using TCP, UDP ports, and optional protocol type information for finer control.



Access Control List: Operation

- ACLs define the set of rules that give added control for packets that enter inbound interfaces, packets that relay through the router, and packets that exit outbound interfaces of the router.
- ACLs can be configured to apply to inbound traffic and outbound traffic.
  - **Note**: ACLs do not act on packets that originate from the router itself.
- An inbound ACL filters packets before they are routed to the outbound interface.
  - An inbound ACL is efficient because it saves the overhead of routing lookups if the packet is discarded.
- An outbound ACL filters packets after being routed, regardless of the inbound interface.

Access Control List: Operation

- When an ACL is applied to an interface, it follows a specific operating procedure. Here are the operational steps used when traffic has entered a router interface with an inbound standard IPv4 ACL configured:
  1. The router extracts the source IPv4 address from the packet header.
  2. The router starts at the top of the ACL and compares the source IPv4 address to each ACE in a sequential order.
  3. When a match is made, the router carries out the instruction, either permitting or denying the packet, and the remaining ACEs in the ACL, if any, are not analyzed.
  4. If the source IPv4 address does not match any ACEs in the ACL, the packet is discarded because there is an implicit deny ACE automatically applied to all ACLs.
- The last ACE statement of an ACL is always an implicit deny that blocks all traffic. It is hidden and not displayed in the configuration.
  - **Note**: An ACL must have at least one permit statement otherwise all traffic will be denied due to the implicit deny ACE statement.

Access Control List: Wildcard Mask

- A wildcard mask is similar to a subnet mask in that it uses the ANDing process to identify which bits in an IPv4 address to match. Unlike a subnet mask, in which binary 1 is equal to a match and binary 0 is not a match, in a wildcard mask, the reverse is true.
  - An IPv4 ACE uses a 32-bit wildcard mask to determine which bits of the address to examine for a match.

Wildcard Mask	Last Octet (Binary)	Meaning (0 - match, 1 - ignore)
0.0.0.0	**00000000**	• Match all octets.
0.0.0.63	**00111111**	• Match the first three octets • Match the two left most bits of the last octet • Ignore the last 6 bits
0.0.0.248	**11111100**	• Match the first three octets • Ignore the six left most bits of the last octet • Match the last two bits
0.0.0.255	**11111111**	• Match the first three octet • Ignore the last octet

Access Control List: Wildcard Mask Overview

Wildcard to Match a Host:

- Assume ACL 10 needs an ACE that only permits the host with IPv4 address 192.168.1.1. Recall that "0" equals a match and "1" equals ignore. To match a specific host IPv4 address, a wildcard mask consisting of all zeroes (i.e., 0.0.0.0) is required.
- When the ACE is processed, the wildcard mask will permit only the 192.168.1.1 address. The resulting ACE in ACL 10 would be **access-list 10 permit 192.168.1.1 0.0.0.0.**

	Decimal	Binary
IPv4 address	192.168.1.1	11000000.10101000.00000001.00000001
Wildcard Mask	0.0.0.0	00000000.00000000.00000000.00000000
Permitted IPv4 Address	**192.168.1.1**	11000000.10101000.00000001.00000001

Access Control List: Wildcard Mask

Wildcard Mask to Match an IPv4 Subnet

- ACL 10 needs an ACE that permits all hosts in the 192.168.1.0/24 network. The wildcard mask 0.0.0.255 stipulates that the very first three octets must match exactly but the fourth octet does not.
- When processed, the wildcard mask 0.0.0.255 permits all hosts in the 192.168.1.0/24 network. The resulting ACE in ACL 10 would be **access-list 10 permit 192.168.1.0 0.0.0.255.**

	Decimal	Binary
IPv4 address	192.168.1.1	11000000.10101000.00000001.00000001
Wildcard Mask	0.0.0.255	00000000.00000000.00000000.11111111
Permitted IPv4 Address	**192.168.1.0/24**	11000000.10101000.00000001.00000000

Access Control List: Wildcard Mask

Wildcard Mask to Match an IPv4 Address Range

- ACL 10 needs an ACE that permits all hosts in the 192.168.16.0/24, 192.168.17.0/24, …, 192.168.31.0/24 networks.
- When processed, the wildcard mask 0.0.15.255 permits all hosts in the 192.168.16.0/24 to 192.168.31.0/24 networks. The resulting ACE in ACL 10 would be **access-list 10 permit 192.168.16.0 0.0.15.255.**

	Decimal	Binary
IPv4 address	192.168.16.0	11000000.10101000.00010000.00000000
Wildcard Mask	0.0.15.255	00000000.00000000.00001111.11111111
Permitted IPv4 Address	**192.168.16.0/24 to 192.168.31.0/24**	11000000.10101000.00010000.00000000 11000000.10101000.00011111.00000000

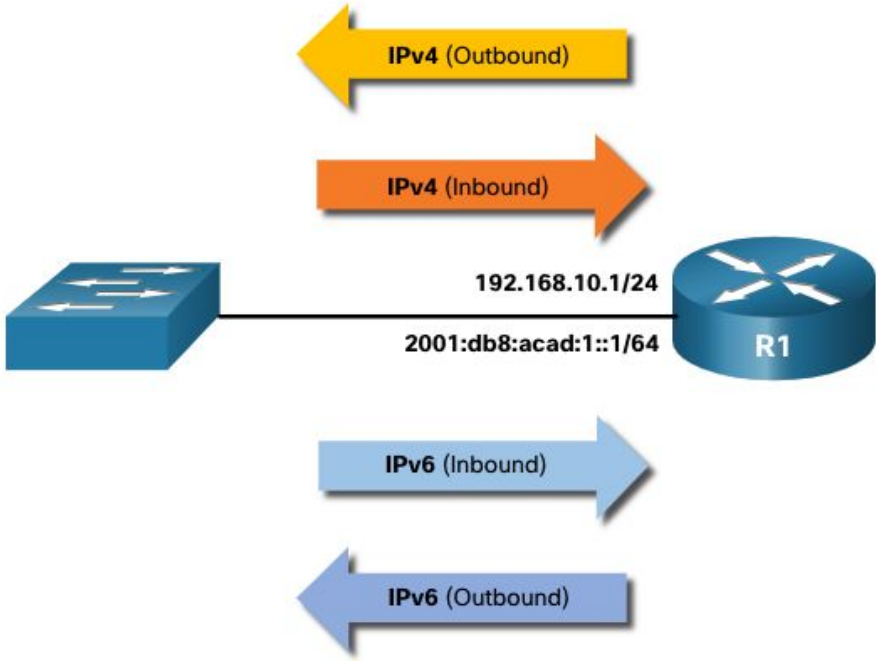
Access Control List: Wildcard Mask

- A shortcut method to calculate wildcard mask is to subtract the subnet mask from 255.255.255.255. Some examples:
  - Assume you wanted an ACE in ACL 10 to permit access to all users in the 192.168.3.0/24 network.
  - To calculate the wildcard mask, subtract the subnet mask (255.255.255.0) from 255.255.255.255.
  - This produces the wildcard mask 0.0.0.255.
  - The ACE would be **access-list 10 permit 192.168.3.0 0.0.0.255.**
- The Cisco IOS provides two keywords to identify the most common uses of wildcard masking. The two keywords are:
  - **host** - This keyword substitutes for the 0.0.0.0 mask. This mask states that all IPv4 address bits must match to filter just one host address.
  - **any** - This keyword substitutes for the 255.255.255.255 mask. This mask says to ignore the entire IPv4 address or to accept any addresses.

Access Control List: Number of ACLs per Interface

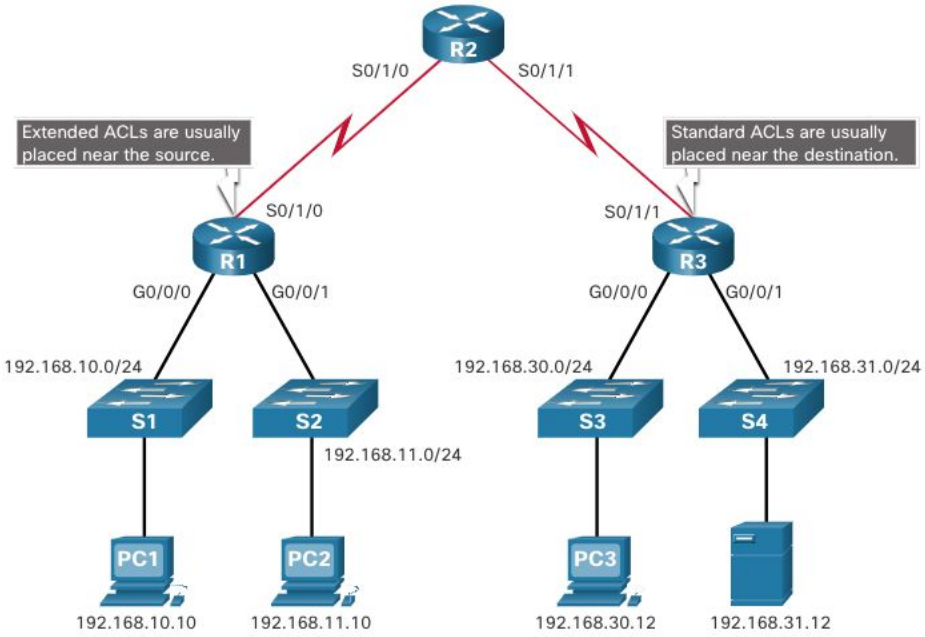
- There is a limit on the number of ACLs that can be applied on a router interface. For example, a dual-stacked (i.e, IPv4 and IPv6) router interface can have up to four ACLs applied, as shown in the figure.
- Specifically, a router interface can have:
  - One outbound IPv4 ACL.
  - One inbound IPv4 ACL.
  - One inbound IPv6 ACL.
  - One outbound IPv6 ACL.

- **Note**: ACLs do not have to be configured in both directions. The number of ACLs and their direction applied to the interface will depend on the security policy of the organization.



Access Control List: Types

- There are two types of IPv4 ACLs:
  - **Standard ACLs** - These permit or deny packets based only on the source IPv4 address.
  - **Extended ACLs** - These permit or deny packets based on the source IPv4 address and destination IPv4 address, protocol type, source and destination TCP or UDP ports and more.
- Every ACL should be placed where it has the greatest impact on efficiency.
- Extended ACLs should be located as close as possible to the source of the traffic to be filtered.
- Standard ACLs should be located as close to the destination as possible.



Access Control List: Types

Numbered ACLs

- ACLs numbered 1-99, or 1300-1999 are standard ACLs, while ACLs numbered 100-199, or 2000-2699 are extended ACLs.

```
R1(config)# access-list ?
 <1-99> IP standard access list
 <100-199> IP extended access list
 <1100-1199> Extended 48-bit MAC address access list
 <1300-1999> IP standard access list (expanded range)
 <200-299> Protocol type-code access list
 <2000-2699> IP extended access list (expanded range)
 <700-799> 48-bit MAC address access list
 rate-limit Simple rate-limit specific access list
 template Enable IP template acls
Router(config)# access-list
```

Access Control List: Types

Named ACLs

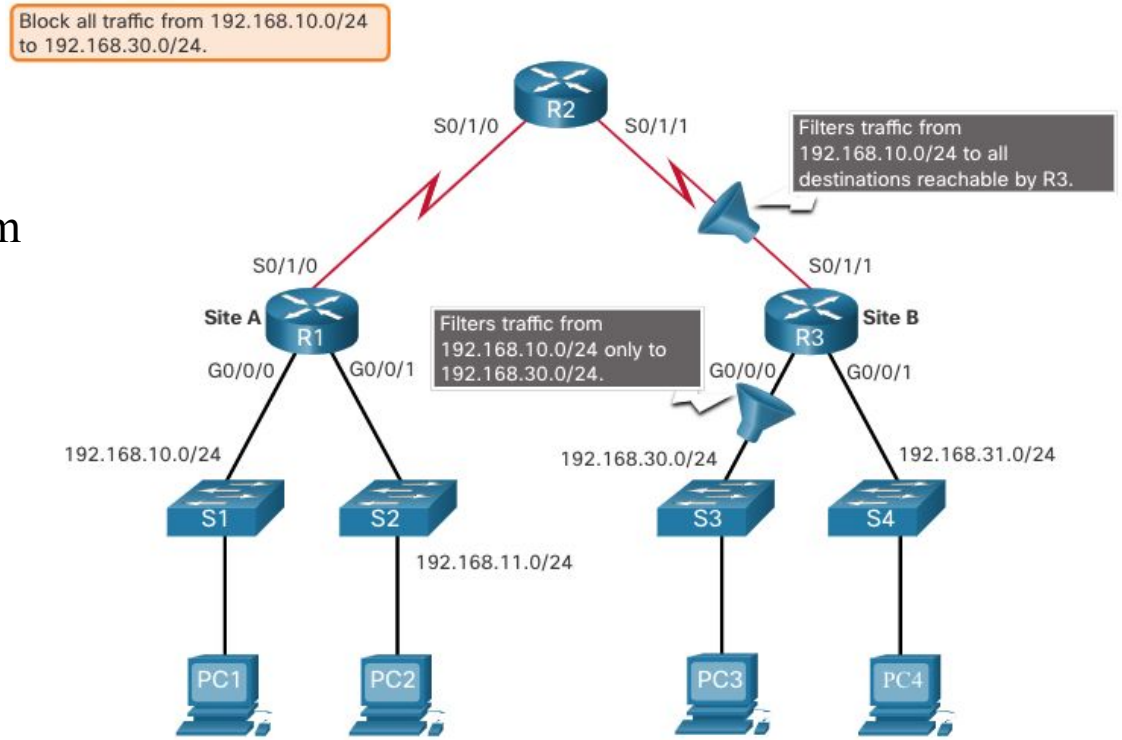
- Named ACLs are the preferred method to use when configuring ACLs. Specifically, standard and extended ACLs can be named to provide information about the purpose of the ACL. For example, naming an extended ACL FTP-FILTER is far better than having a numbered ACL 100.
- The **ip access-list** global configuration command is used to create a named ACL, as shown in the following example.

```
R1(config)# ip access-list extended FTP-FILTER
R1(config-ext-nacl)# permit tcp 192.168.10.0 0.0.0.255 any eq ftp
R1(config-ext-nacl)# permit tcp 192.168.10.0 0.0.0.255 any eq ftp-data
R1(config-ext-nacl)#
```

Access Control List: Types

In the figure, the administrator wants to prevent traffic originating in the 192.168.10.0/24 network from reaching the 192.168.30.0/24 network.

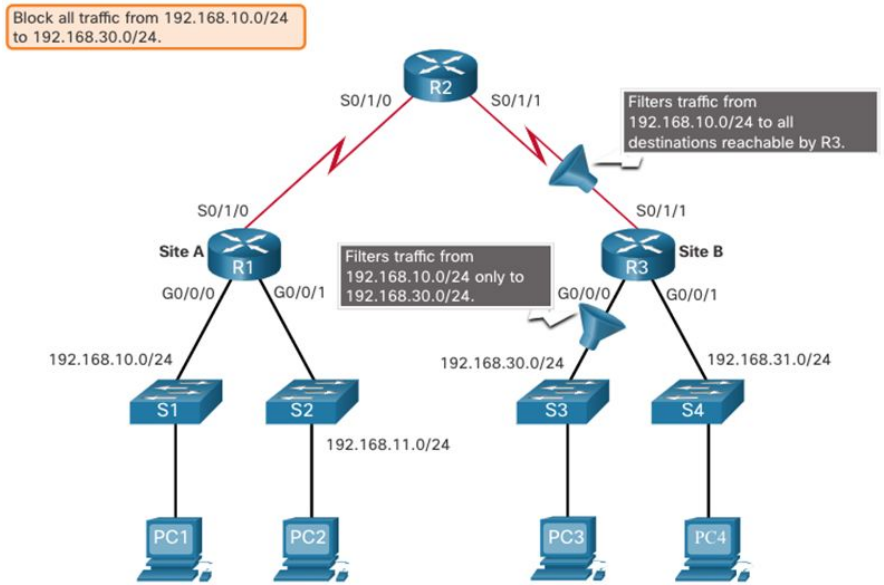
Following the basic placement guidelines, the administrator would place a standard ACL on router R3.



Access Control List: Types

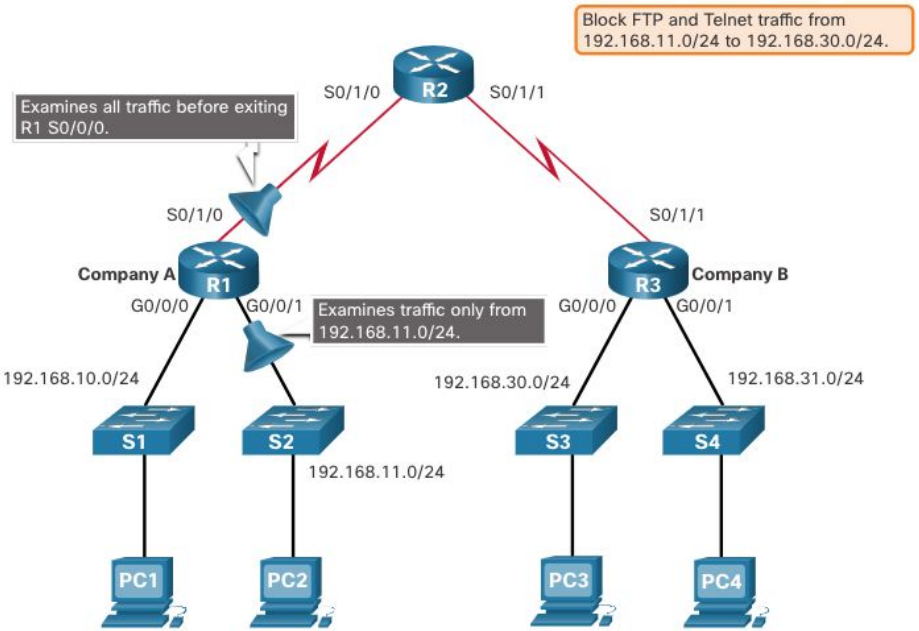
There are two possible interfaces on R3 to apply the standard ACL:

- **R3 S0/1/1 interface (inbound)** - The standard ACL can be applied inbound on the R3 S0/1/1 interface to deny traffic from .10 network. However, it would also filter .10 traffic to the 192.168.31.0/24 (.31 in this example) network. Therefore, the standard ACL should not be applied to this interface.
- **R3 G0/0 interface (outbound)** - The standard ACL can be applied outbound on the R3 G0/0/0 interface. This will not affect other networks that are reachable by R3. Packets from .10 network will still be able to reach the .31 network. This is the best interface to place the standard ACL to meet the traffic requirements.



Access Control List: Types

- Extended ACLs should be located as close to the source as possible.
- However, the organization can only place ACLs on devices that they control. Therefore, the extended ACL placement must be determined in the context of where organizational control extends.
- In the figure, for example, Company A wants to deny Telnet and FTP traffic to Company B's 192.168.30.0/24 network from their 192.168.11.0/24 network, while permitting all other traffic.



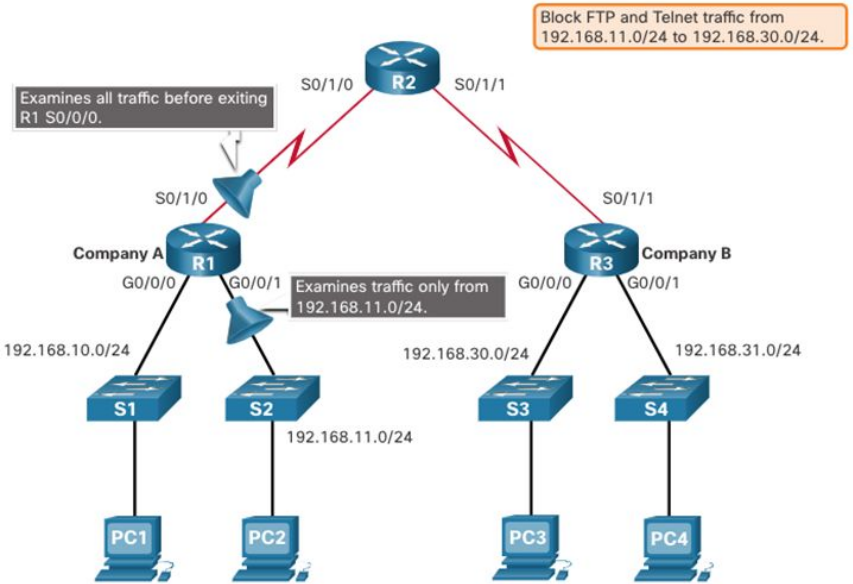
Access Control List: Types

An extended ACL on R3 would accomplish the task, but the administrator does not control R3. In addition, this solution allows unwanted traffic to cross the entire network, only to be blocked at the destination.

The solution is to place an extended ACL on R1 that specifies both source and destination addresses.

There are two possible interfaces on R1 to apply the extended ACL:

- **R1 S0/1/0 interface (outbound)** - The extended ACL can be applied outbound on the S0/1/0 interface. This solution will process all packets leaving R1 including packets from 192.168.10.0/24.
- **R1 G0/0/1 interface (inbound)** - The extended ACL can be applied inbound on the G0/0/1 and only packets from the 192.168.11.0/24 network are subject to ACL processing on R1. Because the filter is to be limited to only those packets leaving the 192.168.11.0/24 network, applying the extended ACL to G0/1 is the best solution.



Configure Standard ACL

To create a numbered standard ACL, use the **access-list** command.

```
Router(config)# access-list access-list-number {deny | permit | remark text} source [source-wildcard]
[log]
```

Parameter	Description
access-list-number	Number range is 1 to 99 or 1300 to 1999
deny	Denies access if the condition is matched
permit	Permits access if the condition is matched
remark *text*	(Optional) text entry for documentation purposes
source	Identifies the source network or host address to filter
source-wildcard	(Optional) 32-bit wildcard mask that is applied to the source
log	(Optional) Generates and sends an informational message when the ACE is matched

Note: Use the **no access-list** *access-list-number* global configuration command to remove a numbered standard ACL.

Configure Standard ACL

To create a named standard ACL, use the **ip access-list standard** command.

- ACL names are alphanumeric, case sensitive, and must be unique.
- Capitalizing ACL names is not required but makes them stand out when viewing the running-config output.

```
Router(config)# ip access-list standard access-list-name
```

```
R1(config)# ip access-list standard NO-ACCESS
R1(config-std-nacl)# ?
Standard Access List configuration commands:
  <1-2147483647>  Sequence Number
  default         Set a command to its defaults
  deny            Specify packets to reject
  exit            Exit from access-list configuration mode
  no              Negate a command or set its defaults
  permit          Specify packets to forward
  remark          Access list entry comment
R1(config-std-nacl)#
```

Configure Standard ACL

After a standard IPv4 ACL is configured, it must be linked to an interface or feature.

- The **ip access-group** command is used to bind a numbered or named standard IPv4 ACL to an interface.
- To remove an ACL from an interface, first enter the **no ip access-group** interface configuration command.

```
Router(config-if) # ip access-group {access-list-number | access-list-name} {in | out}
```

Configure Standard ACL

The example ACL permits traffic from host 192.168.10.10 and all hosts on the 192.168.20.0/24 network out interface serial 0/1/0 on router R1.

```
R1(config)# access-list 10 remark ACE permits ONLY host 192.168.10.10 to the internet
R1(config)# access-list 10 permit host 192.168.10.10
R1(config)# do show access-lists
Standard IP access list 10
    10 permit 192.168.10.10
R1(config)#
```

```
R1(config)# access-list 10 remark ACE permits all host in LAN 2
R1(config)# access-list 10 permit 192.168.20.0 0.0.0.255
R1(config)# do show access-lists
Standard IP access list 10
    10 permit 192.168.10.10
    20 permit 192.168.20.0, wildcard bits 0.0.0.255
R1(config)#
```

```
R1(config)# interface Serial 0/1/0
R1(config-if)# ip access-group 10 out
R1(config-if)# end
R1#
```

Configure Standard ACL

The example ACL permits traffic from host 192.168.10.10 and all hosts on the 192.168.20.0/24 network out interface serial 0/1/0 on router R1.

```
R1(config)# no access-list 10
R1(config)# ip access-list standard PERMIT-ACCESS
R1(config-std-nacl)# remark ACE permits host 192.168.10.10
R1(config-std-nacl)# permit host 192.168.10.10
R1(config-std-nacl)#
```

```
R1(config-std-nacl)# remark ACE permits host 192.168.10.10
R1(config-std-nacl)# permit host 192.168.10.10
R1(config-std-nacl)# remark ACE permits all hosts in LAN 2
R1(config-std-nacl)# permit 192.168.20.0 0.0.0.255
R1(config-std-nacl)# exit
R1(config)#
```

```
R1(config)# interface Serial 0/1/0
R1(config-if)# ip access-group PERMIT-ACCESS out
R1(config-if)# end
R1#
```

Modify ACL

- After an ACL is configured, it may need to be modified. ACLs with multiple ACEs can be complex to configure. Sometimes the configured ACE does not yield the expected behaviors.
- There are two methods to use when modifying an ACL:
  - Use a text editor.
  - Use sequence numbers.

```
R1# show run | section access-list
access-list 1 deny   19.168.10.10
access-list 1 permit 192.168.10.0 0.0.0.255
R1#
```

- To correct an error in an ACL:
  - Copy the ACL from the running configuration and paste it into the text editor.
  - Make the necessary edits or changes.
  - Remove the previously configured ACL on the router.
  - Copy and paste the edited ACL back to the router.

```
R1(config)# no access-list 1
R1(config)#
R1(config)# access-list 1 deny 192.168.10.10
R1(config)# access-list 1 permit 192.168.10.0 0.0.0.255
R1(config)#
```

Modify ACL

An ACL ACE can be deleted or added using the ACL sequence numbers.

- Use the **ip access-list standard** command to edit an ACL.
- Statements cannot be overwritten using an existing sequence number. The current statement must be deleted first with the **no 10** command. Then the correct ACE can be added using sequence number.

```
R1# show access-lists
Standard IP access list 1
    10 deny   19.168.10.10
    20 permit 192.168.10.0, wildcard bits 0.0.0.255
R1#
```

```
R1# conf t
R1(config)# ip access-list standard 1
R1(config-std-nacl)# no 10
R1(config-std-nacl)# 10 deny host 192.168.10.10
R1(config-std-nacl)# end
R1# show access-lists
Standard IP access list 1
    10 deny   192.168.10.10
    20 permit 192.168.10.0, wildcard bits 0.0.0.255
R1#
```

Extended ACLs

- Extended ACLs provide a greater degree of control. They can filter on source address, destination address, protocol (i.e., IP, TCP, UDP, ICMP), and port number.
- Extended ACLs can be created as:
  - **Numbered Extended ACL** - Created using the **access-list** *access-list-number* global configuration command.
  - **Named Extended ACL** - Created using the **ip access-list extended** *access-list-name*.

```
R1(config)# access-list 100 permit ?
  <0-255>        An IP protocol number
  ahp            Authentication Header Protocol
  dvmrp          dvmrp
  eigrp          Cisco's EIGRP routing protocol
  esp            Encapsulation Security Payload
  gre            Cisco's GRE tunneling
  icmp           Internet Control Message Protocol
  igmp           Internet Gateway Message Protocol
  ip             Any Internet Protocol
  ipinip         IP in IP tunneling
  nos            KA9Q NOS compatible IP over IP tunneling
  object-group   Service object group
  ospf           OSPF routing protocol
  pcp            Payload Compression Protocol
  pim            Protocol Independent Multicast
  tcp            Transmission Control Protocol
  udp            User Datagram Protocol
R1(config)# access-list 100 permit
```

Extended ACLs

- Extended ACLs can filter on different port number and port name options.
- This example configures an extended ACL 100 to filter HTTP traffic. The first ACE uses the **www** port name. The second ACE uses the port number **80**. Both ACEs achieve exactly the same result.

```
R1(config)# access-list 100 permit tcp any any eq www
!or...
R1(config)# access-list 100 permit tcp any any eq 80
```

- Configuring the port number is required when there is not a specific protocol name listed such as SSH (port number 22) or an HTTPS (port number 443), as shown in the next example.

```
R1(config)# access-list 100 permit tcp any any eq 22
R1(config)# access-list 100 permit tcp any any eq 443
R1(config)#
```

Extended ACLs

- In this example, the ACL permits both HTTP and HTTPS traffic from the 192.168.10.0 network to go to any destination.

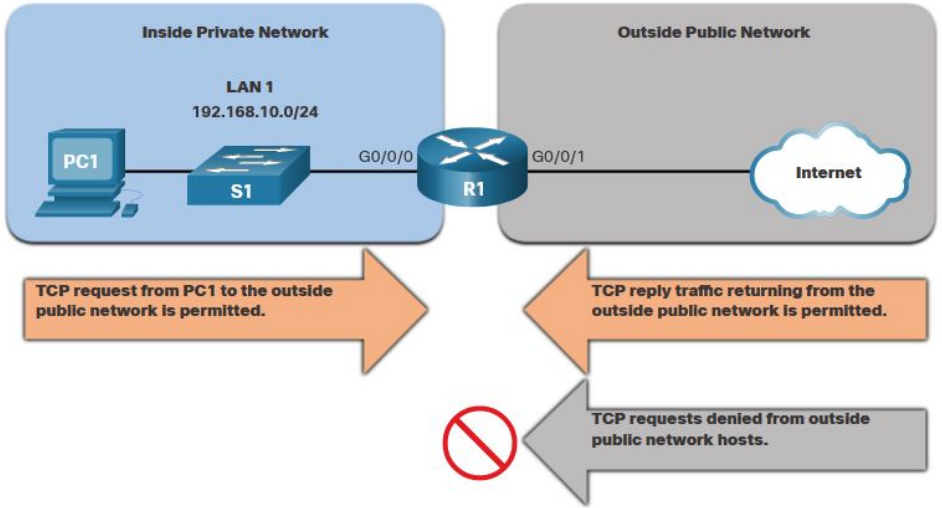
- Extended ACLs can be applied in various locations. However, they are commonly applied close to the source. Here ACL 110 is applied inbound on the R1 G0/0/0 interface.

```
R1(config)# access-list 110 permit tcp 192.168.10.0 0.0.0.255 any eq www
R1(config)# access-list 110 permit tcp 192.168.10.0 0.0.0.255 any eq 443
R1(config)# interface g0/0/0
R1(config-if)# ip access-group 110 in
R1(config-if)# exit
R1(config)#
```

Extended ACLs

TCP can also perform basic stateful firewall services using the TCP **established** keyword.

- The **established** keyword enables inside traffic to exit the inside private network and permits the returning reply traffic to enter the inside private network.
- TCP traffic generated by an outside host and attempting to communicate with an inside host is denied.



Extended ACLs

- ACL 120 is configured to only permit returning web traffic to the inside hosts. The ACL is then applied outbound on the R1 G0/0/0 interface.
- The **show access-lists** command shows that inside hosts are accessing the secure web resources from the internet.

Note: A match occurs if the returning TCP segment has the ACK or reset (RST) flag bits set, indicating that the packet belongs to an existing connection.

```
R1(config)# access-list 120 permit tcp any 192.168.10.0 0.0.0.255 established
R1(config)# interface g0/0/0
R1(config-if)# ip access-group 120 out
R1(config-if)# end
R1# show access-lists
Extended IP access list 110
    10 permit tcp 192.168.10.0 0.0.0.255 any eq www
    20 permit tcp 192.168.10.0 0.0.0.255 any eq 443 (657 matches)
Extended IP access list 120
    10 permit tcp any 192.168.10.0 0.0.0.255 established (1166 matches)
R1#
```

Extended ACLs

Naming an ACL makes it easier to understand its function. To create a named extended ACL, use the **ip access-list extended** configuration command.

In the example, a named extended ACL called NO-FTP-ACCESS is created and the prompt changed to named extended ACL configuration mode. ACE statements are entered in the named extended ACL sub configuration mode.

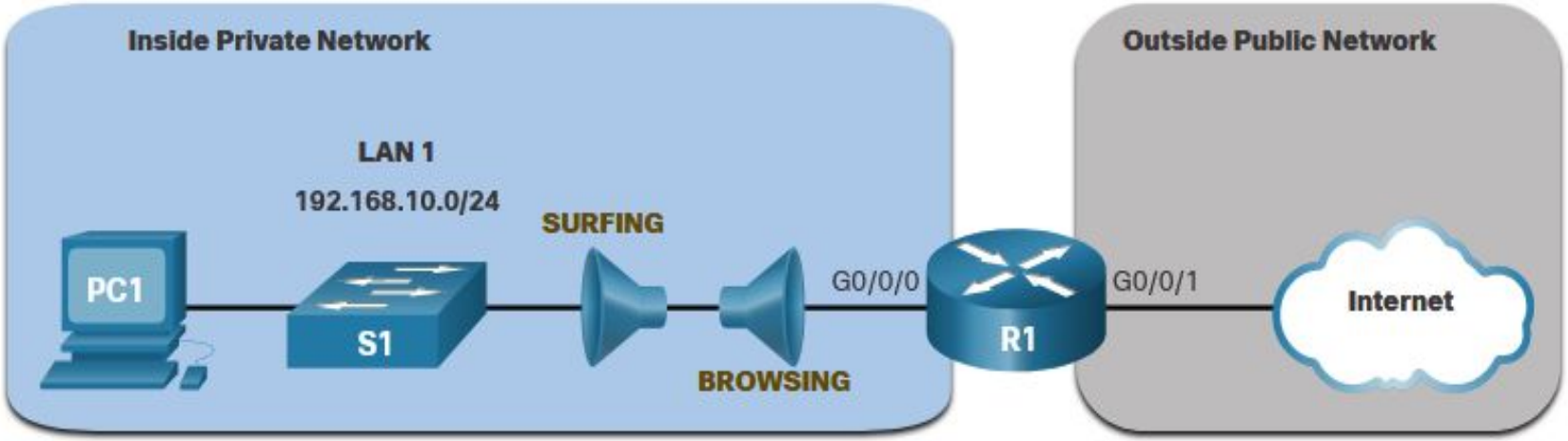
```
Router(config)# ip access-list extended access-list-name
```

```
R1(config)# ip access-list extended NO-FTP-ACCESS
R1(config-ext-nacl)#
```

Extended ACLs

The topology below is used to demonstrate configuring and applying two named extended IPv4 ACLs to an interface:

- **SURFING** - This will permit inside HTTP and HTTPS traffic to exit to the internet.
- **BROWSING** - This will only permit returning web traffic to the inside hosts while all other traffic exiting the R1 G0/0/0 interface is implicitly denied.



Extended ACLs

- The SURFING ACL permits HTTP and HTTPS traffic from inside users to exit the G0/0/1 interface connected to the internet. Web traffic returning from the internet is permitted back into the inside private network by the BROWSING ACL.
- The SURFING ACL is applied inbound and the BROWSING ACL is applied outbound on the R1 G0/0/0 interface.

```
R1(config)# ip access-list extended SURFING
R1(config-ext-nacl)# Remark Permits inside HTTP and HTTPS traffic
R1(config-ext-nacl)# permit tcp 192.168.10.0 0.0.0.255 any eq 80
R1(config-ext-nacl)# permit tcp 192.168.10.0 0.0.0.255 any eq 443
R1(config-ext-nacl)# exit
R1(config)#
R1(config)# ip access-list extended BROWSING
R1(config-ext-nacl)# Remark Only permit returning HTTP and HTTPS traffic
R1(config-ext-nacl)# permit tcp any 192.168.10.0 0.0.0.255 established
R1(config-ext-nacl)# exit
R1(config)# interface g0/0/0
R1(config-if)# ip access-group SURFING in
R1(config-if)# ip access-group BROWSING out
R1(config-if)# end
R1# show access-lists
Extended IP access list SURFING
    10 permit tcp 192.168.10.0 0.0.0.255 any eq www
    20 permit tcp 192.168.10.0 0.0.0.255 any eq 443 (124 matches)
Extended IP access list BROWSING
    10 permit tcp any 192.168.10.0 0.0.0.255 established (369 matches)
R1#
```

Extended ACLs

The show access-lists command is used to verify the ACL statistics. Notice that the permit secure HTTPS counters (i.e., eq 443) in the SURFING ACL and the return established counters in the BROWSING ACL have increased.

```
R1# show access-lists
Extended IP access list BROWSING
    10 permit tcp any 192.168.10.0 0.0.0.255 established
Extended IP access list SURFING
    10 permit tcp 19.168.10.0 0.0.0.255 any eq www
    20 permit tcp 192.168.10.0 0.0.0.255 any eq 443
R1#
```

Extended ACLs

- An extended ACL can be edited using a text editor when many changes are required. Or, if the edit applies to one or two ACEs, then sequence numbers can be used.
- Example:
  - The ACE sequence number 10 in the SURFING ACL has an incorrect source IP networks address.

```
R1# show access-lists
Extended IP access list BROWSING
    10 permit tcp any 192.168.10.0 0.0.0.255 established
Extended IP access list SURFING
    10 permit tcp 19.168.10.0 0.0.0.255 any eq www
    20 permit tcp 192.168.10.0 0.0.0.255 any eq 443
R1#
```

Extended ACLs

- To correct this error the original statement is removed with the **no** *sequence_#* command and the corrected statement is added replacing the original statement.
- The **show access-lists** command output verifies the configuration change.

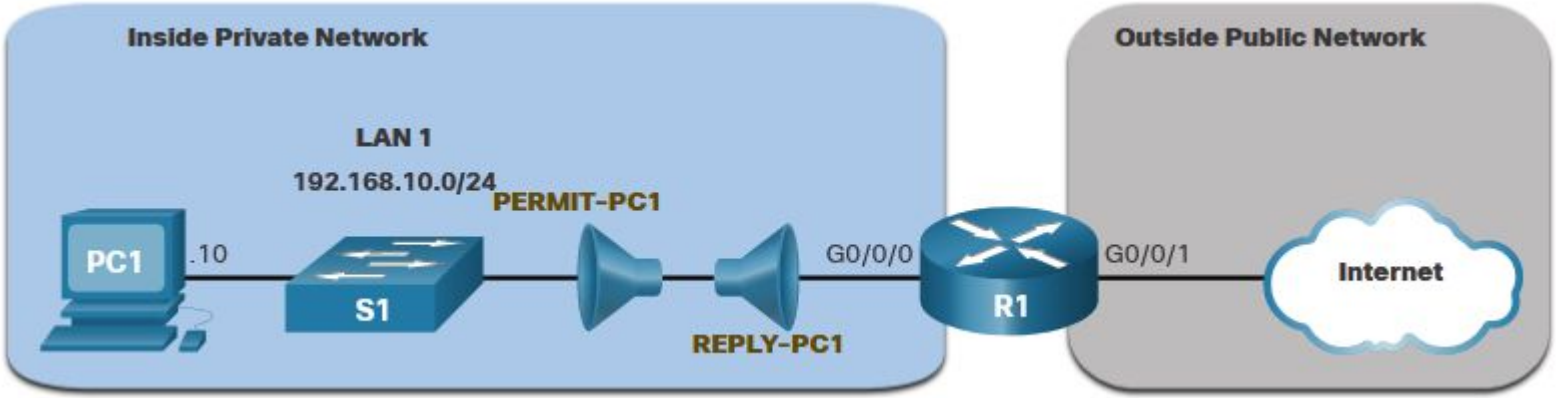
```
R1# configure terminal
R1(config)# ip access-list extended SURFING
R1(config-ext-nacl)# no 10
R1(config-ext-nacl)# 10 permit tcp 192.168.10.0 0.0.0.255 any eq www
R1(config-ext-nacl)# end
```

```
R1# show access-lists
Extended IP access list BROWSING
    10 permit tcp any 192.168.10.0 0.0.0.255 established
Extended IP access list SURFING
    10 permit tcp 192.168.10.0 0.0.0.255 any eq www
    20 permit tcp 192.168.10.0 0.0.0.255 any eq 443
R1#
```

Extended ACLs

Two named extended ACLs will be created:

- **PERMIT-PC1** - This will only permit PC1 TCP access to the internet and deny all other hosts in the private network.
- **REPLY-PC1** - This will only permit specified returning TCP traffic to PC1 implicitly deny all other traffic.



Extended ACLs

- The **PERMIT-PC1** ACL permits PC1 (192.168.10.10) TCP access to the FTP, SSH, Telnet, DNS , HTTP, and HTTPS traffic.
- The **REPLY-PC1** ACL will permit return traffic to PC1.
- The **PERMIT-PC1** ACL is applied inbound and the **REPLY-PC1** ACL applied outbound on the R1 G0/0/0 interface.

```
R1(config)# ip access-list extended PERMIT-PC1
R1(config-ext-nacl)# Remark Permit PC1 TCP access to internet
R1(config-ext-nacl)# permit tcp host 192.168.10.10 any eq 20
R1(config-ext-nacl)# permit tcp host 192.168.10.10 any eq 21
R1(config-ext-nacl)# permit tcp host 192.168.10.10 any eq 22
R1(config-ext-nacl)# permit tcp host 192.168.10.10 any eq 23
R1(config-ext-nacl)# permit tcp host 192.168.10.10 any eq 53
R1(config-ext-nacl)# permit tcp host 192.168.10.10 any eq 80
R1(config-ext-nacl)# permit tcp host 192.168.10.10 any eq 443
R1(config-ext-nacl)# deny ip 192.168.10.0 0.0.0.255 any
R1(config-ext-nacl)# exit
R1(config)#
R1(config)# ip access-list extended REPLY-PC1
R1(config-ext-nacl)# Remark Only permit returning traffic to PC1
R1(config-ext-nacl)# permit tcp any host 192.168.10.10 established
R1(config-ext-nacl)# exit
R1(config)# interface g0/0/0
R1(config-if)# ip access-group PERMIT-PC1 in
R1(config-if)# ip access-group REPLY-PC1 out
R1(config-if)# end
R1#
```

Extended ACLs

The **show ip interface** command is used to verify the ACL on the interface and the direction in which it was applied.

```
R1# show ip interface g0/0/0
GigabitEthernet0/0/0 is up, line protocol is up (connected)
  Internet address is 192.168.10.1/24
  Broadcast address is 255.255.255.255
  Address determined by setup command
  MTU is 1500 bytes
  Helper address is not set
  Directed broadcast forwarding is disabled
  Outgoing access list is REPLY-PC1
  Inbound  access list is PERMIT-PC1
  Proxy ARP is enabled
  Security level is default
  Split horizon is enabled
  ICMP redirects are always sent
  ICMP unreachables are always sent
  ICMP mask replies are never sent
  IP fast switching is disabled
  IP fast switching on the same interface is disabled
  IP Flow switching is disabled
  IP Fast switching turbo vector
  IP multicast fast switching is disabled
  IP multicast distributed fast switching is disabled
  Router Discovery is disabled
R1#
R1# show ip interface g0/0/0 | include access list
Outgoing access list is REPLY-PC1
Inbound access list is PERMIT-PC1
R1#
```

Extended ACLs

The **show access-lists** command can be used to confirm that the ACLs work as expected. The command displays statistic counters that increase whenever an ACE is matched.

Note: Traffic must be generated to verify the operation of the ACL.

```
R1# show access-lists
Extended IP access list PERMIT-PC1
10 permit tcp host 192.168.10.10 any eq 20
20 permit tcp host 192.168.10.10 any eq ftp
30 permit tcp host 192.168.10.10 any eq 22
40 permit tcp host 192.168.10.10 any eq telnet
50 permit tcp host 192.168.10.10 any eq domain
60 permit tcp host 192.168.10.10 any eq www
70 permit tcp host 192.168.10.10 any eq 443
80 deny ip 192.168.10.0 0.0.0.255 any
Extended IP access list REPLY-PC1
10 permit tcp any host 192.168.10.10 established
R1#
```

Extended ACLs

The **show running-config** command can be used to validate what was configured. The command also displays configured remarks.

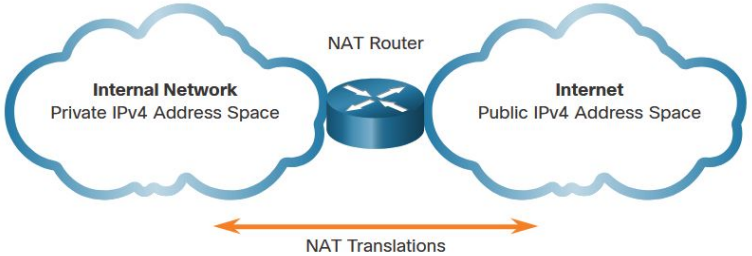
```
R1# show running-config | begin ip access-list
ip access-list extended PERMIT-PC1
remark Permit PC1 TCP access to internet
permit tcp host 192.168.10.10 any eq 20
permit tcp host 192.168.10.10 any eq ftp
permit tcp host 192.168.10.10 any eq 22
permit tcp host 192.168.10.10 any eq telnet
permit tcp host 192.168.10.10 any eq domain
permit tcp host 192.168.10.10 any eq www
permit tcp host 192.168.10.10 any eq 443
deny ip 192.168.10.0 0.0.0.255 any
ip access-list extended REPLY-PC1
remark Only permit returning traffic to PC1
permit tcp any host 192.168.10.10 established
!
```

NAT

NAT Characteristics

- Networks are commonly implemented using private IPv4 addresses, as defined in RFC 1918.
- Private IPv4 addresses cannot be routed over the internet and are used within an organization or site to allow devices to communicate locally.
- To allow a device with a private IPv4 address to access devices and resources outside of the local network, the private address must first be translated to a public address.
- NAT provides the translation of private addresses to public addresses.

Class	Activity Type	Activity Name
A	10.0.0.0 – 10.255.255.255	10.0.0.0/8
B	172.16.0.0 – 172.31.255.255	172.16.0.0/12
C	192.168.0.0 – 192.168.255.255	192.168.0.0/16



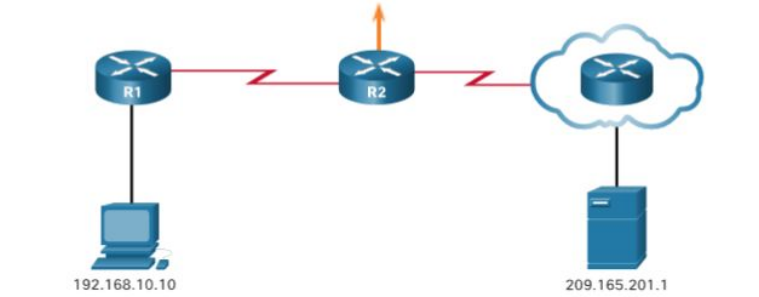
NAT Characteristics

PC1 wants to communicate with an outside web server with public address 209.165.201.1.

1. PC1 sends a packet addressed to the web server.
2. R2 receives the packet and reads the source IPv4 address to determine if it needs translation.
3. R2 adds mapping of the local to global address to the NAT table.
4. R2 sends the packet with the translated source address toward the destination.
5. The web server responds with a packet addressed to the inside global address of PC1 (209.165.200.226).
6. R2 receives the packet with destination address 209.165.200.226. R2 checks the NAT table and finds an entry for this mapping. R2 uses this information and translates the inside global address (209.165.200.226) to the inside local address (192.168.10.10), and the packet is forwarded toward PC1.

NAT Table

Inside Local	Inside Global	Outside Local	Outside Global
192.168.10.10	209.165.200.226	209.165.201.1	209.165.201.1



NAT Characteristics

Inside local address

The address of the source as seen from inside the network. This is typically a private IPv4 address. The inside local address of PC1 is 192.168.10.10.

Inside global addresses

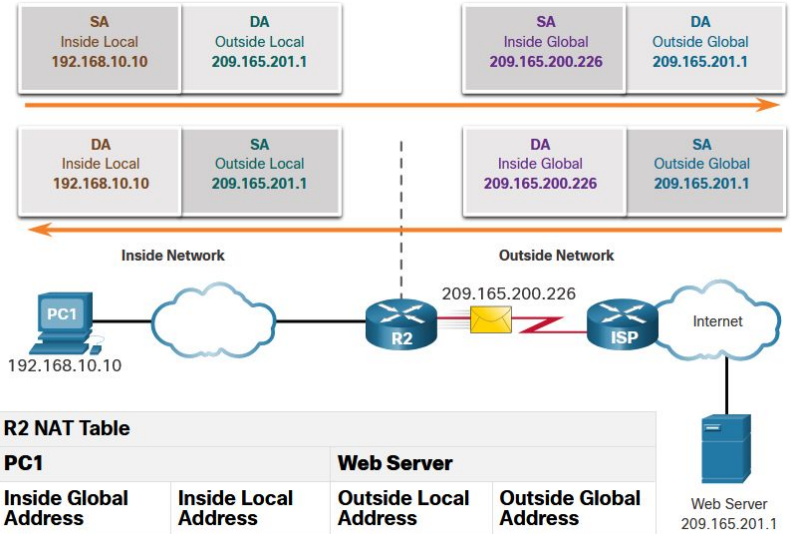
The address of source as seen from the outside network. The inside global address of PC1 is 209.165.200.226

Outside global address

The address of the destination as seen from the outside network. The outside global address of the web server is 209.165.201.1

Outside local address

The address of the destination as seen from the inside network. PC1 sends traffic to the web server at the IPv4 address 209.165.201.1. While uncommon, this address could be different than the globally routable address of the destination.



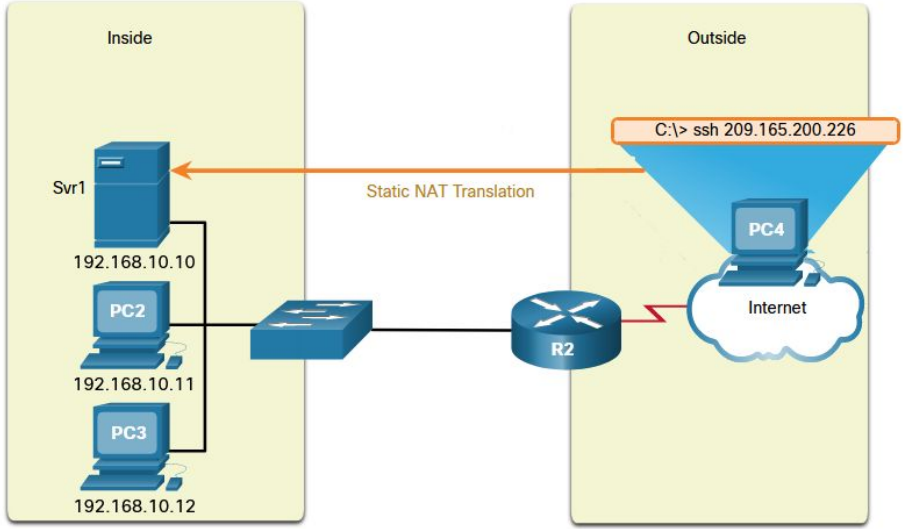
R2 NAT Table			
PC1		**Web Server**	
Inside Global Address	**Inside Local Address**	**Outside Local Address**	**Outside Global Address**
209.165.200.226	**192.168.10.10**	**209.165.201.1**	**209.165.201.1**

NAT: Types

Static NAT uses a one-to-one mapping of local and global addresses configured by the network administrator that remain constant.

- Static NAT is useful for web servers or devices that must have a consistent address that is accessible from the internet, such as a company web server.
- It is also useful for devices that must be accessible by authorized personnel when offsite, but not by the general public on the internet.

Note: Static NAT requires that enough public addresses are available to satisfy the total number of simultaneous user sessions.



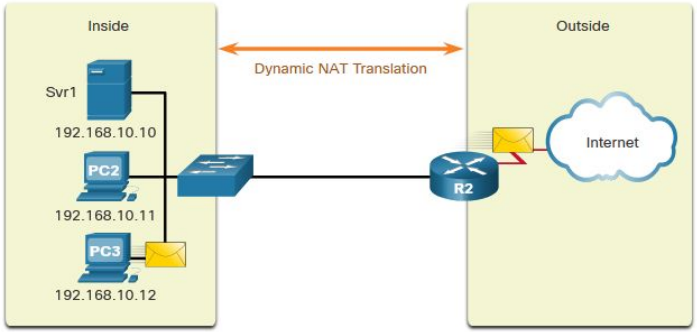
Static NAT Table

Inside Local Address	Inside Global Address - Addresses reachable via R2
192.168.10.10	209.165.200.226
192.168.10.11	209.165.200.227
192.168.10.12	209.165.200.228

NAT: Types

Dynamic NAT uses a pool of public addresses and assigns them on a first-come, first-served basis.

- When an inside device requests access to an outside network, dynamic NAT assigns an available public IPv4 address from the pool.
- The other addresses in the pool are still available for use.



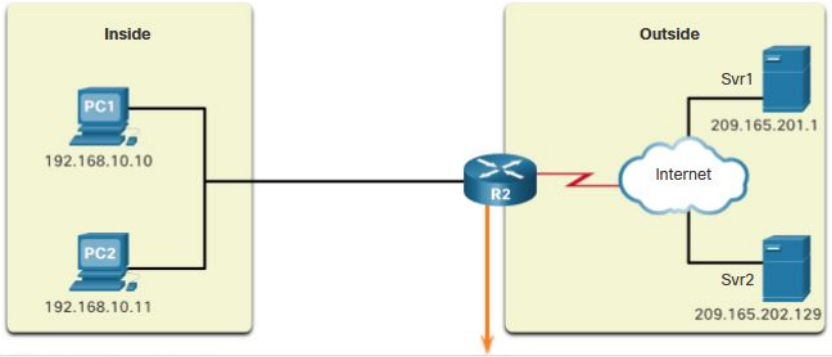
IPv4 NAT Pool	
Inside Local Address	**Inside Global Address Pool - Addresses reachable via R2**
192.168.10.12	209.165.200.226
Available	209.165.200.227
Available	209.165.200.228
Available	209.165.200.229
Available	209.165.200.230

Note: Dynamic NAT requires that enough public addresses are available to satisfy the total number of simultaneous user sessions.

NAT: Types

Port Address Translation (PAT), also known as NAT **overload**, maps multiple private IPv4 addresses to a single public IPv4 address or a few addresses.

- With PAT, when the NAT router receives a packet from the client, it uses the source port number to uniquely identify the specific NAT translation.
- PAT ensures that devices use a different TCP port number for each session with a server on the internet.



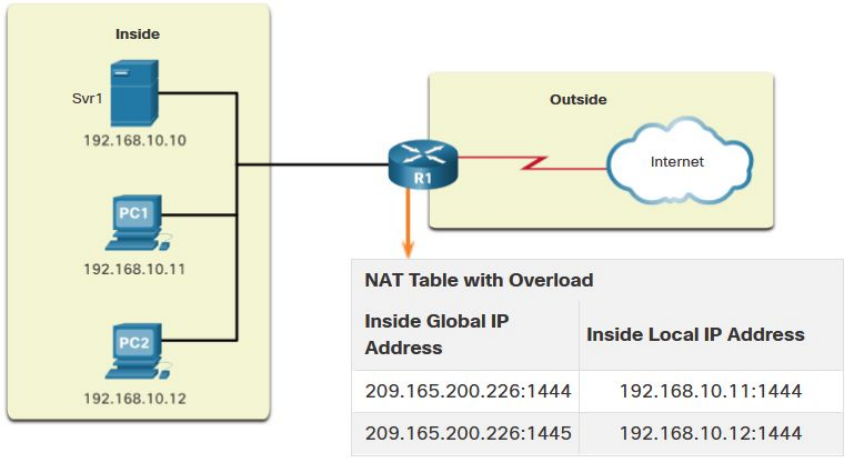
NAT Table with Overload

Inside Local IP Address	Inside Global IP Address	Outside Local IP Address	Outside Global IP Address
192.168.10.10:1555	209.165.200.226:1555	209.165.201.1:80	209.165.201.1:80
192.168.10.11:1331	209.165.200.226:1331	209.165.202.129:80	209.165.202.129:80

NAT: Types

PAT attempts to preserve the original source port. If the original source port is already used, PAT assigns the first available port number starting from the beginning of the appropriate port group 0-511, 512-1,023, or 1,024-65,535.

- When there are no more ports available and there is more than one external address in the address pool, PAT moves to the next address to try to allocate the original source port.
- The process continues until there are no more available ports or external IPv4 addresses in the address pool.



NAT: Types

Summary of the differences between NAT and PAT.

NAT - Only modifies the IPv4 addresses

Inside Global Address	Inside Local Address
209.165.200.226	192.168.10.10

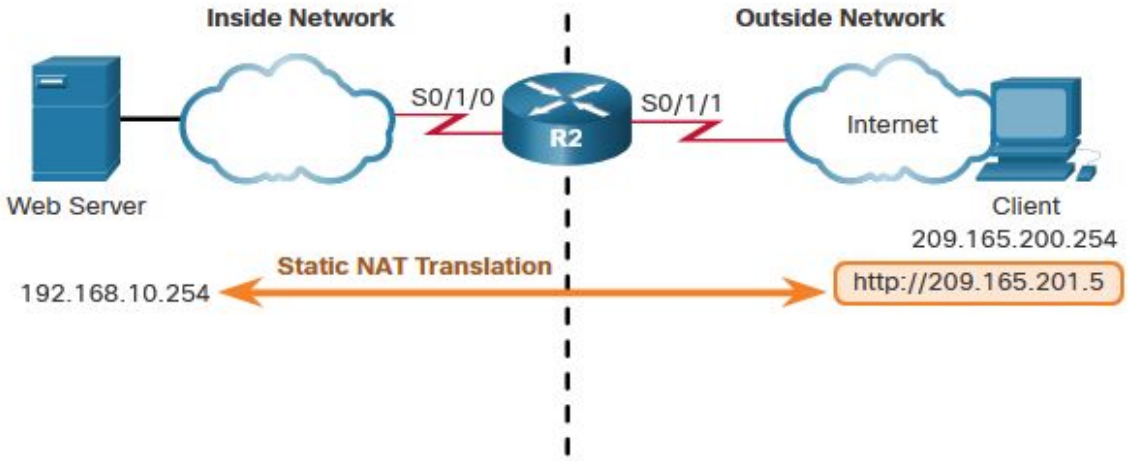
PAT - PAT modifies both the IPv4 address and the port number.

Inside Global Address	Inside Local Address
209.165.200.226:2031	192.168.10.10:2031

NAT	PAT
One-to-one mapping between Inside Local and Inside Global addresses.	One Inside Global address can be mapped to many Inside Local addresses.
Uses only IPv4 addresses in translation process.	Uses IPv4 addresses and TCP or UDP source port numbers in translation process.
A unique Inside Global address is required for each inside host accessing the outside network.	A single unique Inside Global address can be shared by many inside hosts accessing the outside network.

Static NAT

- Static NAT is a **one-to-one mapping** between an inside address and an outside address.
- Static NAT **allows external devices to initiate connections** to internal devices using the statically assigned public address.
- For instance, an internal web server may be mapped to a specific inside global address so that it is accessible from outside networks.

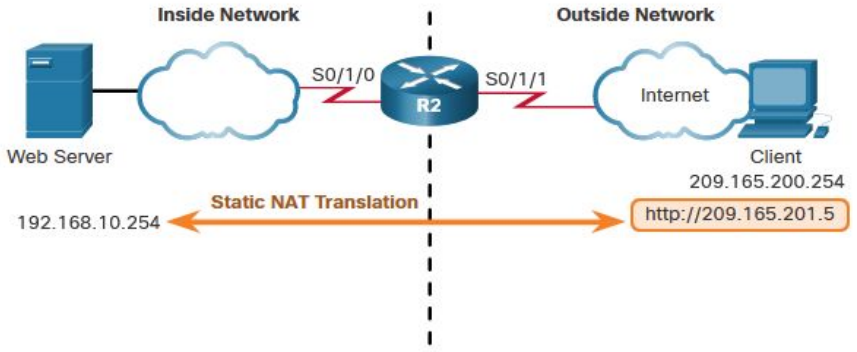


Static NAT

There are two basic tasks when configuring static NAT translations:

- **Step 1** - Create a mapping between the inside local address and the inside global addresses using the **ip nat inside source static** command.
- **Step 2** - The interfaces participating in the translation are configured as inside or outside relative to NAT with the **ip nat inside** and **ip nat outside** commands.

```
R2(config)# ip nat inside source static 192.168.10.254 209.165.201.5
R2(config)#
R2(config)# interface serial 0/1/0
R2(config-if)# ip address 192.168.1.2 255.255.255.252
R2(config-if)# ip nat inside
R2(config-if)# exit
R2(config)# interface serial 0/1/1
R2(config-if)# ip address 209.165.200.1 255.255.255.252
R2(config-if)# ip nat outside
```



Static NAT

To verify NAT operation, issue the **show ip nat translations** command.

- This command shows active NAT translations.
- Because the example is a static NAT configuration, the translation is always present in the NAT table regardless of any active communications.
- If the command is issued during an active session, the output also indicates the address of the outside device.

```
R2# show ip nat translations
Pro  Inside global       Inside local       Outside local      Outside global
---  209.165.201.5       192.168.10.254     ---                ---
Total number of translations: 1
```

```
R2# show ip nat translations
Pro  Inside global       Inside local       Outside local        Outside global
tcp  209.165.201.5       192.168.10.254     209.165.200.254      209.165.200.254
---  209.165.201.5       192.168.10.254       ---                  ---
Total number of translations: 2
```

Static NAT

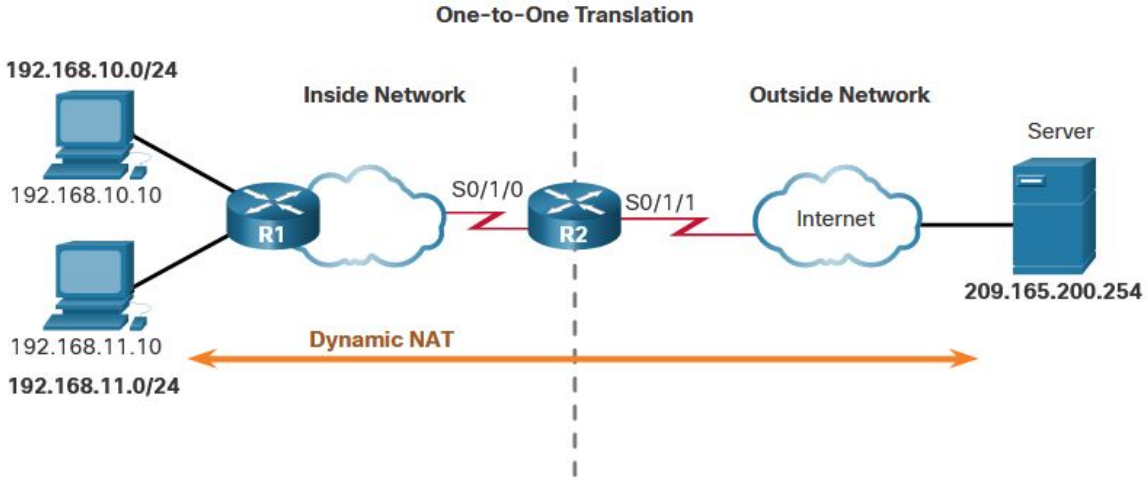
Another useful command is **show ip nat statistics.**

- It displays information about the total number of active translations, NAT configuration parameters, the number of addresses in the pool, and the number of addresses that have been allocated.
- To verify that the NAT translation is working, it is best to clear statistics from any past translations using the **clear ip nat statistics** command before testing.

```
R2# show ip nat statistics
Total active translations: 1 (1 static, 0 dynamic; 0 extended)
Outside interfaces:
  Serial0/1/1
Inside interfaces:
  Serial0/1/0
Hits: 4  Misses: 1
(output omitted)
```

Dynamic NAT

- Dynamic NAT automatically maps inside local addresses to inside global addresses.
- Dynamic NAT uses a pool of inside global addresses.
- The pool of inside global addresses is available to any device on the inside network on a first-come first-served basis.
- If all addresses in the pool are in use, a device must wait for an available address before it can access the outside network.

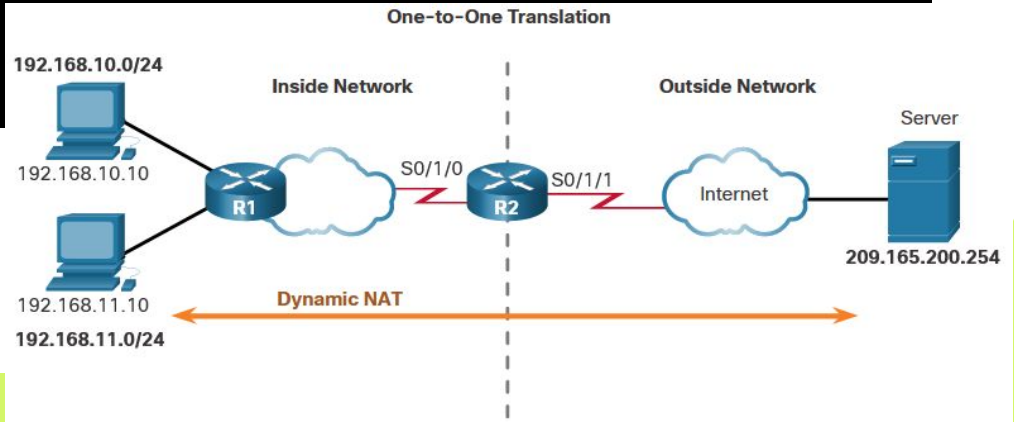


Dynamic NAT

There are five tasks when configuring dynamic NAT translations:

- **Step 1** - Define the pool of addresses that will be used for translation using the **ip nat pool** command.
- **Step 2** - Configure a standard ACL to identify (permit) only those addresses that are to be translated.
- **Step 3** - Bind the ACL to the pool, using the **ip nat inside source list** command.
- **Step 4** - Identify which interfaces are inside.
- **Step 5** - Identify which interfaces are outside.

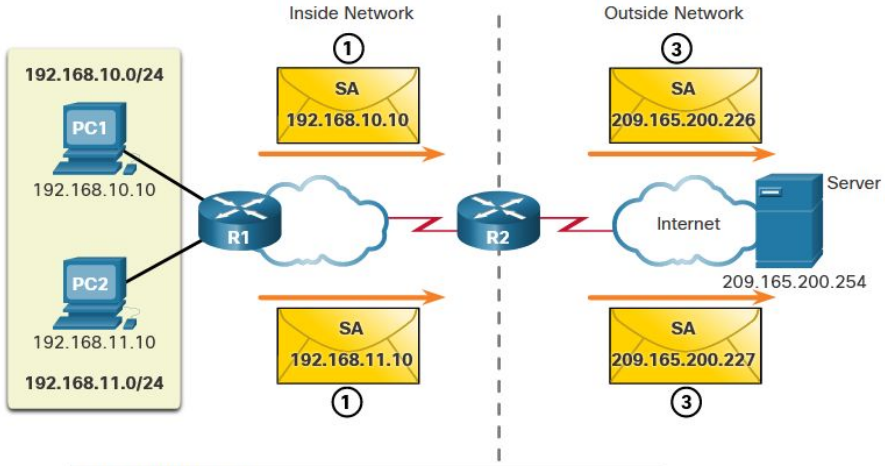
```
R2(config)# ip nat pool NAT-POOL1 209.165.200.226 209.165.200.240 netmask 255.255.255.224
R2(config)# access-list 1 permit 192.168.0.0 0.0.255.255
R2(config)# ip nat inside source list 1 pool NAT-POOL1
R2(config)# interface serial 0/1/0
R2(config-if)# ip nat inside
R2(config-if)# interface serial 0/1/1
R2(config-if)# ip nat outside
```



Dynamic NAT

Dynamic NAT translation process:

1. PC1 and PC2 send packets requesting a connection to the server.
2. R2 receives the first packet from PC1, checks the ALC to determine if the packet should be translated, selects an available global address, and creates a translation entry in the NAT table.
3. R2 replaces the inside local source address of PC1, 192.168.10.10, with the translated inside global address of 209.165.200.226 and forwards the packet. (The same process occurs for the packet from PC2 using the translated address of 209.165.200.227.)

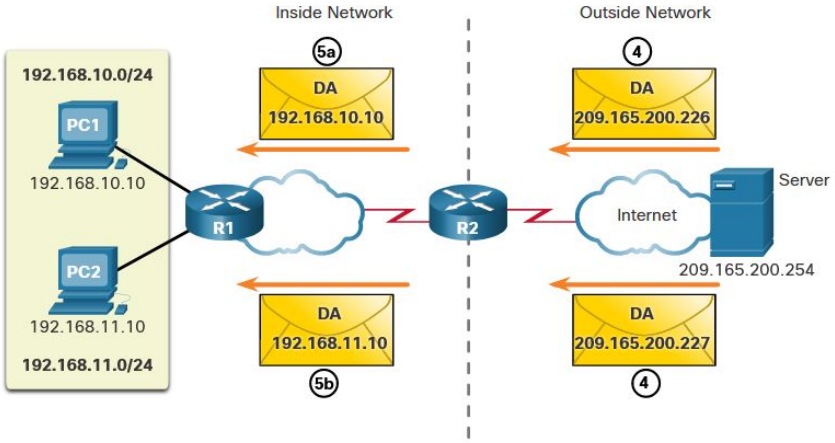


IPv4 NAT Pool	
Inside Local Address Pool	Inside Global Address
192.168.10.10	209.165.200.226
192.168.11.10	209.165.200.227

Dynamic NAT

Dynamic NAT translation process:

4. The server receives the packet from PC1 and responds using the destination address of 209.165.200.226. The server receives the packet from PC2, it responds to using the destination address of 209.165.200.227.
5. (a) When R2 receives the packet with the destination address of 209.165.200.226; it performs a NAT table lookup and translates the address back to the inside local address and forwards the packet toward PC1.
   (b) When R2 receives the packet with the destination address of 209.165.200.227; it performs a NAT table lookup and translates the address back to the inside local address 192.168.11.10 and forwards the packet toward PC2.
6. PC1 and PC2 receive the packets and continue the conversation. The router performs Steps 2 to 5 for each packet.



	IPv4 NAT Pool	
	Inside Local Address Pool	**Inside Global Address**
5a	192.168.10.10	209.165.200.226
5b	192.168.11.10	209.165.200.227

Dynamic NAT

The output of the **show ip nat translations** command displays all static translations that have been configured and any dynamic translations that have been created by traffic.

```
R2# show ip nat translations
Pro Inside global      Inside local       Outside local      Outside global
--- 209.165.200.228    192.168.10.10      ---                ---
--- 209.165.200.229    192.168.11.10      ---                ---
R2#
```

```
R2# show ip nat translation verbose
Pro Inside global      Inside local       Outside local      Outside global
tcp 209.165.200.228    192.168.10.10      ---                ---
    create 00:02:11, use 00:02:11 timeout:86400000, left 23:57:48, Map-Id(In): 1,
    flags:
none, use_count: 0, entry-id: 10, lc_entries: 0
tcp 209.165.200.229    192.168.11.10      ---                ---
    create 00:02:10, use 00:02:10 timeout:86400000, left 23:57:49, Map-Id(In): 1,
    flags:
none, use_count: 0, entry-id: 12, lc_entries: 0
R2#
```

Dynamic NAT

By default, translation entries time out after 24 hours, unless the timers have been reconfigured with the **ip nat translation timeout** *timeout-seconds* command in global configuration mode. To clear dynamic entries before the timeout has expired, use the **clear ip nat translation** privileged EXEC mode command.

```
R2# clear ip nat translation *
R2# show ip nat translation
```

Command	Description
clear ip nat translation *	Clears all dynamic address translation entries from the NAT translation table.
clear ip nat translation inside *global-ip local-ip* [**outside** *local-ip global-ip*]	Clears a simple dynamic translation entry containing an inside translation or both inside and outside translation.
clear ip nat translation *protocol* **inside** *global-ip global-port local-ip local-port* [**outside** *local-ip local-port global-ip global-port*]	Clears an extended dynamic translation entry.

Dynamic NAT

The **show ip nat statistics** command displays information about the total number of active translations, NAT configuration parameters, the number of addresses in the pool, and how many of the addresses have been allocated.

```
R2# show ip nat statistics
Total active translations: 4 (0 static, 4 dynamic; 0 extended)
Peak translations: 4, occurred 00:31:43 ago
Outside interfaces:
  Serial0/1/1
Inside interfaces:
  Serial0/1/0
Hits: 47  Misses: 0
CEF Translated packets: 47, CEF Punted packets: 0
Expired translations: 5
Dynamic mappings:
-- Inside Source
[Id: 1] access-list 1 pool NAT-POOL1 refcount 4
 pool NAT-POOL1: netmask 255.255.255.224
      start 209.165.200.226 end 209.165.200.240
      type generic, total addresses 15, allocated 2 (13%), misses 0
(output omitted)
R2#
```

Configure PAT

To configure PAT to use a single IPv4 address, add the keyword **overload** to the **ip nat inside source** command.

In the example, all hosts from network 192.168.0.0/16 (matching ACL 1) that send traffic through router R2 to the internet will be translated to IPv4 address 209.165.200.225 (IPv4 address of interface S0/1/1). The traffic flows will be identified by port numbers in the NAT table because the **overload** keyword is configured.

```
R2(config)# ip nat inside source list 1 interface serial 0/1/0 overload
R2(config)# access-list 1 permit 192.168.0.0 0.0.255.255
R2(config)# interface serial0/1/0
R2(config-if)# ip nat inside
R2(config-if)# exit
R2(config)# interface Serial0/1/1
R2(config-if)# ip nat outside
```

Configure PAT

An ISP may allocate more than one public IPv4 address to an organization. In this scenario the organization can configure PAT to use a pool of IPv4 public addresses for translation.

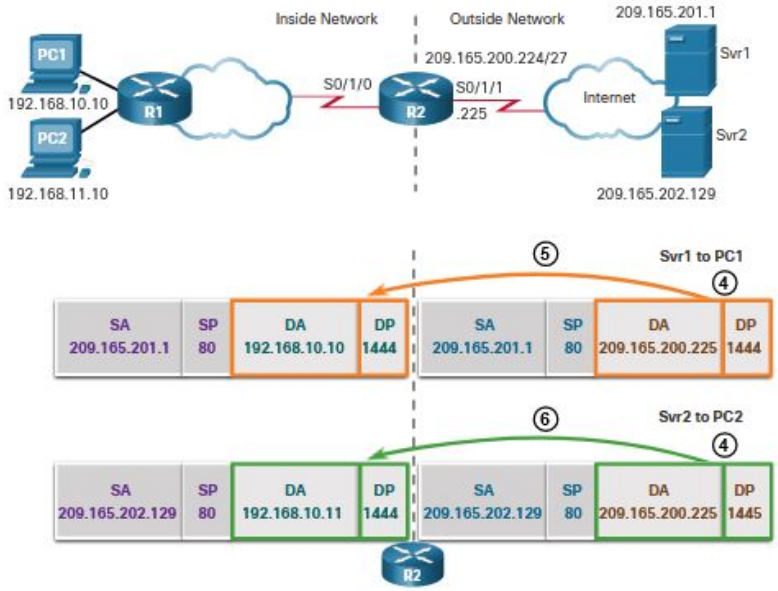
To configure PAT for a dynamic NAT address pool, simply add the keyword **overload** to the **ip nat inside source** command.

In the example, NAT-POOL2 is bound to an ACL to permit 192.168.0.0/16 to be translated. These hosts can share an IPv4 address from the pool because PAT is enabled with the keyword **overload**.

```
R2(config)# ip nat pool NAT-POOL2 209.165.200.226 209.165.200.240 netmask 255.255.255.224
R2(config)# access-list 1 permit 192.168.0.0 0.0.255.255
R2(config)# ip nat inside source list 1 pool NAT-POOL2 overload
R2(config)# interface serial0/1/0
R2(config-if)# ip nat inside
R2(config-if)# interface serial0/1/0
R2(config-if)# ip nat outside
```

Configure PAT

1. PC1 and PC2 send packets to Svr1 and Svr2.
2. The packet from PC1 reaches R2 first. R2 modifies the source IPv4 address to 209.165.200.225 (inside global address). The packet is then forwarded towards Svr1.
3. The packet from PC2 arrives at R2. PAT changes the source IPv4 address of PC2 to the inside global address 209.165.200.225. PC2 has the same source port number as the translation for PC1. PAT increments the source port number until it is a unique value in its table. In this instance, it is 1445.



NAT Table

Inside Local Address	Inside Global Address	Outside Global Address	Outside Local Address
192.168.10.10:1444	209.165.200.225:1444	209.165.201.1:80	209.165.201.1:80
192.168.10.11:1444	209.165.200.225:1445	209.165.201.129:80	209.165.202.129:80

Configure PAT

```
R2# show ip nat translations
Pro Inside global          Inside local         Outside local      Outside global
tcp 209.165.200.225:1444  192.168.10.10:1444  209.165.201.1:80   209.165.201.1:80
tcp 209.165.200.225:1445  192.168.11.10:1444  209.165.202.129:80 209.165.202.129:80
R2#
```

```
R2# show ip nat statistics
Total active translations: 4 (0 static, 2 dynamic; 2 extended)
Peak translations: 2, occurred 00:31:43 ago
Outside interfaces:
  Serial0/1/1
Inside interfaces:
  Serial0/1/0
Hits: 4  Misses: 0
CEF Translated packets: 47, CEF Punted packets: 0
Expired translations: 0
Dynamic mappings:
-- Inside Source
[Id: 3] access-list 1 pool NAT-POOL2 refcount 2
 pool NAT-POOL2: netmask 255.255.255.224
      start 209.165.200.225 end 209.165.200.240
      type generic, total addresses 15, allocated 1 (6%), misses 0
(output omitted)
R2#
```

END.