



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, on page 1](#)
- [Prerequisites for Creating an IP Access List to Filter IP Options TCP Flags Noncontiguous Ports , on page 1](#)
- [Information About Creating an IP Access List to Filter IP Options, TCP Flags, Noncontiguous Ports , on page 2](#)
- [How to Create an IP Access List to Filter IP Options TCP Flags Noncontiguous Ports , on page 4](#)
- [Configuration Examples for Filtering IP Options, TCP Flags, Noncontiguous Ports , on page 11](#)
- [Additional References, on page 13](#)

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://cfng.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 1: 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] [log] [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] [log] [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 kmdl
```

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*] [**log**] [**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] [log] [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] [log] [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] [log] [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*] [**log**] [**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*] [**log**] [**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:

Example: Creating an Access List Entry with Noncontiguous Ports

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

RFCs

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

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

