Security Configuration Guide: Access Control Lists, Cisco IOS XE 17 (Cisco NCS 520 Series)

First Published: 2019-11-26

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS

CHAPTER 1

IP Access List Overview

Access control lists (ACLs) perform packet filtering to control which packets move through the network and where. Such control provides security by helping to limit network traffic, restrict the access of users and devices to the network, and prevent traffic from leaving a network. IP access lists can reduce the chance of spoofing and denial-of-service attacks and allow dynamic, temporary user access through a firewall.

IP access lists can also be used for purposes other than security, such as bandwidth control, restricting the content of routing updates, redistributing routes, triggering dial-on-demand (DDR) calls, limiting debug output, and identifying or classifying traffic for quality of service (QoS) features. This module provides an overview of IP access lists.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to https://cfnng.cisco.com/. An account on Cisco.com is not required.

Information About IP Access Lists

Benefits of IP Access Lists

Access control lists (ACLs) perform packet filtering to control the flow of packets through a network. Packet filtering can restrict the access of users and devices to a network, providing a measure of security. Access lists can save network resources by reducing traffic. The benefits of using access lists are as follows:

- Authenticate incoming rsh and rcp requests—Access lists can simplify the identification of local users, remote hosts, and remote users in an authentication database that is configured to control access to a device. The authentication database enables Cisco software to receive incoming remote shell (rsh) and remote copy (rcp) protocol requests.

- Block unwanted traffic or users—Access lists can filter incoming or outgoing packets on an interface, thereby controlling access to a network based on source addresses, destination addresses, or user authentication. You can also use access lists to determine the types of traffic that are forwarded or blocked at device interfaces. For example, you can use access lists to permit e-mail traffic to be routed through a network and to block all Telnet traffic from entering the network.
- Control access to vty—Access lists on an inbound vty (Telnet) can control who can access the lines to a device. Access lists on an outbound vty can control the destinations that the lines from a device can reach.
- Identify or classify traffic for QoS features—Access lists provide congestion avoidance by setting the IP precedence for Weighted Random Early Detection (WRED) and committed access rate (CAR). Access lists also provide congestion management for class-based weighted fair queueing (CBWFQ), priority queueing, and custom queueing.
- Limit debug command output—Access lists can limit debug output based on an IP address or a protocol.
- Provide bandwidth control—Access lists on a slow link can prevent excess traffic on a network.
- Provide NAT control—Access lists can control which addresses are translated by Network Address Translation (NAT).
- Reduce the chance of DoS attacks—Access lists reduce the chance of denial-of-service (DoS) attacks. Specify IP source addresses to control traffic from hosts, networks, or users from accessing your network. Configure the TCP Intercept feature to can prevent servers from being flooded with requests for connection.
- Restrict the content of routing updates—Access lists can control routing updates that are sent, received, or redistributed in networks.
- Trigger dial-on-demand calls—Access lists can enforce dial and disconnect criteria.

Restrictions for Access Control Lists

- The **deny ip any any** command does not deny the packets that are not fragmented first when it is preceded by the permit tcp any any *port-number* or permit udp any any *port-number*command.

  Example:

  ```
  permit tcp any any eq 3000
  deny ip any any fragment
  ```

  TCP stream with port number 4000 is not denied for the packets that are not fragmented first. ACE works fine for the first fragment as it has TCP port information.

- Defining multiple access control parameters in a single ACL statement is not allowed. Each access control parameter should be defined with a unique ACL statement.

Restrictions for Security ACLs

- Egress ACLs are not supported.
- Ingress MAC ACLs are supported only EFP interfaces.

- Ingress IP ACLs are supported only on EFP interfaces.
- IPv6 ACLs are not supported.
- MAC ACL is not supported for reserved MAC addresses.
- In MAC ACLs, ether type is not supported as one of the filter criteria.
- ACL statistics is not supported.
- ACL logging is not supported.

Border Routers and Firewall Routers Should Use Access Lists

There are many reasons to configure access lists; for example, you can use access lists to restrict contents of routing updates or to provide traffic flow control. One of the most important reasons to configure access lists is to provide a basic level of security for your network by controlling access to it. If you do not configure access lists on your router, all packets passing through the router could be allowed onto all parts of your network.

An access list can allow one host to access a part of your network and prevent another host from accessing the same area. In the figure below, by applying an appropriate access list to the interfaces of the router, Host A is allowed to access the Human Resources network and Host B is prevented from accessing the Human Resources network.

Access lists should be used in firewall routers, which are often positioned between your internal network and an external network such as the Internet. You can also use access lists on a router positioned between two parts of your network, to control traffic entering or exiting a specific part of your internal network.

To provide some security benefits of access lists, you should at least configure access lists on border routers--routers located at the edges of your networks. Such an access list provides a basic buffer from the outside network or from a less controlled area of your own network into a more sensitive area of your network. On these border routers, you should configure access lists for each network protocol configured on the router interfaces. You can configure access lists so that inbound traffic or outbound traffic or both are filtered on an interface.

Access lists are defined on a per-protocol basis. In other words, you should define access lists for every protocol enabled on an interface if you want to control traffic flow for that protocol.

Definition of an Access List

An access list is a sequential list consisting of at least one **permit** statement and possibly one or more **deny** statements. In the case of IP access lists, the statements can apply to IP addresses, upper-layer IP protocols, or other fields in IP packets. The access list is identified and referenced by a name or a number. The access list acts as a packet filter, filtering packets based on the criteria defined in the access list.

An access list may be configured, but it does not take effect until the access list is either applied to an interface (with the **ip access-group** command), a virtual terminal line (vty) (with the **access-class**command), or referenced by some other command that accepts an access list. Access lists have many uses, and therefore many Cisco IOS software commands accept a reference to an access list in their command syntax. Multiple commands can reference the same access list.

In the following configuration excerpt, the first three lines are an example of an IP access list named branchoffices, which is applied to serial interface 0 on incoming packets. No sources other than those on the networks specified by each source address and mask pair can access this interface. The destinations for packets coming from sources on network 172.20.7.0 are unrestricted. The destination for packets coming from sources on network 172.29.2.0 must be 172.25.5.4.

```
ip access-list extended branchoffices
 10 permit 172.20.7.0 0.0.0.3 any
 20 permit 172.29.2.0 0.0.0.255 host 172.25.5.4
!
interface serial 0
 ip access-group branchoffices in
```

Software Processing of an Access List

The following general steps describe how the Cisco IOS software processes an access list when it is applied to an interface, a vty, or referenced by some other Cisco IOS command. These steps apply to an access list that has 13 or fewer access list entries.

- The software receives an IP packet and tests parts of each packet being filtered against the conditions in the access list, one condition (**permit** or **deny** statement) at a time. For example, the software tests the source and destination addresses of the packet against the source and destination addresses in a **permit** or **deny**statement.
- If a packet does not match an access list statement, the packet is then tested against the next statement in the list.
- If a packet and an access list statement match, the rest of the statements in the list are skipped and the packet is permitted or denied as specified in the matched statement. The first entry that the packet matches determines whether the software permits or denies the packet. That is, after the first match, no subsequent entries are considered.
- If the access list denies a packet, the software discards the packet and returns an ICMP Host Unreachable message.
- If no conditions match, the software drops the packet. This is because each access list ends with an unwritten, implicit **deny** statement. That is, if the packet has not been permitted by the time it was tested against each statement, it is denied.

In later Cisco IOS releases such as Release 12.4, 12.2S, and 12.0S, by default, an access list that has more than 13 access list entries is processed differently from one that has 13 or fewer entries. In order to be more efficient, an access list with more than 13 entries is processed using a trie-based lookup algorithm. This process will happen automatically; it does not need to be configured.

Access List Rules

The following rules apply to access lists:

- Only one access list per interface, per protocol, and per direction is allowed.
- An access list must contain at least one **permit** statement or all packets are denied entry into the network.
- The order in which access list conditions or match criteria are configured is important. While deciding whether to forward or block a packet, Cisco software tests the packet against each criteria statement in the order in which these statements are created. After a match is found, no more criteria statements are

checked. The same **permit** or **deny** statements specified in a different order can result in a packet being passed under one circumstance and denied in another circumstance.

- If an access list is referenced by a name, but the access list does not exist, all packets pass. An interface or command with an empty access list applied to it permits all traffic into the network.
- Standard access lists and extended access lists cannot have the same name.
- Inbound access lists process packets before the packets are routed to an outbound interface. Inbound access lists that have filtering criteria that deny packet access to a network saves the overhead of routing lookup. Packets that are permitted access to a network based on the configured filtering criteria are processed for routing. For inbound access lists, when you configure a **permit** statement, packets are processed after they are received, and when you configure a **deny** statement, packets are discarded.
- Outbound access lists process packets before they leave the device. Incoming packets are routed to the outbound interface and then processed by the outbound access list. For outbound access lists, when you configure a **permit** statement, packets are sent to the output buffer, and when you configure a **deny** statement, packets are discarded.
- An access list can control traffic arriving at a device or leaving a device, but not traffic originating at a device.

Helpful Hints for Creating IP Access Lists

The following tips will help you avoid unintended consequences and help you create more efficient, useful access lists.

- Create the access list before applying it to an interface (or elsewhere), because if you apply a nonexistent access list to an interface and then proceed to configure the access list, the first statement is put into effect, and the implicit **deny** statement that follows could cause you immediate access problems.
- Another reason to configure an access list before applying it is because an interface with an empty access list applied to it permits all traffic.
- All access lists need at least one **permit** statement; otherwise, all packets are denied and no traffic passes.
- Because the software stops testing conditions after it encounters the first match (to either a **permit** or **deny** statement), you will reduce processing time and resources if you put the statements that packets are most likely to match at the beginning of the access list. Place more frequently occurring conditions before less frequent conditions.
- Organize your access list so that more specific references in a network or subnet appear before more general ones.
- Use the statement **permit any any** if you want to allow all other packets not already denied. Using the statement **permit any any** in effect avoids denying all other packets with the implicit deny statement at the end of an access list. Do not make your first access list entry **permit any any** because all traffic will get through; no packets will reach the subsequent testing. In fact, once you specify **permit any any**, all traffic not already denied will get through.
- Although all access lists end with an implicit **deny** statement, we recommend use of an explicit **deny** statement (for example, **deny ip any any**). On most platforms, you can display the count of packets denied by issuing the **show access-list**command, thus finding out more information about who your access list is disallowing. Only packets denied by explicit **deny** statements are counted, which is why the explicit **deny** statement will yield more complete data for you.

- While you are creating an access list or after it is created, you might want to delete an entry.
    - You cannot delete an entry from a numbered access list; trying to do so will delete the entire access list. If you need to delete an entry, you need to delete the entire access list and start over.
    - You can delete an entry from a named access list. Use the **no permit**or **no deny** command to delete the appropriate entry.
- In order to make the purpose of individual statements more scannable and easily understood at a glance, you can write a helpful remark before or after any statement by using the **remark** command.
- If you want to deny access to a particular host or network and find out if someone from that network or host is attempting to gain access.
- This hint applies to the placement of your access list. When trying to save resources, remember that an inbound access list applies the filter conditions before the routing table lookup. An outbound access list applies the filter conditions after the routing table lookup.
- Before you add new ACL statements, provide time to the parser to clean up the deletion.

Named or Numbered Access Lists

All access lists must be identified by a name or a number. Named and numbered access lists have different command syntax. Named access lists are compatible with Cisco IOS Release 11.2 and later. Named access lists are more convenient than numbered access lists because you can specify a meaningful name that is easier to remember and associate with a purpose. You may reorder statements in or add statements to a named access list.

Named access list are newer than numbered access lists and support the following features that are not supported in numbered access lists:

- TCP flag filtering
- IP option filtering
- noncontiguous ports
- ability to delete entries with the **no permit** or **no deny** command

Not all commands that accept a numbered access list will accept a named access list. For example, virtual terminal lines use only numbered access lists.

Standard or Extended Access Lists

All access lists are either standard or extended access lists. If you only intend to filter on a source address, the simpler standard access list is sufficient. For filtering on anything other than a source address, an extended access list is necessary.

- Named access lists are specified as standard or extended based on the keyword **standard** or **extended** in the **ip access-list** command syntax.
- Numbered access lists are specified as standard or extended based on their number in the **access-list** command syntax. Standard IP access lists are numbered 1 to 99 or 1300 to 1999; extended IP access lists are numbered 100 to 199 or 2000 to 2699. The range of standard IP access lists was initially only 1 to

99, and was subsequently expanded with the range 1300 to 1999 (the intervening numbers were assigned to other protocols). The extended access list range was similarly expanded.

Standard Access Lists

Standard IP access lists test only source addresses of packets (except for two exceptions). Because standard access lists test source addresses, they are very efficient at blocking traffic close to a destination. There are two exceptions when the address in a standard access list is not a source address:

- On outbound VTY access lists, when someone is trying to telnet, the address in the access list entry is used as a destination address rather than a source address.
- When filtering routes, you are filtering the network being advertised to you rather than a source address.

Extended Access Lists

Extended access lists are good for blocking traffic anywhere. Extended access lists test source and destination addresses and other IP packet data, such as protocols, TCP or UDP port numbers, type of service (ToS), precedence, TCP flags, IP options, and TTL value. Extended access lists can also provide capabilities that standard access lists cannot, such as the following:

- Filtering IP Options
- Filtering TCP flags
- Filtering noninitial fragments of packets (see the module “Refining an IP Access List”)
- Time-based entries (see "Time-Based and Distributed Time-Based Access Lists" and the module “Refining an IP Access List”)
- Dynamic access lists (see the section "Types of IP Access Lists")

Note Packets that are subject to an extended access list will not be autonomous switched.

IP Packet Fields You Can Filter to Control Access

You can use an extended access list to filter on any of the following fields in an IP packet. Source address and destination address are the two most frequently specified fields on which to base an access list:

- Source address--Specifies a source address to control packets coming from certain networking devices or hosts.
- Destination address--Specifies a destination address to control packets being sent to certain networking devices or hosts.
- Protocol--Specifies an IP protocol indicated by the keyword **eigrp**, **gre**, **icmp**, **igmp**, **ip**, **ipinip**, **nos**, **ospf**, **tcp**, or **udp**, or indicated by an integer in the range from 0 to 255 (representing an Internet protocol). If you specify a transport layer protocol (**icmp**, **igmp**, **tcp**, or **udp**), the command has a specific syntax.
    - Ports and non-contiguous ports--Specifies TCP or UDP ports by a port name or port number. The port numbers can be noncontiguous port numbers. Port numbers can be useful to filter Telnet traffic or HTTP traffic, for example.

- TCP flags--Specifies that packets match any flag or all flags set in TCP packets. Filtering on specific TCP flags can help prevent false synchronization packets.

- IP options--Specifies IP options; one reason to filter on IP options is to prevent routers from being saturated with spurious packets containing them.

Wildcard Mask for Addresses in an Access List

Address filtering uses wildcard masking to indicate to the software whether to check or ignore corresponding IP address bits when comparing the address bits in an access list entry to a packet being submitted to the access list. By carefully setting wildcard masks, you can specify one or more IP addresses for permit or deny tests.

Wildcard masking for IP address bits uses the number 1 and the number 0 to specify how the software treats the corresponding IP address bits. A wildcard mask is sometimes referred to as an inverted mask because a 1 and 0 mean the opposite of what they mean in a subnet (network) mask.

- A wildcard mask bit 0 means check the corresponding bit value; they must match.
- A wildcard mask bit 1 means ignore that corresponding bit value; they need not match.

If you do not supply a wildcard mask with a source or destination address in an access list statement, the software assumes an implicit wildcard mask of 0.0.0.0, meaning all values must match.

Unlike subnet masks, which require contiguous bits indicating network and subnet to be ones, wildcard masks allow noncontiguous bits in the mask.

The table below shows examples of IP addresses and masks from an access list, along with the corresponding addresses that are considered a match.

Table 1: Sample IP Addresses, Wildcard Masks, and Match Results

Address	**Wildcard Mask**	**Match Results**
0.0.0.0	255.255.255.255	All addresses will match the access list conditions.
172.18.0.0/16	0.0.255.255	Network 172.18.0.0
172.18.5.2/16	0.0.0.0	Only host 172.18.5.2 matches
172.18.8.0	0.0.0.7	Only subnet 172.18.8.0/29 matches
172.18.8.8	0.0.0.7	Only subnet 172.18.8.8/29 matches
172.18.8.15	0.0.0.3	Only subnet 172.18.8.15/30 matches
10.1.2.0	0.0.254.255 (noncontiguous bits in mask)	Matches any even-numbered network in the range of 10.1.2.0 to 10.1.254.0

Access List Sequence Numbers

The ability to apply sequence numbers to IP access list entries simplifies access list changes. Prior to the IP Access List Entry Sequence Numbering feature, there was no way to specify the position of an entry within an access list. If you wanted to insert an entry in the middle of an existing list, all of the entries after the desired position had to be removed, then the new entry was added, and then all the removed entries had to be reentered. This method was cumbersome and error prone.

This feature allows users to add sequence numbers to access list entries and resequence them. When you add a new entry, you specify the sequence number so that it is in a desired position in the access list. If necessary, entries currently in the access list can be resequenced to create room to insert the new entry.

Access List Logging

The Cisco IOS software can provide logging messages about packets permitted or denied by a single standard or extended IP access list entry. That is, any packet that matches the entry will cause an informational logging message about the packet to be sent to the console. The level of messages logged to the console is controlled by the **logging console** global configuration command.

The first packet that triggers the access list entry causes an immediate logging message, and subsequent packets are collected over 5-minute intervals before they are displayed or logged. The logging message includes the access list number, whether the packet was permitted or denied, the source IP address of the packet, and the number of packets from that source permitted or denied in the prior 5-minute interval.

Even if you use the **ip access-list log-update** command, the 5-minute timer remains in effect, so each cache is emptied at the end of 5 minutes, regardless of the count of messages in each cache. Regardless of when the log message is sent, the cache is flushed and the count reset to 0 for that message the same way it is when a threshold is not specified.

Note The logging facility might drop some logging message packets if there are too many to be handled or if there is more than one logging message to be handled in 1 second. This behavior prevents the router from crashing due to too many logging packets. Therefore, the logging facility should not be used as a billing tool or an accurate source of the number of matches to an access list.

Alternative to Access List Logging

Packets matching an entry in an ACL with a log option are process switched. It is not recommended to use the log option on ACLs, but rather use NetFlow export and match on a destination interface of Null0. This is done in the CEF path. The destination interface of Null0 is set for any packet that is dropped by the ACL.

Additional IP Access List Features

Beyond the basic steps to create a standard or extended access list, you can enhance your access lists as mentioned below. Each of these methods is described completely in the module entitled “Refining an Access List.”

- You can impose dates and times when **permit** or **deny** statements in an extended access list are in effect, making your access list more granular and specific to an absolute or periodic time period.
- After you create a named access list, you might want to add entries or change the order of the entries, known as resequencing an access list.
- You can achieve finer granularity when filtering packets by filtering on noninitial fragments of packets.

Time-Based and Distributed Time-Based Access Lists

Time-based access lists implement access list entries based on particular times of the day or week. This is an advantage when you don’t want access list entries always in effect or in effect as soon as they are applied. Use time-based access lists to make the enforcement of permit or deny conditions granular, based on time and date.

Distributed time-based access lists are those that are supported on line cards for the Cisco 7500 series routers. Packets destined for an interface configured with time-based access lists are distributed switched through the line card.

Types of IP Access Lists

There are several types of access lists that are distinct because of how they are triggered, their temporary nature, or how their behavior differs from an ordinary access list.

Authentication Proxy

Authentication proxy provides dynamic, per-user authentication and authorization, authenticating users against industry standard TACACS+ and RADIUS authentication protocols. Authenticating and authorizing connections by users provides more robust protection against network attacks.

Context-Based Access Control

Context-based access control (CBAC) examines not only network layer and transport layer information, but also the application-layer protocol information (such as FTP information) to learn about the state of TCP and UDP connections. CBAC maintains connection state information for individual connections. This state information is used to make intelligent decisions about whether packets should be permitted or denied, and dynamically creates and deletes temporary openings in the firewall.

Dynamic Access Lists with the Lock-and-Key Feature

Dynamic access lists provide temporary access to designated users who are using Telnet to reach designated hosts through a firewall. Dynamic access lists involve user authentication and authorization.

Where to Apply an Access List

If you are applying an access list to an interface, carefully consider whether to specify it as **in** (inbound) or **out** (outbound). Applying an access list to an incoming or outgoing interface controls the traffic that will enter or leave the router’s interface or process level (in the case of filtering on TTL values).

- When an inbound access list is applied to an interface, after the software receives a packet, the software checks the packet against the access list statements. If the access list permits the packet, the software continues to process the packet. Therefore, filtering on incoming packets can save router resources because filtered packets will not go through the router.
- Access lists that apply to outbound packets are filtering packets that have already gone through the router. Packets that pass the access list are transmitted (sent) out the interface.
- The TCP ACL splitting feature of Rate-Based Satellite Control Protocol (RBSCP) is an example of a feature that can be used on an outgoing interface. The access list controls which packets are subject to TCP ACK splitting.

Access lists can be used in ways other than applying them to interfaces. The following are additional places to apply an access list.

- To restrict incoming and outgoing connections between a particular vty (into a Cisco device) and the network devices at addresses in an access list, apply an access list to a line. See the “Controlling Access to a Virtual Terminal Line” module.
- Referencing an access list from a **debug** command limits the amount of information displayed to only the information permitted by the access list, such as sources, destinations, or protocols, for example.
- Access lists can be used to control routing updates, to control dial-on-demand routing (DDR), and to control quality of service (QoS) features, for example. See the appropriate configuration chapters for using access lists with these features.

Where to Go Next

You must first decide what you want to restrict, and then select the type of access list that achieves your goal. Next, you will create an access list that permits or denies packets based on values in the fields you specify, and finally, you will apply the access list (which determines its placement).

Assuming you have decided what you want to restrict and what type of access list you need, your next step is to create an access list. Creating an access list based on source address, destination address, or protocol is described in the “Creating an IP Access List and Applying It to an Interface” module. You could create an access list that filters on other fields, as described in “Creating an IP Access List to Filter IP Options, TCP Flags, Noncontiguous Ports, or TTL Values.” If you want to control access to a virtual line, see “Controlling Access to a Virtual Terminal Line.” If the purpose of your access list is to control routing updates or QoS features, for example, see the appropriate technology chapter.

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
IP access list commands: complete command syntax, command mode, command history, defaults, usage guidelines, and examples	*Cisco IOS IP Application Services Command Reference*
Filtering on source address, destination address, or protocol	“Creating an IP Access List and Applying It to an Interface”
Filtering on IP Options, TCP flags, noncontiguous ports, or TTL	“Creating an IP Access List to Filter IP Options, TCP Flags, Noncontiguous Ports, or TTL Values”
Restricting access to a vty line.	"Controlling Access to a Virtual Terminal Line"

Standards

Standard	Title
None	--

MIBs

MIB	MIBs Link
None	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
None	--

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

CHAPTER 2

Creating an IP Access List and Applying It to an Interface

IP access lists provide many benefits for securing a network and achieving nonsecurity goals, such as determining quality of service (QoS) factors or limiting **debug** command output. This module describes how to create standard, extended, named, and numbered IP access lists. An access list can be referenced by a name or a number. Standard access lists filter on only the source address in IP packets. Extended access lists can filter on source address, destination address, and other fields in an IP packet.

After you create an access list, you must apply it to something in order for it to have any effect. This module describes how to apply an access list to an interface. However, there are many other uses for an access list, which are referenced in this module and described in other modules and in other configuration guides for various technologies.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to https://cfnng.cisco.com/. An account on Cisco.com is not required.

Prerequisites for Creating an IP Access List and Applying It to an Interface

Before you create or apply an IP access list, you should understand the concepts in the "IP Access List Overview" module. You should also have IP running in your network.

Restrictions for Creating an IP Access List and Applying It to an Interface

The following restrictions apply when configuring IPv4 Access Control Lists (ACLs):

- Application control engine (ACE)-specific counters are not supported.
- MAC ACLs are not supported on Ethernet flow points (EFPs) or trunk EFP interfaces to which Layer 3 IPv4 ACLs are applied.
- A maximum of 256 TCAM is supported. Hence 256 ACEs per ACL are supported.
- IPv4 ACL is supported in EFP interface only.
- Object-groups are not supported on IP ACLs.
- ACL statistics is not supported.
- If the first statement is **deny permit**, you must add statement for the **permit** traffic.

Information About Creating an IP Access List and Applying It to an Interface

Helpful Hints for Creating IP Access Lists

The following tips will help you avoid unintended consequences and help you create more efficient access lists.

- Create the access list before applying it to an interface (or elsewhere), because if you apply a nonexistent access list to an interface and then proceed to configure the access list, the first statement is put into effect, and the implicit **deny** statement that follows could cause you immediate access problems.
- Another reason to configure an access list before applying it is because an interface with an empty access list applied to it permits all traffic.
- All access lists need at least one **permit** statement; otherwise, all packets are denied and no traffic passes.
- Because the software stops testing conditions after it encounters the first match (to either a **permit** or **deny** statement), you will reduce processing time and resources if you put the statements that packets

are most likely to match at the beginning of the access list. Place more frequently occurring conditions before less frequent conditions.

- Organize your access list so that more specific references in a network or subnet appear before more general ones.
- Use the statement **permit any any** if you want to allow all other packets not already denied. Using the statement **permit any any** in effect avoids denying all other packets with the implicit deny statement at the end of an access list. Do not make your first access list entry **permit any any** because all traffic will get through; no packets will reach the subsequent testing. In fact, once you specify **permit any any**, all traffic not already denied will get through.
- Although all access lists end with an implicit **deny** statement, we recommend use of an explicit **deny** statement (for example, **deny ip any any**). On most platforms, you can display the count of packets denied by issuing the **show access-list**command, thus finding out more information about who your access list is disallowing. Only packets denied by explicit **deny** statements are counted, which is why the explicit **deny** statement will yield more complete data for you.
- While you are creating an access list or after it is created, you might want to delete an entry.
    - You cannot delete an entry from a numbered access list; trying to do so will delete the entire access list. If you need to delete an entry, you need to delete the entire access list and start over.
    - You can delete an entry from a named access list. Use the **no permit**or **no deny** command to delete the appropriate entry.
- In order to make the purpose of individual statements more scannable and easily understood at a glance, you can write a helpful remark before or after any statement by using the **remark** command.
- If you want to deny access to a particular host or network and find out if someone from that network or host is attempting to gain access.
- This hint applies to the placement of your access list. When trying to save resources, remember that an inbound access list applies the filter conditions before the routing table lookup. An outbound access list applies the filter conditions after the routing table lookup.

Access List Remarks

You can include comments or remarks about entries in any IP access list. An access list remark is an optional remark before or after an access list entry that describes the entry so that you do not have to interpret the purpose of the entry. Each remark is limited to 100 characters in length.

The remark can go before or after a **permit** or **deny** statement. Be consistent about where you add remarks. Users may be confused if some remarks precede the associated **permit** or **deny** statements and some remarks follow the associated statements.

The following is an example of a remark that describes function of the subsequent **deny** statement:

```
ip access-list extended telnetting
 remark Do not allow host1 subnet to telnet out
 deny tcp host 172.16.2.88 any eq telnet
```

Additional IP Access List Features

Beyond the basic steps to create a standard or extended access list, you can enhance your access lists as mentioned below. Each of these methods is described completely in the *Refining an IP Access List module*.

- You can impose dates and times when **permit** or **deny** statements in an extended access list are in effect, making your access list more granular and specific to an absolute or periodic time period.
- After you create a named or numbered access list, you might want to add entries or change the order of the entries, which are known as resequencing an access list.
- You can achieve finer granularity when filtering packets by filtering on noninitial fragments of packets.

How to Create an IP Access List and Apply It to an Interface

This section describes the general ways to create a standard or extended access list using either a name or a number. Access lists are very flexible; the tasks simply illustrate one **permit** command and one **deny** command to provide you the command syntax of each. Only you can determine how many **permit** and **deny** commands you need and their order.

Note The first two tasks in this module create an access list; you must apply the access list in order for it to function. If you want to apply the access list to an interface, perform the task "Applying the Access List to an Interface". If you don’t intend to apply the access list to an interface, see the "Where to Go Next" for pointers to modules that describe other ways to apply access lists.

Creating a Standard Access List to Filter on Source Address

If you want to filter on source address only, a standard access list is simple and sufficient. There are two alternative types of standard access list: named and numbered. Named access lists allow you to identify your access lists with a more intuitive name rather than a number, and they also support more features than numbered access lists.

Creating a Named Access List to Filter on Source Address

Use a standard, named access list if you need to filter on source address only. This task illustrates one **permit** statement and one **deny** statement, but the actual statements you use and their order depend on what you want to filter or allow. Define your **permit** and **deny** statements in the order that achieves your filtering goals.

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **ip access-list standard** *name*

Example:

```
Device(config)# ip access-list standard R&D
```

Defines a standard IP access list using a name and enters standard named access list configuration mode.

Step 4 **remark** *remark*

Example:

```
Device(config-std-nacl)# remark deny Sales network
```

(Optional) Adds a user-friendly comment about an access list entry.

- A remark can precede or follow an access list entry.
- In this example, the remark reminds the network administrator that the subsequent entry denies the Sales network access to the interface (assuming this access list is later applied to an interface).

Step 5 **deny** {*source* [*source-wildcard*] | **any**}

Example:

```
Device(config-std-nacl)# deny 172.16.0.0 0.0.255.255
```

(Optional) Denies the specified source based on a source address and wildcard mask.

- If the *source-wildcard* is omitted, a wildcard mask of 0.0.0.0 is assumed, meaning match on all bits of the source address.
- Optionally use the keyword **any** as a substitute for the *source source-wildcard* to specify the source and source wildcard of 0.0.0.0 255.255.255.255.
- In this example, all hosts on network 172.16.0.0 are denied passing the access list.

Step 6 **remark** *remark*

Example:

```
Device(config-std-nacl)# remark Give access to Tester’s host
```

(Optional) Adds a user-friendly comment about an access list entry.

- A remark can precede or follow an access list entry.
- This remark reminds the network administrator that the subsequent entry allows the Tester’s host access to the interface.

Step 7 **permit** {*source* [*source-wildcard*] | **any**}

Example:

```
Device(config-std-nacl)# permit 172.18.5.22 0.0.0.0
```

Permits the specified source based on a source address and wildcard mask.

- Every access list needs at least one **permit** statement; it need not be the first entry.
- If the *source-wildcard* is omitted, a wildcard mask of 0.0.0.0 is assumed, meaning match on all bits of the source address.
- Optionally use the keyword **any** as a substitute for the *source source-wildcard* to specify the source and source wildcard of 0.0.0.0 255.255.255.255.
- In this example, host 172.18.5.22 is allowed to pass the access list.

Step 8 Repeat some combination of Steps 4 through 7 until you have specified the sources on which you want to base your access list.

Remember that all sources not specifically permitted are denied by an implicit **deny** statement at the end of the access list.

Step 9 **end**

Example:

```
Device(config-std-nacl)# end
```

Exits standard named access list configuration mode and enters privileged EXEC mode.

Step 10 **show ip access-list**

Example:

```
Device# show ip access-list
```

(Optional) Displays the contents of all current IP access lists.

Creating a Numbered Access List to Filter on Source Address

Configure a standard, numbered access list if you need to filter on source address only and you prefer not to use a named access list.

IP standard access lists are numbered 1 to 99 or 1300 to 1999. This task illustrates one **permit** statement and one **deny** statement, but the actual statements you use and their order depend on what you want to filter or allow. Define your **permit** and **deny** statements in the order that achieves your filtering goals.

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **access-list** *access-list-number permit* {*source* [*source-wildcard*] | **any**}

Example:

```
Device(config)# access-list 1 permit 172.16.5.22 0.0.0.0
```

Permits the specified source based on a source address and wildcard mask.

- Every access list needs at least one permit statement; it need not be the first entry.
- Standard IP access lists are numbered 1 to 99 or 1300 to 1999.
- If the source-wildcard is omitted, a wildcard mask of 0.0.0.0 is assumed, meaning match on all bits of the source address.
- Optionally use the keyword any as a substitute for the source source-wildcard to specify the source and source wildcard of 0.0.0.0 255.255.255.255.
- In this example, host 172.16.5.22 is allowed to pass the access list.

Step 4 **access-list** *access-list-number* **deny** {*source* [*source-wildcard*] | **any**}

Example:

```
Device(config)# access-list 1 deny 172.16.7.34 0.0.0.0
```

Denies the specified source based on a source address and wildcard mask.

- If the *source-wildcard* is omitted, a wildcard mask of 0.0.0.0 is assumed, meaning match on all bits of the source address.
- Optionally use the abbreviation **any** as a substitute for the *source source-wildcard* to specify the source and source wildcard of 0.0.0.0 255.255.255.255.
- In this example, host 172.16.7.34 is denied passing the access list.

Step 5 Repeat some combination of Steps 3 through 6 until you have specified the sources on which you want to base your access list.

Remember that all sources not specifically permitted are denied by an implicit **deny** statement at the end of the access list.

Step 6 **end**

Example:

```
Device(config)# end
```

Exits global configuration mode and enters privileged EXEC mode.

Step 7 **show ip access-list**

Example:

```
Device# show ip access-list
```

(Optional) Displays the contents of all current IP access lists.

Creating an Extended Access List

If you want to filter on anything other than source address, you need to create an extended access list. There are two alternative types of extended access list: named and numbered. Named access lists allow you to identify your access lists with a more intuitive name rather than a number, and they also support more features.

For details on how to filter something other than source or destination address, see the syntax descriptions in the command reference documentation.

Creating a Named Extended Access List

Create a named extended access list if you want to filter on source and destination address, or a combination of addresses and other IP fields.

Step 1 **enable**

Example:

```
Router> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Router# configure terminal
```

Enters global configuration mode.

Step 3 **ip access-list extended** *name*

Example:

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

Defines an extended IP access list using a name and enters extended named access list configuration mode.

Step 4 **remark** *remark*

Example:

```
Router(config-ext-nacl)# remark protect server by denying access from the Marketing network
```

(Optional) Adds a user-friendly comment about an access list entry.

- A remark can precede or follow an access list entry.
- In this example, the remark reminds the network administrator that the subsequent entry denies the Sales network access to the interface.

Step 5 **deny** *protocol source* [*source-wildcard*] *destination* [*destination-wildcard*] [**option** *option-name*] [**precedence** *precedence*] [**tos** *tos*] [**established**] [**time-range** *time-range-name*] [**fragments**]

Example:

```
Router(config-ext-nacl)# deny ip 172.18.0.0 0.0.255.255 host 172.16.40.10
```

(Optional) Denies any packet that matches all of the conditions specified in the statement.

- If the *source-wildcard* or *destination-wildcard*isomitted, a wildcard mask of 0.0.0.0 is assumed, meaning match on all bits of the source or destination address, respectively.
- Optionally use the keyword **any** as a substitute for the *source source-wildcard*or *destination destination-wildcard*to specify the address and wildcard of 0.0.0.0 255.255.255.255.
- Optionally use the keyword **host** *source* to indicate a source and source wildcard of *source* 0.0.0.0 or the abbreviation **host** *destination*to indicate a destination and destination wildcard of *destination* 0.0.0.0.

Step 6 **remark** *remark*

Example:

```
Router(config-ext-nacl)# remark allow TCP from any source to any destination
```

(Optional) Adds a user-friendly comment about an access list entry.

- A remark can precede or follow an access list entry.

Step 7 **permit** *protocol source* [*source-wildcard*] *destination* [*destination-wildcard*] [**option** *option-name*] [**precedence** *precedence*] [**tos** *tos*] [**established**] [**time-range** *time-range-name*] [**fragments**]

Example:

```
Router(config-ext-nacl)# permit tcp any any
```

Permits any packet that matches all of the conditions specified in the statement.

- Every access list needs at least one permit statement.
- If the *source-wildcard* or *destination-wildcard*isomitted, a wildcard mask of 0.0.0.0 is assumed, meaning match on all bits of the source or destination address, respectively.
- Optionally use the keyword **any** as a substitute for the *source source-wildcard*or *destination destination-wildcard*to specify the address and wildcard of 0.0.0.0 255.255.255.255.
- In this example, TCP packets are allowed from any source to any destination.

Step 8 Repeat some combination of Steps 4 through 7 until you have specified the fields and values on which you want to base your access list.

Remember that all sources not specifically permitted are denied by an implicit **deny** statement at the end of the access list.

Step 9 **end**

Example:

```
Router(config-ext-nacl)# end
```

Ends configuration mode and brings the system to privileged EXEC mode.

Step 10 **show ip access-list**

Example:

```
Router# show ip access-list
```

(Optional) Displays the contents of all current IP access lists.

Creating a Numbered Extended Access List

Create a numbered extended access list if you want to filter on source and destination address, or a combination of addresses and other IP fields, and you prefer not to use a name. Extended IP access lists are numbered 100 to 199 or 2000 to 2699.

Step 1 **enable**

Example:

```
Router> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Router# configure terminal
```

Enters global configuration mode.

Step 3 **access-list** *access-list-number* **remark** *remark*

Example:

```
Router(config)# access-list 107 remark allow Telnet packets from any source to network 172.69.0.0
(headquarters)
```

(Optional) Adds a user-friendly comment about an access list entry.

- A remark of up to 100 characters can precede or follow an access list entry.

Step 4 **access-list** *access-list-number* **permit** *protocol* {*source* [*source-wildcard*] | **any**} {*destination* [*destination-wildcard*] | **any**} [**precedence** *precedence*] [**tos** *tos*] [**established**] [**time-range** *time-range-name*] [**fragments**]

Example:

```
Router(config)# access-list 107 permit tcp any 172.69.0.0 0.0.255.255 eq telnet
```

Permits any packet that matches all of the conditions specified in the statement.

- Every access list needs at least one **permit** statement; it need not be the first entry.
- Extended IP access lists are numbered 100 to 199 or 2000 to 2699.
- If the *source-wildcard* or *destination-wildcard*isomitted, a wildcard mask of 0.0.0.0 is assumed, meaning match on all bits of the source or destination address, respectively.
- Optionally use the keyword **any** as a substitute for the *source source-wildcard*or *destination destination-wildcard*to specify the address and wildcard of 0.0.0.0 255.255.255.255.
- TCP and other protocols have additional syntax available. See the **access-list** command in the command reference for complete syntax.

Step 5 **access-list** *access-list-number* **remark** *remark*

Example:

```
Router(config)# access-list 107 remark deny all other TCP packets
```

(Optional) Adds a user-friendly comment about an access list entry.

- A remark of up to 100 characters can precede or follow an access list entry.

Step 6 **access-list** *access-list-number* **deny** *protocol* {*source* [*source-wildcard*] | **any**} {*destination* [*destination-wildcard*] | **any**} [**precedence** *precedence*] [**tos** *tos*] [**established**] [**time-range** *time-range-name*] [**fragments**]

Example:

```
Router(config)# access-list 107 deny tcp any any
```

Denies any packet that matches all of the conditions specified in the statement.

- If the *source-wildcard* or *destination-wildcard*isomitted, a wildcard mask of 0.0.0.0 is assumed, meaning match on all bits of the source or destination address, respectively.
- Optionally use the keyword **any** as a substitute for the *source source-wildcard*or *destination destination-wildcard*to specify the address and wildcard of 0.0.0.0 255.255.255.255.

Step 7 Repeat some combination of Steps 3 through 6 until you have specified the fields and values on which you want to base your access list.

Remember that all sources not specifically permitted are denied by an implicit **deny** statement at the end of the access list.

Step 8 **end**

Example:

```
Router(config)# end
```

Ends configuration mode and brings the system to privileged EXEC mode.

Step 9 **show ip access-list**

Example:

```
Router# show ip access-list
```

(Optional) Displays the contents of all current IP access lists.

Applying the Access List to an Interface

Perform this task to apply an access list to an interface.

Step 1 **enable**

Example:

```
Router> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Router# configure terminal
```

Enters global configuration mode.

Step 3 **interface** *type slot /subslot /port*

Example:

```
Router(config)# interface GigabitEthernet 0/0/1
```

Specifies an interface and enters interface configuration mode.

Step 4 **ip access-group** {*access-list-number* | *access-list-name*} {**in** | **out**}

Example:

```
Router(config-if)# ip access-group noncorp in
```

Applies the specified access list to the incoming or outgoing interface.

- When you are filtering on source addresses, you typically apply the access list to an incoming interface.
- Filtering on source addresses is most efficient when applied near the destination.

Configuration Examples for Creating an IP Access List and Applying It to an Interface

Example: Filtering on Source Address (Hosts)

In the following example, the workstation belonging to Jones is allowed access to Gigabitethernet interface 0/0/2 and the workstation belonging to Smith is not allowed access:

```
interface Gigabitethernet 0/0/2
service instance 10 ethernet
 ip access-group workstations in
!
ip access-list standard workstations
 remark Permit only Jones workstation through
 permit 172.16.2.88
 remark Do not allow Smith workstation through
 deny 172.16.3.13
```

Example: Filtering on Source Address (Subnet)

In the following example, the Jones subnet is not allowed access to Gigabitethernet interface 0/0/2, but the Main subnet is allowed access:

```
interface Gigabitethernet 0/0/0
service instance 10 ethernet
 ip access-group prevention in
!
ip access-list standard prevention
 remark Do not allow Jones subnet through
 deny 172.22.0.0 0.0.255.255
 remark Allow Main subnet
 permit 172.25.0.0 0.0.255.255
```

Example: Filtering on Source Address Destination Address and IP Protocols

The following configuration example shows an interface with two access lists, one applied to incoming packets. The only packets allowed out the interface must be from source 172.16.3.4.

The extended access list named marketing_group filters incoming packets. The access list permits Telnet packets from any source to network 172.26.0.0 and denies all other TCP packets. It permits any ICMP packets. It denies UDP packets from any source to network 172.26.0 0 on port numbers less than 1024. Finally, the access list denies all other IP packets and performs logging of packets passed or denied by that entry.

```
interface Gigabitethernet 0/0/5
service instance 10 ethernet


 ip access-group marketing_group in
!

ip access-list extended marketing_group
```

```
 permit tcp any 172.26.0.0 0.0.255.255 eq telnet
 deny tcp any any
 permit icmp any any
 deny udp any 172.26.0.0 0.0.255.255 lt 1024
 deny ip any any
```

Example: Filtering on Source Address (Host and Subnets) Using a Numbered Access List

In the following example, network 10.0.0.0 is a Class A network whose second octet specifies a subnet; that is, its subnet mask is 255.255.0.0. The third and fourth octets of a network 10.0.0.0 address specify a particular host. Using access list 2, the Cisco IOS software would accept one address on subnet 48 and reject all others on that subnet. The last line of the list shows that the software would accept addresses on all other network 10.0.0.0 subnets.

```
interface Gigabitethernet 0/0/3
service instance 10 ethernet
 ip access-group 2 in
!
access-list 2 permit 10.48.0.3
access-list 2 deny 10.48.0.0  0.0.255.255
access-list 2 permit 10.0.0.0  0.255.255.255
```

Example: Preventing Telnet Access to a Subnet

In the following example, the Jones subnet is not allowed to Telnet out Gigabitethernet interface 0/0/2:

```
interface Gigabitethernet 0/0/2
service instance 10 ethernet
 ip access-group telnetting  in
!
ip access-list extended telnetting
 remark Do not allow Jones subnet to telnet out
 deny tcp 172.20.0.0 0.0.255.255 any eq telnet
 remark Allow Top subnet to telnet out
 permit tcp 172.33.0.0 0.0.255.255 any eq telnet
```

Example: Filtering on TCP and ICMP Using Port Numbers

In the following example, the first line of the extended access list named goodports permits any incoming TCP connections with destination ports greater than 1023. The second line permits incoming TCP connections to the Simple Mail Transfer Protocol (SMTP) port of host 172.28.1.2. The last line permits incoming ICMP messages for error feedback.

```
interface Gigabitethernet 0/0/2
service instance 10 ethernet
 ip access-group goodports in
!
ip access-list extended goodports
 permit tcp any 172.28.0.0 0.0.255.255 gt 1023
 permit tcp any host 172.28.1.2 eq 25
 permit icmp any 172.28.0.0 255.255.255.255
```

Example: Allowing SMTP (E-mail) and Established TCP Connections

Suppose you have a network connected to the Internet, and you want any host on an Ethernet to be able to form TCP connections to any host on the Internet. However, you do not want IP hosts to be able to form TCP connections to hosts on the Ethernet except to the mail (SMTP) port of a dedicated mail host.

SMTP uses TCP port 25 on one end of the connection and a random port number on the other end. The same two port numbers are used throughout the life of the connection. Mail packets coming in from the Internet will have a destination port of 25. Outbound packets will have the port numbers reversed. The fact that the secure system behind the router always will accept mail connections on port 25 is what makes possible separate control of incoming and outgoing services. The access list can be configured on either the outbound or inbound interface.

In the following example, the Ethernet network is a Class B network with the address 172.18.0.0, and the address of the mail host is 172.18.1.2. The **established**keyword is used only for the TCP protocol to indicate an established connection. A match occurs if the TCP datagram has the ACK or RST bits set, which indicate that the packet belongs to an existing connection.

```
interface Gigabitethernet 0/0/2
service instance 10 ethernet
 ip access-group 102 in
!
access-list 102 permit tcp any 172.18.0.0 0.0.255.255 established
access-list 102 permit tcp any host 172.18.1.2 eq 25
```

Example: Preventing Access to the Web By Filtering on Port Name

In the following example, the Winter and Smith workstations are not allowed web access; other hosts on network 172.20.0.0 are allowed web access:

```
interface Gigabitethernet 0/0/2
service instance 10 ethernet
 ip access-group no_web  in
!
ip access-list extended no_web
 remark Do not allow Winter to browse the web
 deny host 172.20.3.85 any eq http
 remark Do not allow Smith to browse the web
 deny host 172.20.3.13 any eq http
 remark Allow others on our network to browse the web
 permit 172.20.0.0 0.0.255.255 any eq http
```

Example: Limiting Debug Output

The following sample configuration uses an access list to limit the **debug** command output. Limiting the **debug** output restricts the volume of data to what you are interested in, saving you time and resources.

```
Device(config)# ip access-list standard acl1

!Displays only advertisements for LDP peer in acl1
Device(config-std-nacl)# permit host 10.0.0.44

Device# debug mpls ldp advertisements peer-acl acl1

tagcon: peer 10.0.0.44:0 (pp 0x60E105BC): advertise 172.17.0.33
tagcon: peer 10.0.0.44:0 (pp 0x60E105BC): advertise 172.16.0.31
```

```
tagcon: peer 10.0.0.44:0 (pp 0x60E105BC): advertise 172.22.0.33
tagcon: peer 10.0.0.44:0 (pp 0x60E105BC): advertise 192.168.0.1
tagcon: peer 10.0.0.44:0 (pp 0x60E105BC): advertise 192.168.0.3
tagcon: peer 10.0.0.44:0 (pp 0x60E105BC): advertise 192.168.1.33
```

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/mcl/allreleasemcl/all-book.html

Standards and RFCs

Standard/RFC	Title
No specific Standards and RFCs are supported by the features in this document.	—

MIBs

MIB	MIBs Link
—	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

CHAPTER 3

Creating an IP Access List to Filter IP Options, TCP Flags, Noncontiguous Ports

This module describes how to use an IP access list to filter IP packets that contain certain IP Options, TCP flags, noncontiguous ports.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to https://cfnng.cisco.com/. An account on Cisco.com is not required.

Prerequisites for Creating an IP Access List to Filter IP Options TCP Flags Noncontiguous Ports

Before you perform any of the tasks in this module, you should be familiar with the information in the following modules:

- “IP Access List Overview”
- “Creating an IP Access List and Applying It to an Interface”

Information About Creating an IP Access List to Filter IP Options, TCP Flags, Noncontiguous Ports

IP Options

IP uses four key mechanisms in providing its service: Type of Service, Time to Live, Options, and Header Checksum.

The Options, commonly referred to as IP Options, provide for control functions that are required in some situations but unnecessary for the most common communications. IP Options include provisions for time stamps, security, and special routing.

IP Options may or may not appear in datagrams. They must be implemented by all IP modules (host and gateways). What is optional is their transmission in any particular datagram, not their implementation. In some environments the security option may be required in all datagrams.

The option field is variable in length. There may be zero or more options. IP Options can have one of two formats:

- Format 1: A single octet of option-type.
- Format 2: An option-type octet, an option-length octet, and the actual option-data octets.

The option-length octet counts the option-type octet, the option-length octet, and the option-data octets.

The option-type octet is viewed as having three fields: a 1-bit copied flag, a 2-bit option class, and a 5-bit option number. These fields form an 8-bit value for the option type field. IP Options are commonly referred to by their 8-bit value.

For a complete list and description of IP Options, refer to RFC 791, *Internet Protocol* at the following URL: http://www.faqs.org/rfcs/rfc791.html

Benefits of Filtering IP Options

- Filtering of packets that contain IP Options from the network relieves downstream devices and hosts of the load from options packets.
- This feature also minimizes load to the Route Processor (RP) for packets with IP Options that require RP processing on distributed systems. Previously, the packets were always routed to or processed by the RP CPU. Filtering the packets prevents them from impacting the RP.

Benefits of Filtering on TCP Flags

The ACL TCP Flags Filtering feature provides a flexible mechanism for filtering on TCP flags. Previously, an incoming packet was matched as long as any TCP flag in the packet matched a flag specified in the access control entry (ACE). This behavior allows for a security loophole, because packets with all flags set could get past the access control list (ACL). The ACL TCP Flags Filtering feature allows you to select any combination of flags on which to filter. The ability to match on a flag set and on a flag not set gives you a greater degree of control for filtering on TCP flags, thus enhancing security.

Because TCP packets can be sent as false synchronization packets that can be accepted by a listening port, it is recommended that administrators of firewall devices set up some filtering rules to drop false TCP packets.

The ACEs that make up an access list can be configured to detect and drop unauthorized TCP packets by allowing only the packets that have a very specific group of TCP flags set or not set. The ACL TCP Flags Filtering feature provides a greater degree of packet-filtering control in the following ways:

- You can select any desired combination of TCP flags on which to filter TCP packets.
- You can configure ACEs to allow matching on a flag that is set, as well as on a flag that is not set.

TCP Flags

The table below lists the TCP flags, which are further described in RFC 793, *Transmission Control Protocol*.

Table 2: TCP Flags

TCP Flag	Purpose
ACK	Acknowledge flag—Indicates that the acknowledgment field of a segment specifies the next sequence number the sender of this segment is expecting to receive.
FIN	Finish flag—Used to clear connections.
PSH	Push flag—Indicates the data in the call should be immediately pushed through to the receiving user.
RST	Reset flag—Indicates that the receiver should delete the connection without further interaction.
SYN	Synchronize flag—Used to establish connections.
URG	Urgent flag—Indicates that the urgent field is meaningful and must be added to the segment sequence number.

Benefits of Using the Named ACL Support for Noncontiguous Ports on an Access Control Entry Feature

This feature greatly reduces the number of access control entries (ACEs) required in an access control list to handle multiple entries for the same source address, destination address, and protocol. If you maintain large numbers of ACEs, use this feature to consolidate existing groups of access list entries wherever it is possible and when you create new access list entries. When you configure access list entries with noncontiguous ports, you will have fewer access list entries to maintain.

How to Create an IP Access List to Filter IP Options TCP Flags Noncontiguous Ports

Filtering Packets That Contain IP Options

Complete these steps to configure an access list to filter packets that contain IP options and to verify that the access list has been configured correctly.

Note

- The ACL Support for Filtering IP Options feature can be used only with named, extended ACLs.
- Resource Reservation Protocol (RSVP) Multiprotocol Label Switching Traffic Engineering (MPLS TE), Internet Group Management Protocol Version 2 (IGMPV2), and other protocols that use IP options packets may not function in drop or ignore mode if this feature is configured.
- On most Cisco devices, a packet with IP options is not switched in hardware, but requires control plane software processing (primarily because there is a need to process the options and rewrite the IP header), so all IP packets with IP options will be filtered and switched in software.

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **ip access-list extended** *access-list-name*

Example:

```
Device(config)# ip access-list extended mylist1
```

Specifies the IP access list by name and enters named access list configuration mode.

Step 4 [*sequence-number*] **deny** *protocol source source-wildcard destination destination-wildcard* [**option** *option-value*] [**precedence** *precedence*] [**tos** *tos*] [**time-range** *time-range-name*] [**fragments**]

Example:

```
Device(config-ext-nacl)# deny ip any any option traceroute
```

(Optional) Specifies a **deny** statement in named IP access list mode.

- This access list happens to use a **deny**statement first, but a **permit** statement could appear first, depending on the order of statements you need.

- Use the **option** keyword and *option-value* argument to filter packets that contain a particular IP Option.
- In this example, any packet that contains the traceroute IP option will be filtered out.
- Use the **no** *sequence-number* form of this command to delete an entry.

Step 5 [*sequence-number*] **permit** *protocol source source-wildcard destination destination-wildcard* [**option** *option-value*] [**precedence** *precedence*] [**tos** *tos*] [**time-range** *time-range-name*] [**fragments**]

Example:

```
Device(config-ext-nacl)# permit ip any any option security
```

Specifies a **permit** statement in named IP access list mode.

- In this example, any packet (not already filtered) that contains the security IP option will be permitted.
- Use the **no** *sequence-number* form of this command to delete an entry.

Step 6 Repeat Step 4 or Step 5 as necessary.

Allows you to revise the access list.

Step 7 **end**

Example:

```
Device(config-ext-nacl)# end
```

(Optional) Exits named access list configuration mode and returns to privileged EXEC mode.

Step 8 **show ip access-lists** *access-list-name*

Example:

```
Device# show ip access-lists mylist1
```

(Optional) Displays the contents of the IP access list.

What to Do Next

Apply the access list to an interface or reference it from a command that accepts an access list.

Note To effectively eliminate all packets that contain IP Options, we recommend that you configure the global **ip options drop** command.

Filtering Packets That Contain TCP Flags

This task configures an access list to filter packets that contain TCP flags and verifies that the access list has been configured correctly.

Note

- TCP flag filtering can be used only with named, extended ACLs.
- The ACL TCP Flags Filtering feature is supported only for Cisco ACLs.
- Previously, the following command-line interface (CLI) format could be used to configure a TCP flag-checking mechanism:

permit tcp any any rst The following format that represents the same ACE can now be used: **permit tcp any any match-any +rst** Both the CLI formats are accepted; however, if the new keywords **match-all** or **match-any** are chosen, they must be followed by the new flags that are prefixed with "+" or "-". It is advisable to use only the old format or the new format in a single ACL. You cannot mix and match the old and new CLI formats.

Caution If a device having ACEs with the new syntax format is reloaded with a previous version of the Cisco software that does not support the ACL TCP Flags Filtering feature, the ACEs will not be applied, leading to possible security loopholes.

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **ip access-list extended** *access-list-name*

Example:

```
Device(config)# ip access-list extended kmd1
```

Specifies the IP access list by name and enters named access list configuration mode.

Step 4 [*sequence-number*] **permit tcp** *source source-wildcard* [*operator* [*port*]] *destination destination-wildcard* [*operator* [*port*]] [**established**|{**match-any** | **match-all**} {+ | -} *flag-name*] [**precedence** *precedence*] [**tos** *tos*] [**time-range** *time-range-name*] [**fragments**]

Example:

```
Device(config-ext-nacl)# permit tcp any any match-any +rst
```

Specifies a **permit** statement in named IP access list mode.

- This access list happens to use a **permit**statement first, but a **deny** statement could appear first, depending on the order of statements you need.
- Use the TCP command syntax of the **permit**command.
- Any packet with the RST TCP header flag set will be matched and allowed to pass the named access list kmd1 in Step 3.

Step 5 [*sequence-number*] **deny tcp** *source source-wildcard* [*operator* [*port*]] *destination destination-wildcard* [*operator* [*port*]] [**established**|{**match-any** | **match-all**} {**+** | **-**} *flag-name*] [**precedence** *precedence*] [**tos** *tos*] [**time-range** *time-range-name*] [**fragments**]

Example:

```
Device(config-ext-nacl)# deny tcp any any match-all -ack -fin
```

(Optional) Specifies a **deny** statement in named IP access list mode.

- This access list happens to use a **permit**statement first, but a **deny** statement could appear first, depending on the order of statements you need.
- Use the TCP command syntax of the **deny**command.
- Any packet that does not have the ACK flag set, and also does not have the FIN flag set, will not be allowed to pass the named access list kmd1 in Step 3.
- See the **deny**(IP) command for additional command syntax to permit upper-layer protocols (ICMP, IGMP, TCP, and UDP).

Step 6 Repeat Step 4 or Step 5 as necessary, adding statements by sequence number where you planned. Use the **no** *sequence-number*command to delete an entry.

Allows you to revise the access list.

Step 7 **end**

Example:

```
Device(config-ext-nacl)# end
```

(Optional) Exits the configuration mode and returns to privileged EXEC mode.

Step 8 **show ip access-lists** *access-list-name*

Example:

```
Device# show ip access-lists kmd1
```

(Optional) Displays the contents of the IP access list.

- Review the output to confirm that the access list includes the new entry.

Configuring an Access Control Entry with Noncontiguous Ports

Perform this task to create access list entries that use noncontiguous TCP or UDP port numbers. Although this task uses TCP ports, you could use the UDP syntax of the **permit** and **deny** commands to filter noncontiguous UDP ports.

Although this task uses a **permit** command first, use the **permit** and **deny** commands in the order that achieves your filtering goals.

Note The ACL—Named ACL Support for Noncontiguous Ports on an Access Control Entry feature can be used only with named, extended ACLs.

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **ip access-list extended** *access-list-name*

Example:

```
Device(config)# ip access-list extended acl-extd-1
```

Specifies the IP access list by name and enters named access list configuration mode.

Step 4 [*sequence-number*] **permit tcp** *source source-wildcard* [*operator port* [*port*]] *destination destination-wildcard* [*operator* [*port*]] [**established** {**match-any** | **match-all**} {**+** | **-**} *flag-name*] [**precedence** *precedence*] [**tos** *tos*] [**time-range** *time-range-name*] [**fragments**]

Example:

```
Device(config-ext-nacl)# permit tcp any eq telnet ftp any eq 450 679
```

Specifies a **permit** statement in named IP access list configuration mode.

- Operators include **lt** (less than), **gt** (greater than), **eq** (equal), **neq** (not equal), and **range** (inclusive range).
- If the operator is positioned after the source and source-wildcard arguments, it must match the source port. If the operator is positioned after the destination and destination-wildcard arguments, it must match the destination port.
- The **range** operator requires two port numbers. You can configure up to 10 ports after the **eq** and **neq**operators. All other operators require one port number.
- To filter UDP ports, use the UDP syntax of this command.

Step 5 [*sequence-number*] **deny tcp** *source source-wildcard* [*operator port* [*port*]] *destination destination-wildcard* [*operator* [*port*]] [**established** {**match-any** | **match-all**} {+ | -} *flag-name*] [**precedence** *precedence*] [**tos** *tos*] [**time-range** *time-range-name*] [**fragments**]

Example:

```
Device(config-ext-nacl)# deny tcp any neq 45 565 632 any
```

(Optional) Specifies a **deny** statement in named access list configuration mode.

- Operators include **lt** (less than), **gt** (greater than), **eq** (equal), **neq** (not equal), and **range** (inclusive range).
- If the *operator* is positioned after the *source* and *source-wildcard* arguments, it must match the source port. If the *operator* is positioned after the *destination* and *destination-wildcard* arguments, it must match the destination port.
- The **range** operator requires two port numbers. You can configure up to 10 ports after the **eq** and **neq**operators. All other operators require one port number.
- To filter UDP ports, use the UDP syntax of this command.

Step 6 Repeat Step 4 or Step 5 as necessary, adding statements by sequence number where you planned. Use the **no** *sequence-number* command to delete an entry.

Allows you to revise the access list.

Step 7 **end**

Example:

```
Device(config-ext-nacl)# end
```

(Optional) Exits named access list configuration mode and returns to privileged EXEC mode.

Step 8 **show ip access-lists** *access-list-name*

Example:

```
Device# show ip access-lists kmd1
```

(Optional) Displays the contents of the access list.

Consolidating Access List Entries with Noncontiguous Ports into One Access List Entry

Perform this task to consolidate a group of access list entries with noncontiguous ports into one access list entry.

Although this task uses TCP ports, you could use the UDP syntax of the **permit** and **deny** commands to filter noncontiguous UDP ports.

Although this task uses a **permit** command first, use the **permit** and **deny** commands in the order that achieves your filtering goals.

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **show ip access-lists** *access-list-name*

Example:

```
Device# show ip access-lists mylist1
```

(Optional) Displays the contents of the IP access list.

- Review the output to see if you can consolidate any access list entries.

Step 3 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 4 **ip access-list extended** *access-list-name*

Example:

```
Device(config)# ip access-list extended mylist1
```

Specifies the IP access list by name and enters named access list configuration mode.

Step 5 **no** [*sequence-number*] **permit** *protocol source source-wildcard destination destination-wildcard*[**option** *option-name*] [**precedence** *precedence*][**tos** *tos*] [**time-range** *time-range-name*] [**fragments**]

Example:

```
Device(config-ext-nacl)# no 10
```

Removes the redundant access list entry that can be consolidated.

- Repeat this step to remove entries to be consolidated because only the port numbers differ.
- After this step is repeated to remove the access list entries 20, 30, and 40, for example, those entries are removed because they will be consolidated into one **permit** statement.
- If a *sequence-number* is specified, the rest of the command syntax is optional.

Step 6 [*sequence-number*] **permit** *protocol source source-wildcard*[*operator port*[*port*]] *destination destination-wildcard*[*operator port*[*port*]] [**option** *option-name*] [**precedence** *precedence*][**tos** *tos*] [**time-range** *time-range-name*] [**fragments**]

Example:

```
Device(config-ext-nacl)# permit tcp any neq 45 565 632 any eq 23 45 34 43
```

Specifies a **permit** statement in named access list configuration mode.

- In this instance, a group of access list entries with noncontiguous ports was consolidated into one **permit** statement.
- You can configure up to 10 ports after the **eq** and **neq** operators.

Step 7 Repeat Steps 5 and 6 as necessary, adding **permit** or **deny** statements to consolidate access list entries where possible. Use the **no** *sequence-number* command to delete an entry.

Allows you to revise the access list.

Step 8 **end**

Example:

```
Device(config-std-nacl)# end
```

(Optional) Exits named access list configuration mode and returns to privileged EXEC mode.

Step 9 **show ip access-lists** *access-list-name*

Example:

```
Device# show ip access-lists mylist1
```

(Optional) Displays the contents of the access list.

What To Do Next

Apply the access list to an interface or reference it from a command that accepts an access list.

Configuration Examples for Filtering IP Options, TCP Flags, Noncontiguous Ports

Example: Filtering Packets That Contain IP Options

The following example shows an extended access list named mylist2 that contains access list entries (ACEs) that are configured to permit TCP packets only if they contain the IP Options that are specified in the ACEs:

```
ip access-list extended mylist2
 10 permit ip any any option eool
 20 permit ip any any option record-route
 30 permit ip any any option zsu
 40 permit ip any any option mtup
```

The **show access-list** command has been entered to show how many packets were matched and therefore permitted:

```
Device# show ip access-list mylist2
Extended IP access list test
10 permit ip any any option eool (1 match)
20 permit ip any any option record-route (1 match)
30 permit ip any any option zsu (1 match)
40 permit ip any any option mtup (1 match)
```

Example: Filtering Packets That Contain TCP Flags

The following access list allows TCP packets only if the TCP flags ACK and SYN are set and the FIN flag is not set:

```
ip access-list extended aaa
 permit tcp any any match-all +ack +syn -fin
 end
```

The **show access-list** command has been entered to display the ACL:

```
Device# show access-list aaa

Extended IP access list aaa
 10 permit tcp any any match-all +ack +syn -fin
```

Example: Creating an Access List Entry with Noncontiguous Ports

The following access list entry can be created because up to ten ports can be entered after the **eq** and **neq** operators:

```
ip access-list extended aaa
 permit tcp any eq telnet ftp any eq 23 45 34
 end
```

Enter the **show access-lists** command to display the newly created access list entry.

```
Device# show access-lists aaa

Extended IP access list aaa
 10 permit tcp any eq telnet ftp any eq 23 45 34
```

Example: Consolidating Some Existing Access List Entries into One Access List Entry with Noncontiguous Ports

The **show access-lists** command is used to display a group of access list entries for the access list named abc:

```
Device# show access-lists abc
Extended IP access list abc
 10 permit tcp any eq telnet any eq 450
 20 permit tcp any eq telnet any eq 679
 30 permit tcp any eq ftp any eq 450
 40 permit tcp any eq ftp any eq 679
```

Because the entries are all for the same **permit** statement and simply show different ports, they can be consolidated into one new access list entry. The following example shows the removal of the redundant access list entries and the creation of a new access list entry that consolidates the previously displayed group of access list entries:

```
ip access-list extended abc
 no 10
 no 20
 no 30
 no 40
 permit tcp any eq telnet ftp any eq 450 679
 end
```

When the **show access-lists** command is reentered, the consolidated access list entry is displayed:

```
Device# show access-lists abc
Extended IP access list abc
 10 permit tcp any eq telnet ftp any eq 450 679
```

Additional References

Related Documents

Related Topic	Document Title
Security commands	*Cisco IOS Security Command Reference*
Configuring the device to drop or ignore packets containing IP Options by using the **no ip options** command.	*ACL IP Options Selective Drop*
Overview information about access lists.	*IP Access List Overview*
Information about creating an IP access list and applying it to an interface	*Creating an IP Access List and Applying It to an Interface*
QoS commands	*Cisco IOS Quality of Service Solutions Command Reference*

RFCs

RFC	Title
RFC 791	*Internet Protocol* http://www.faqs.org/rfcs/rfc791.html
RFC 793	*Transmission Control Protocol*
RFC 1393	*Traceroute Using an IP Option*

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

CHAPTER 4

MAC Access Control Lists

This chapter describes how to configure MAC access control lists (ACLs) on a Cisco router. It contains the following sections:

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to https://cfnng.cisco.com/. An account on Cisco.com is not required.

Prerequisites for MAC Access Control Lists

- You must be familiar with MAC addressing and non-IP protocols to configure MAC ACLs.

Restrictions for MAC Access Control Lists

- MAC ACL is supported only on EFP or TEFP.
- MAC ACL is not supported for IP packets.
- MAC ACL counters are not supported.
- MAC ACLs are not supported on port, routed interface, and BDI.

- ACL and QoS can be applied on the same EFP.
- Outbound MAC ACL is not supported.

Information About MAC Access Control Lists

MAC Access Control Lists

MAC ACLs are ACLs that filter traffic using information in the Layer 2 header of each packet. You can use ACLs to control which hosts can access different parts of a network or to decide which types of traffic are forwarded or blocked at the router interfaces. MAC ACL is supported on EFP and Cross-Connect.

How to Configure MAC Access Control Lists

Configuring ACL

To configure ACL, perform the steps below.

Step 1 **enable**

Example:

```
Router> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Router# configure terminal
```

Enters global configuration mode.

Step 3 **mac access-list extended** *name*

Example:

```
Router(config)# mac access-list ext macext2
```

Creates an extended MAC access control list (ACL) and define its access control entries (ACEs).

- *name*—Name of the ACL to which the entry belongs.

Step 4 {**permit** | **deny**} {**any** | **host** *src-MAC-addr*} {**any** | **host** *dst-MAC-addr*}

Example:

```
Router(config-ext-macl)# deny any any
```

Permits or denies Layer 2 traffic to be forwarded if the conditions are matched.

- **permit**—Permits Layer 2 traffic to be forwarded if the conditions are matched.

- **deny**—Denies Layer 2 traffic to be forwarded if the conditions are matched.
- **any**—Keyword to deny any source or destination MAC address.
- **host** *src-MAC-addr*—Defines a host MAC address. MAC address-based subnets are not allowed.
- **host** dst-MAC-addr—Defines a destination MAC address. MAC address-based subnets are not allowed.

Step 5 **end**

Example:

```
Router(config-ext-macl)# end
```

Returns to privileged EXEC mode.

Verifying MAC Access Control Lists

To verify the MAC ACL configuration, use the following **show** command.

- **show access-lists** *name*—Displays information about the named access list.

```
Router# show access-list macext4
```

```
Extended MAC access list macext4
    permit any host 0000.0000.0009
    permit any host 0000.0000.0010
    permit any host 0000.0000.0011
    permit any host 0000.0000.0012
```

Configuration Examples for MAC Access Control Lists

MAC ACL Configuration

Example: Allowing Specified Source or Destination MAC Addresses

```
(config)#mac access-list extended macext5
(config-ext-macl)#permit any host 0000.0000.0009
(config-ext-macl)#permit any host 0000.0000.0010
(config-ext-macl)#permit any host 0000.0000.0011
(config-ext-macl)#permit any host 0000.0000.0012
```

Example: Allowing any Source or Destination MAC Address

```
(config)#mac access-list extended macext9
(config-ext-macl)#permit any any
```

Additional References for MAC Access Control Lists

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases

Standards and RFCs

Standard/RFC	Title
Standard	—

MIBs

MIB	MIBs Link
• CISCO-MIB	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

CHAPTER 5

Configuring Storm Control

This document describes how to configure Storm Control on the router.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to https://cfnng.cisco.com/. An account on Cisco.com is not required.

Prerequisites for Storm Control

- Port-level storm control should be configured on EVC interfaces.
- Storm control threshold value should be configured as CIR (bps, kbps, %).
- Applicable to all types of Storms (unicast, broadcast, and multicast).

Restrictions for Storm control

- Storm control is only enabled for ports with EVC configurations.
- Storm control is specific to the Layer2 physical interfaces and port-channels; It is *not* supported on the Layer 3 interfaces or BDI.

- Strom control on port-channel interface is *not* supported.
- Storm control is supported only for unknown unicast, broadcast, and unknown multicast ingress traffic; It is *not* supported for egress traffic.
- Port-level storm control is supported on the router. EFP-level storm control is *not* supported.
- Strom control on local connect and cross-connect is *not* supported.

Information on Storm Control

A storm occurs when huge amount of broadcast, multicast, or unicast packets flood the LAN, creating excessive traffic and degrading network performance. Errors in the protocol-stack implementation or in the network configuration can also cause a storm. The mechanism to prevent and control such events is known as storm control.

Storm Control monitors incoming traffic levels over a 1-second traffic storm control interval and, during the interval compares the traffic level with the traffic storm control level configured. The traffic storm control threshold level is a percentage of the total available bandwidth of the port. Each port has different storm control levels for broadcast, multicast, and unicast type of traffic.

Storm control uses rising and falling thresholds to block and then restore the forwarding of broadcast, unicast, or multicast packets.

- The rising threshold is the traffic limit after which, that particular traffic is blocked.
- The falling threshold is the traffic limit below which, that particular starts forwarding again, if it was already blocked.

Note If a particular type of ingress traffic (unicast, broadcast and multicast) is more than the rising threshold configured on it, the interface goes to blocked state for that particular traffic.

Storm control prevents traffic on a LAN from being disrupted by a broadcast, multicast, or unicast storm on a port. Storm control is applicable for physical interfaces and is used to restrict the unicast, broadcast and multicast ingress traffic on the Layer2 interfaces. The feature is disabled by default on the router.

Configuring Storm Control

Before you begin:

- Configure the ports with EVC configuration.

Note To disable Storm Control feature, use the **no storm-control** command.

To configure storm control:

- Enables privileged EXEC mode. Enter your password if prompted.

```
Router> enable
```

- Enters global configuration mode.

```
Router# configure terminal
```

- Specifies an interface type and enters interface configuration mode.

```
Router(config)# interface gigabitethernet 0/0/0
```

- Specifies the global broadcast, multicast, or unicast storm control suppression level.
  - **broadcast**—Configure broadcast storm control.
  - **multicast**—Configure multicast storm control.
  - **unicast**—Configure unknown unicast storm control.
  - **level**—Specifies the threshold levels for broadcast, multicast, or unicast traffic.
  - *rising_threshold*—Upper threshold level.
  - *falling_threshold*—Lower threshold level.
  - **bps**—Specifies the suppression level in bits per second.
  - **pps**—Specifies the suppression level in packets per second.

```
Router(config)# storm-control broadcast level 1 .50
```

- Specifies the action to take when a storm occurs on a port :
  - **shutdown**—Disables the port during a storm. The **shutdown** action sets the port to shut state during a storm. The port remains in shutdown state until recovered by giving a **no shutdown** command when the storm goes below the configured lower threshold .
  - **trap**—Sends an SNMP trap. The **trap** action generates an SNMP trap when a storm is detected . The default is to restrict the particular ingress traffic and not to send out traps.

    ```
    Router(config)# storm-control action trap
    ```

- Exits interface configuration mode and returns the router to global configuration mode.

```
Router(config)# exit
```

Configuration Example:

```
interface GigabitEthernet0/0/1
 no ip address
 negotiation auto
 storm-control broadcast level bps 50k 40k
 storm-control multicast level pps 100 90
 storm-control unicast level 1.00 0.50
service instance 1 ethernet
  encapsulation dot1q 2
  rewrite ingress tag pop 1 symmetric
  bridge-domain 1
!
```

Verifying Storm Control

- Use the **show storm-control** command to verify the Storm Control feature configuration.

```
Router# show storm-control

Interface   Type   Filter State   Upper        Lower        Current
---------  ------   -------------  -----------  -----------  ----------
Gi0/0/0    Bcast    Forwarding          0 pps        0 pps        0 pps
Gi0/0/0    Ucast    Forwarding      80.00%       20.00%       39.99%
Gi0/0/1    Bcast    Blocking          50k bps      40k bps  362.25k bps
Gi0/0/1    Mcast    Blocking          100 pps       90 pps      265 pps
Gi0/0/1    Ucast    Blocking         1.00%        0.50%        1.28%
```

Note With each current traffic rate under display method, the storm control blocks traffic on the interface and the current traffic rate will always show 0%.

- Use the **show storm-control** command to display the storm control configurations for the configured storm type.

```
Router# show storm-control broadcast
Interface   Type   Filter State    Upper        Lower        Current
---------  ------   -------------  -----------  -----------  ----------
Gi0/0/2    Bcast    Blocking         280k pps     260k pps  284.64k pps
Te0/0/12   Bcast    Blocking        2.29g bps       2g bps    2.49g bps
Te0/0/13   Bcast    Blocking         4.6m pps     4.3m pps    4.69m pps
Po4        Bcast    Link Down         45k pps      43k pps        0 pps
Po5        Bcast    Blocking         240k pps     235k pps  241.85k pps
Router#
Router#show storm-control unicast
Interface   Type   Filter State   Upper        Lower        Current
---------  ------   -------------  -----------  -----------  ----------
Gi0/0/2    Ucast    Blocking         290k pps     280k pps  298.19k pps
Te0/0/12   Ucast    Blocking           4m pps     3.5m pps    4.74m pps
Te0/0/13   Ucast    Blocking         4.5m pps     4.3m pps    4.91m pps
Po4        Ucast    Link Down         45k pps      43k pps        0 pps
Po5        Ucast    Blocking         250k pps     240k pps  253.36k pps
Router#
Router#show storm-control multicast
Interface   Type   Filter State   Upper        Lower        Current
---------  ------   -------------  -----------  -----------  ----------
Gi0/0/2    Mcast    Blocking         240m bps     220m bps   260.5m bps
Te0/0/12   Mcast    Blocking        2.25g bps       2g bps    2.38g bps
Te0/0/13   Mcast    Blocking           3g bps     2.5g bps     3.3g bps
Po4        Mcast    Link Down         45k pps      43k pps        0 pps
Po5        Mcast    Blocking         200k pps     190k pps  206.28k pps
Router#
```

- Use the **show storm-control GigabitEthernet** command to verify the Storm Control feature configuration at the interface.

```
Router # show storm control GigabitEthernet 0/0/1

Interface   Type   Filter State   Upper        Lower        Current
---------  ------   -------------  -----------  -----------  ----------
Gi0/0/1    Bcast    Blocking          50k bps      40k bps  362.25k bps
Gi0/0/1    Mcast    Blocking          100 pps       90 pps      265 pps
Gi0/0/1    Ucast    Blocking         1.00%        0.50%        1.28%
```

- Use the **show run interface** command to verify the action trap configured on the port.

```
Router# show run interface GigabitEthernet 0/0/1

Building configuration...
Current configuration : 300 bytes
!
interface GigabitEthernet0/0/1
 no ip address
 negotiation auto
storm-control broadcast level 9.00 7.00
 storm-control action trap
 service instance trunk 1 ethernet
  encapsulation dot1q 1-200
  rewrite ingress tag pop 1 symmetric
  bridge-domain from-encapsulation
 !
end
```

- The following example shows the **action trap** being sent when a storm is hit.

```
Router# show storm-control G 0/0/1
Interface   Type   Filter State   Upper        Lower        Current
---------  ------   -------------  -----------  -----------  ----------
Gi0/4/2    Bcast    Blocking         9.00%        7.00%       11.00%
May 29 14:46:28.008 IST: %STORM_CONTROL-3-TRAP: A packet storm was detected on Gi0/4/2.

Sending SNMP trap
```

- The following example shows the **action shutdown** configured.

```
Router# show run interface Gi0/0/1

Building configuration...
Current configuration : 300 bytes
!
interface GigabitEthernet0/0/1
 no ip address
 negotiation auto
storm-control broadcast level 9.00 7.00
 storm-control action shutdown
 service instance trunk 1 ethernet
  encapsulation dot1q 1-200
  rewrite ingress tag pop 1 symmetric
  bridge-domain from-encapsulation
 !
end
```

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/mcl/allreleasemcl/all-book.html

Standards and RFCs

Standard/RFC	Title
No specific Standards and RFCs are supported by the features in this document.	—

MIBs

MIB	MIBs Link
—	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html