Prerequisites for Implementing Access Lists and Prefix Lists

The following prerequisite applies to implementing access lists and prefix lists:

All command task IDs are listed in individual command references and in the Cisco IOS XR Task ID Reference Guide.If you need assistance with your task group assignment, contact your system administrator.

Restrictions for Implementing Access Lists and Prefix Lists

The following restrictions apply to implementing access lists and prefix lists:

- Layer 2/Layer 3 ACLs are not supported on Layer 2 interfaces.
- Object group ACLs are not supported.
- IPv4 ACLs are not supported for loopback and interflex interfaces.
- IPv6 ACLs are not supported for loopback, interflex and L2 Ethernet Flow Point (EFP) main or subinterfaces.
- IPv6 ACL configuration on bundle interfaces (Ethernet LAG bundles only) is not supported.

- If the TCAM utilization is high and large ACLs are modified, then an error may occur. During such instances, do the following to edit an ACL:

  Note

  1. Remove the ACL from the interface.
  2. Reconfigure the ACL.
  3. Reapply the ACL to the interface.

  Use the **show prm server tcam summary all acl all location** and **show pfilter-ea fea summary location** commands to view the TCAM utilization.

- Filtering of MPLS packets through common ACL and interface ACL is not supported.
- Video Monitoring is not supported through ACLs on IPv6 interfaces.
- You can configure an ACL name with a maximum of 64 characters.
- You can configure an ACL name to comprise of only letters and numbers.

Restrictions for Implementing ACL-based Forwarding

The following restrictions apply to implementing ACL-based forwarding (ABF):

- No support for IPv4 multicast traffic.
- No support for ACL-based forwarding from a software switching path (for example, IPv4 option packets).
- Support is only on physical interfaces, subinterfaces, and bundles.
- ACL-based forwarding is an ingress-only feature.
- When optional parameter interface-statistics is provided, the system pre-allocates a fixed number of contiguous statistic entries per ACE. The maximum number of contiguous statistic entries with unique counters for each ACL, of each direction of each line card, can be up to eight entries.

Information About Implementing Access Lists and Prefix Lists

To implement access lists and prefix lists, you must understand the following concepts:

Access Lists and Prefix Lists Feature Highlights

This section lists the feature highlights for access lists and prefix lists.

- Cisco IOS XR software provides the ability to clear counters for an access list or prefix list using a specific sequence number.
- Cisco IOS XR software provides the ability to copy the contents of an existing access list or prefix list to another access list or prefix list.

- Cisco IOS XR software allows users to apply sequence numbers to permit or deny statements and to resequence, add, or remove such statements from a named access list or prefix list.

Note Resequencing is only for IPv4 prefix lists.

- Cisco IOS XR software does not differentiate between standard and extended access lists. Standard access list support is provided for backward compatibility.
- Cisco IOS XR software provides ACL-based forwarding to forward packets to a next-hop that is specified by the ACL rule.
- Atomic update is supported for both ACL and ACL-based forwarding.

Purpose of IP Access Lists

Access lists perform packet filtering to control which packets move through the network and where. Such controls help to limit network traffic and restrict the access of users and devices to the network. Access lists have many uses, and therefore many commands accept a reference to an access list in their command syntax. Access lists can be used to do the following:

- Filter incoming packets on an interface.
- Filter outgoing packets on an interface.
- Restrict the contents of routing updates.
- Limit debug output based on an address or protocol.
- Control vty access.
- Identify or classify traffic for advanced features, such as congestion avoidance, congestion management, and priority and custom queueing.

How an IP Access List Works

An access list is a sequential list consisting of permit and deny statements that apply to IP addresses and possibly upper-layer IP protocols. The access list has a name by which it is referenced. Many software commands accept an access list as part of their syntax.

An access list can be configured and named, but it is not in effect until the access list is referenced by a command that accepts an access list. Multiple commands can reference the same access list. An access list can control traffic arriving at the router or leaving the router, but not traffic originating at the router. Note that, traffic such as SSH, ICMP and telnet traffic are blocked by ACL, in spite of being originated from the router. This is because, those packets are not injected as high priority packets, and hence get subjected to ACL processing. At the same time, BGP traffic bypasses the ACL applied on the interface, as it is a control packet which is injected as a critical inject packet from RSP or LC. Such packets are handled in the system with high priority and do not get dropped.

IP Access List Process and Rules

Use the following process and rules when configuring an IP access list:

- The software tests the source or destination address or the protocol of each packet being filtered against the conditions in the access list, one condition (permit or deny statement) at a time.
- 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 remaining 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 the address or protocol, the software discards the packet and returns an Internet Control Message Protocol (ICMP) Host Unreachable message. ICMP is configurable in the Cisco IOS XR software.
- If no conditions match, the software drops the packet because each access list ends with an unwritten or implicit deny statement. That is, if the packet has not been permitted or denied by the time it was tested against each statement, it is denied.
- The access list should contain at least one permit statement or else all packets are denied.
- Because the software stops testing conditions after the first match, the order of the conditions is critical. The same permit or deny statements specified in a different order could result in a packet being passed under one circumstance and denied in another circumstance.
- Only one access list per interface, per protocol, per direction is allowed.
- Inbound access lists process packets arriving at the router. Incoming packets are processed before being routed to an outbound interface. An inbound access list is efficient because it saves the overhead of routing lookups if the packet is to be discarded because it is denied by the filtering tests. If the packet is permitted by the tests, it is then processed for routing. For inbound lists, permit means continue to process the packet after receiving it on an inbound interface; **deny** means discard the packet.
- Outbound access lists process packets before they leave the router. Incoming packets are routed to the outbound interface and then processed through the outbound access list. For outbound lists, permit means send it to the output buffer; deny means discard the packet.
- An access list can not be removed if that access list is being applied by an access group in use. To remove an access list, remove the access group that is referencing the access list and then remove the access list.
- An access list must exist before you can use the **ipv4 access group** command.

Helpful Hints for Creating IP Access Lists

Consider the following when creating an IP access list:

- Create the access list before applying it to an interface. An interface to which an empty access list is applied permits all traffic.
- If you applied a nonexistent access list to an interface and then proceed to configure the access list, the first statement is placed into effect, and the the implicit deny statement that follows could cause all other traffic that needs to be permitted on the interface to be dropped, until you configure statements allowing the dropped traffic to be permitted.
- Organize your access list so that more specific references in a network or subnet appear before more general ones.

- To make the purpose of individual statements more easily understood at a glance, you can write a helpful remark before or after any statement.

Source and Destination Addresses

Source address and destination addresses are two of the most typical fields in an IP packet on which to base an access list. Specify source addresses to control packets from certain networking devices or hosts. Specify destination addresses to control packets being sent to certain networking devices or hosts.

Wildcard Mask and Implicit Wildcard Mask

Address filtering uses wildcard masking to indicate whether the software checks or ignores 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, an administrator can select a single or several 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.
- A wildcard mask bit 1 means *ignore* that corresponding bit value.

You do not have to supply a wildcard mask with a source or destination address in an access list statement. If you use the **host** keyword, the software assumes a wildcard mask of 0.0.0.0.

Unlike subnet masks, which require contiguous bits indicating network and subnet to be ones, wildcard masks allow noncontiguous bits in the mask. For IPv6 access lists, only contiguous bits are supported.

You can also use CIDR format (/x) in place of wildcard bits. For example, the IPv4 address 1.2.3.4 0.255.255.255 corresponds to 1.2.3.4/8

Transport Layer Information

You can filter packets on the basis of transport layer information, such as whether the packet is a TCP, UDP, SCTP, ICMP, or IGMP packet.

IP Access List Entry Sequence Numbering

The ability to apply sequence numbers to IP access-list entries simplifies access list changes. Prior to this feature, there was no way to specify the position of an entry within an access list. If a user wanted to insert an entry (statement) in the middle of an existing list, all 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.

The IP Access List Entry Sequence Numbering feature allows users to add sequence numbers to access-list entries and resequence them. When you add a new entry, you choose 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.

Sequence Numbering Behavior

The following details the sequence numbering behavior:

- If entries with no sequence numbers are applied, the first entry is assigned a sequence number of 10, and successive entries are incremented by 10. The maximum configurable sequence number is 2147483643 for IPv4 and IPv6 entries. For other entries, the maximum configurable sequence number is 2147483646. If the generated sequence number exceeds this maximum number, the following message displays:

```
Exceeded maximum sequence number.
```

- If you provide an entry without a sequence number, it is assigned a sequence number that is 10 greater than the last sequence number in that access list and is placed at the end of the list.
- ACL entries can be added without affecting traffic flow and hardware performance.
- If a new access list is entered from global configuration mode, then sequence numbers for that access list are generated automatically.
- Distributed support is provided so that the sequence numbers of entries in the route processor (RP) and line card (LC) are synchronized at all times.
- This feature works with named standard and extended IP access lists. Because the name of an access list can be designated as a number, numbers are acceptable.

Understanding IP Access List Logging Messages

Cisco IOS XR software can provide logging messages about packets permitted or denied by a standard IP access list. That is, any packet that matches the access list causes 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** command in global configuration mode.

Note ACL logging isn't supported for ingress MPLS packets

The first packet that triggers the access list causes an immediate logging message, and subsequent packets are collected over 5-minute intervals before they are displayed or logged.

However, you can use the { **ipv4 | ipv6** } **access-list log-update threshold** command to set the number of packets that, when they match an access list (and are permitted or denied), cause the system to generate a log message. You might do this to receive log messages more frequently than at 5-minute intervals.

Caution If you set the *update-number* argument to 1, a log message is sent right away, rather than caching it; every packet that matches an access list causes a log message. A setting of 1 isn't recommended because the volume of log messages could overwhelm the system.

Even if you use the { **ipv4 | ipv6} access-list log-update threshold** command, the 5-minute timer remains in effect, so each cache is emptied at the end of 5 minutes, regardless of the number 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's when a threshold isn't specified.

Note The logging facility might drop some logging message packets if there are too many to be handled or if more than one logging message is handled in 1 second. This behavior prevents the router from using excessive CPU cycles because of too many logging packets. Therefore, the logging facility shouldn't be used as a billing tool or as an accurate source of the number of matches to an access list.

Enable Logging on ACE

This section shows you how to enable the ACE of an ACL to log informational messages when it matches incoming packets, using the optional keyword **log**. The router supports this feature only for IPv4 or IPv6 ingress ACLs. 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.

```
Router#configure
Router(config)#ipv4 access-list test
Router(config-ipv4-acl)#10 permit udp 10.85.1.0 255.255.255.0 log
Router(config-ipv4-acl)#exit
Router(config)# interface FortyGigE0/0/0/22
Router(config-if)# ipv4 access-group test ingress
Router(config-if)# commit
```

Note Set log-level to **informational** or higher with the **logging console** command, so that the router displays the ACL log-messages on the console.

```
Router#configure
Router(config)#logging console informational
Router(config)# commit
```

For more information on log-levels, see section *Syslog Message Severity Levels* in the *Implementing System Logging* chapter of the *System Monitoring Configuration Guide for Cisco NCS 5500 Series Routers*.

The following snippet shows a sample log message:

```
Router: ipv4_acl_mgr[350]: %ACL-IPV4_ACL-6-IPACCESSLOGP : access-list test (10) permit udp

10.85.1.2(0) -> 10.0.0.1(0), 1 packet
```

Extended Access Lists with Fragment Control

In earlier releases, the non-fragmented packets and the initial fragments of a packet were processed by IP extended access lists (if you apply this access list), but non-initial fragments were permitted, by default. However, now, the IP Extended Access Lists with Fragment Control feature allows more granularity of control over non-initial fragments of a packet. Using this feature, you can specify whether the system examines non-initial IP fragments of packets when applying an IP extended access list.

As non-initial fragments contain only Layer 3 information, these access-list entries containing only Layer 3 information, can now be applied to non-initial fragments also. The fragment has all the information the system requires to filter, so the access-list entry is applied to the fragments of a packet.

This feature adds the optional **fragments** keyword to the following IP access list commands: **deny (IPv4), permit (IPv4)** , **deny (IPv6)** , **permit (IPv6).** By specifying the **fragments** keyword in an access-list entry,

that particular access-list entry applies only to non-initial fragments of packets; the fragment is either permitted or denied accordingly.

The behavior of access-list entries regarding the presence or absence of the **fragments** keyword can be summarized as follows:

If the Access-List Entry has...	Then...

...no **fragments** keyword and all of the access-list entry information matches,	For an access-list entry containing only Layer 3 information:
	• The entry is applied to non-fragmented packets, initial fragments, and non-initial fragments.
	
	For an access-list entry containing Layer 3 and Layer 4 information:
	• The entry is applied to non-fragmented packets and initial fragments.
	  • If the entry matches and is a permit statement, the packet or fragment is permitted.
	  • If the entry matches and is a deny statement, the packet or fragment is denied.
	• The entry is also applied to non-initial fragments in the following manner. Because non-initial fragments contain only Layer 3 information, only the Layer 3 portion of an access-list entry can be applied. If the Layer 3 portion of the access-list entry matches, and
	  • If the entry is a **permit** statement, the non-initial fragment is permitted.
	  • If the entry is a deny statement, the next access-list entry is processed.
	
	Note Note that the deny statements are handled differently for non-initial fragments versus non-fragmented or initial fragments.

...the **fragments** keyword and all of the access-list entry information matches,	The access-list entry is applied only to non-initial fragments. **Note** The **fragments** keyword cannot be configured for an access-list entry that contains any Layer 4 information.

You should not add the **fragments** keyword to every access-list entry, because the first fragment of the IP packet is considered a non-fragment and is treated independently of the subsequent fragments. Because an initial fragment will not match an access list permit or deny entry that contains the **fragments** keyword, the packet is compared to the next access list entry until it is either permitted or denied by an access list entry that does not contain the **fragments** keyword. Therefore, you may need two access list entries for every deny entry. The first deny entry of the pair will not include the **fragments** keyword, and applies to the initial fragment. The second deny entry of the pair will include the **fragments** keyword and applies to the subsequent fragments. In the cases where there are multiple **deny** access list entries for the same host but with different Layer 4 ports, a single deny access-list entry with the **fragments** keyword for that host is all that has to be added. Thus all the fragments of a packet are handled in the same manner by the access list.

Packet fragments of IP datagrams are considered individual packets and each fragment counts individually as a packet in access-list accounting and access-list violation counts.

Note The **fragments** keyword cannot solve all cases involving access lists and IP fragments.

Note Within the scope of ACL processing, Layer 3 information refers to fields located within the IPv4 header; for example, source, destination, protocol. Layer 4 information refers to other data contained beyond the IPv4 header; for example, source and destination ports for TCP or UDP, flags for TCP, type and code for ICMP.

Policy Routing

Fragmentation and the fragment control feature affect policy routing if the policy routing is based on the **match ip address** command and the access list had entries that match on Layer 4 through Layer 7 information. It is possible that noninitial fragments pass the access list and are policy routed, even if the first fragment was not policy routed or the reverse.

By using the **fragments** keyword in access-list entries as described earlier, a better match between the action taken for initial and noninitial fragments can be made and it is more likely policy routing will occur as intended.

Comments About Entries in Access Lists

You can include comments (remarks) about entries in any named IP access list using the **remark** access list configuration command. The remarks make the access list easier for the network administrator to understand and scan. Each remark line is limited to 255 characters.

The remark can go before or after a **permit** or **deny** statement. You should be consistent about where you put the remark so it is clear which remark describes which **permit** or **deny** statement. For example, it would be confusing to have some remarks *before* the associated **permit** or **deny** statements and some remarks *after* the associated statements. Remarks can be sequenced.

Remember to apply the access list to an interface or terminal line after the access list is created. See the"Applying Access Lists, on page 14" section for more information.

Access Control List Counters

In Cisco IOS XR software, ACL counters are maintained both in hardware and software. Hardware counters are used for packet filtering applications such as when an access group is applied on an interface. Software counters are used by all the applications mainly involving software packet processing.

Packet filtering makes use of 64-bit hardware counters per ACE. If the same access group is applied on interfaces that are on the same line card in a given direction, the hardware counters for the ACL are shared between two interfaces.

To display the hardware counters for a given access group, use the **show access-lists ipv4** [*access-list-name* **hardware** {**ingress** | **egress**} [**interface** *type interface-path-id*] {**location** *node-id*}] command in EXEC mode.

To clear the hardware counters, use the **clear access-list ipv4** *access-list-name* [**hardware** {**ingress** | **egress**} [**interface** *type interface-path-id*] {**location** *node-id*}] command in EXEC mode.

Hardware counting is not enabled by default for IPv4 ACLs because of a small performance penalty. To enable hardware counting, use the **ipv4 access-group** *access-list-name* {**ingress** | **egress**} [**hardware-count**] command in interface configuration mode. This command can be used as desired, and counting is enabled only on the specified interface.

Software counters are updated for the packets processed in software, for example, exception packets punted to the LC CPU for processing, or ACL used by routing protocols, and so on. The counters that are maintained are an aggregate of all the software applications using that ACL. To display software-only ACL counters, use the **show access-lists ipv4** *access-list-name* [**sequence** *number*] command in EXEC mode.

All the above information is true for IPv6, except that hardware counting is always enabled; there is no **hardware-count** option in the IPv6 access-group command-line interface (CLI).

BGP Filtering Using Prefix Lists

Prefix lists can be used as an alternative to access lists in many BGP route filtering commands. The advantages of using prefix lists are as follows:

- Significant performance improvement in loading and route lookup of large lists.
- Incremental updates are supported.
- More user friendly CLI. The CLI for using access lists to filter BGP updates is difficult to understand and use because it uses the packet filtering format.
- Greater flexibility.

Before using a prefix list in a command, you must set up a prefix list, and you may want to assign sequence numbers to the entries in the prefix list.

How the System Filters Traffic by Prefix List

Filtering by prefix list involves matching the prefixes of routes with those listed in the prefix list. When there is a match, the route is used. More specifically, whether a prefix is permitted or denied is based upon the following rules:

- An empty prefix list permits all prefixes.
- An implicit deny is assumed if a given prefix does not match any entries of a prefix list.
- When multiple entries of a prefix list match a given prefix, the longest, most specific match is chosen.

Sequence numbers are generated automatically unless you disable this automatic generation. If you disable the automatic generation of sequence numbers, you must specify the sequence number for each entry using the *sequence-number* argument of the **permit** and **deny** commands in either IPv4 or IPv6 prefix list configuration command. Use the **no** form of the **permit** or **deny** command with the *sequence-number* argument to remove a prefix-list entry.

The **show** commands include the sequence numbers in their output.

Information About Implementing ACL-based Forwarding

To implement access lists and prefix lists, you must understand the following concepts:

ACL-based Forwarding Overview

Converged networks carry voice, video and data. Users may need to route certain traffic through specific paths instead of using the paths computed by routing protocols. This is achieved by specifying the next-hop address in ACL configurations, so that the configured next-hop address from ACL is used for fowarding packet towards its destination instead of routing packet-based destination address lookup. This feature of using next-hop in ACL configurations for forwarding is called ACL Based Forwarding (ABF).

Traffic engineering over an IP or MPLS backbone can be done without MPLS-TE. The ability to divert certain kinds of traffic on top of routing allows you to let only voice traffic travel over certain links, while allowing data traffic to be sent using regular routing.

ACL-based forwarding enables you to choose service from multiple providers for broadcast TV over IP, IP telephony, data, and so on, which provides a cafeteria-like access to the Internet. Service providers can divert user traffic to various content providers.

ACL-based Forwarding Functions

ACL-based forwarding (ABF) enables you to configure filters for IPv4 packets. Each packet is based on the information from an IP source or destination address, TCP ports, precedence, DSCP, and so on. If a match occurs, ABF forwards the packet to one of the multiple next hops (up to three). ABF provides an alternative to regular routing by giving the ability to forward a next hop, based on packet content that extends beyond the destination IP address.

The ABF rule does not apply to “For Us” packets.

By implementing ABF, you can perform the following functions:

- Specify up to three next hops in the ACL rules.
- Forward IPv4 packets that are being forwarded on default routes to the next hop, as specified by the ACL rule.
- Use the existing ACL matching functionality to pick up the next-hop IP address that is based on the ACE configuration. The highest preferred, active, next-hop IP address—which is based on the ACE configuration—is chosen.
- Use the traditional destination IP address forwarding if the ABF next hops are not reachable.
- Use ABF as an ingress-only feature; it is not available for packets switched or originated by the software.
- Specify no rejection when both VRF and ABF configurations are applied on an interface. The ABF configuration is silently ignored by the forwarding software.

How to Implement Access Lists and Prefix Lists

This section contains the following procedures:

Configuring Extended Access Lists

This task configures an extended IPv4 or IPv6 access list.

SUMMARY STEPS

1. **configure**
2. {**ipv4** | **ipv6**} **access-list** *name*
3. [*sequence-number*] **remark** *remark*
4. Do one of the following:
   - [*sequence-number*]{**permit** | **deny**} *source source-wildcard destination destination-wildcard* [**precedence** *precedence*] [**dscp** *dscp*] [**fragments**] [**log** | **log-input**]
   - [*sequence-number*] {**permit** | **deny**} *protocol* {*source-ipv6-prefix*/*prefix-length* | **any** | **host** *source-ipv6-address*} [*operator* {*port* | *protocol-port*}] {*destination-ipv6-prefix*/*prefix-length* | **any** | **host** *destination-ipv6-address*} [*operator* {*port* | *protocol-port*}] [**dscp** *value*] [*routing*] [**authen**] [**destopts**] [**fragments**] [**log** | **log-input**]
5. Repeat Step 4 as necessary, adding statements by sequence number where you planned. Use the **no** *sequence-number* command to delete an entry.
6. **commit**
7. **show access-lists** {**ipv4** | **ipv6**} [*access-list-name* **hardware** {**ingress** | **egress**} [**interface** *type interface-path-id*] {**sequence** *number* | **location** *node-id*} | **summary** [*access-list-name*] | *access-list-name* [*sequence-number*] | **maximum** [**detail**] [**usage** {**pfilter location** *node-id*}]]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	

Step 2	{**ipv4** | **ipv6**} **access-list** *name*
	Example:
	`RP/0/RP0/CPU0:router(config)# ipv4 access-list acl_1`
	or
	`RP/0/RP0/CPU0:router(config)# ipv6 access-list acl_2`
		Enters either IPv4 or IPv6 access list configuration mode and configures the named access list.

Step 3	[*sequence-number*] **remark** *remark* **Example:** `RP/0/RP0/CPU0:router(config-ipv4-acl)# 10 remark Do not allow user1 to telnet out`	(Optional) Allows you to comment about a **permit** or **deny** statement in a named access list. • The remark can be up to 255 characters; anything longer is truncated. • Remarks can be configured before or after **permit** or **deny** statements, but their location should be consistent.
Step 4	Do one of the following: • [*sequence-number*]{**permit** | **deny**} *source source-wildcard destination destination-wildcard* [**precedence** *precedence*] [**dscp** *dscp*] [**fragments**] [**log** | **log-input**]	Specifies one or more conditions allowed or denied in IPv4 access list acl_1. • The optional **log** keyword causes an information logging message about the packet that matches the entry to be sent to the console.

	Command or Action	Purpose

	• [*sequence-number*] {**permit** | **deny**} *protocol* {*source-ipv6-prefix*/*prefix-length* | **any** | **host** *source-ipv6-address*} [*operator* {*port* | *protocol-port*}] {*destination-ipv6-prefix*/*prefix-length* | **any** | **host** *destination-ipv6-address*} [*operator* {*port* | *protocol-port*}] [**dscp** *value*] [*routing*] [**authen**] [**destopts**] [**fragments**] [**log** | **log-input**]
	Example:
	`RP/0/RP0/CPU0:router(config-ipv4-acl)# 10 permit 172.16.0.0 0.0.255.255`
	`RP/0/RP0/CPU0:router(config-ipv4-acl)# 20 deny 192.168.34.0 0.0.0.255`
	or
	`RP/0/RP0/CPU0:router(config-ipv6-acl)# 20 permit icmp any any`
	`RP/0/RP0/CPU0:router(config-ipv6-acl)# 30 deny tcp any any gt 5000`
		• The optional **log-input** keyword provides the same function as the **log** keyword, except that the logging message also includes the input interface.
		or
		Specifies one or more conditions allowed or denied in IPv6 access list acl_2.
		• Refer to the **deny** (IPv6) and **permit** (IPv6) commands for more information on filtering IPv6 traffic based on based on IPv6 option headers and optional, upper-layer protocol type information.
		Note Every IPv6 access list has an implicit **deny ipv6 any any** statement as its last match condition. An IPv6 access list must contain at least one entry for the implicit **deny ipv6 any any** statement to take effect.

Step 5	Repeat Step 4 as necessary, adding statements by sequence number where you planned. Use the **no** *sequence-number* command to delete an entry.	Allows you to revise an access list.
Step 6	**commit**	
Step 7	**show access-lists** {**ipv4** | **ipv6**} [*access-list-name* **hardware** {**ingress** | **egress**} [**interface** *type interface-path-id*] {**sequence** *number* | **location** *node-id*} | **summary** [*access-list-name*] | *access-list-name* [*sequence-number*] | **maximum** [**detail**] [**usage** {**pfilter location** *node-id*}]] **Example:** `RP/0/RP0/CPU0:router# show access-lists ipv4 acl_1`	(Optional) Displays the contents of current IPv4 or IPv6 access lists. • Use the *access-list-name* argument to display the contents of a specific access list. • Use the **hardware**, **ingress** or **egress**, and **location** or **sequence** keywords to display the access-list hardware contents and counters for all interfaces that use the specified access list in a given direction (ingress or egress). The access group for an interface must be configured using the **ipv4 access-group** command for access-list hardware counters to be enabled. • Use the **summary** keyword to display a summary of all current IPv4 or IPv6 access-lists. • Use the **interface** keyword to display interface statistics.

What to do next

After creating an access list, you must apply it to a line or interface. See the Applying Access Lists, on page 14 section for information about how to apply an access list.

ACL commit fails while adding and removing unique Access List Entries (ACE). This happens due to the absence of an assigned manager process. The user has to exit the config-ipv4-acl mode to configuration mode and re-enter the config-ipv4-acl mode before adding the first ACE.

Applying Access Lists

After you create an access list, you must reference the access list to make it work. Access lists can be applied on *either* outbound or inbound interfaces. This section describes guidelines on how to accomplish this task for both terminal lines and network interfaces.

Set identical restrictions on all the virtual terminal lines, because a user can attempt to connect to any of them.

For inbound access lists, after receiving a packet, Cisco IOS XR software checks the source address of the packet against the access list. If the access list permits the address, the software continues to process the packet. If the access list rejects the address, the software discards the packet and returns an ICMP host unreachable message. The ICMP message is configurable.

For outbound access lists, after receiving and routing a packet to a controlled interface, the software checks the source address of the packet against the access list. If the access list permits the address, the software sends the packet. If the access list rejects the address, the software discards the packet and returns an ICMP host unreachable message.

When you apply an access list that has not yet been defined to an interface, the software acts as if the access list has not been applied to the interface and accepts all packets. Note this behavior if you use undefined access lists as a means of security in your network.

Controlling Access to an Interface

This task applies an access list to an interface to restrict access to that interface.

Access lists can be applied on *either* outbound or inbound interfaces.

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-path-id*
3. Do one of the following:
   - **ipv4 access-group** *access-list-name* {**ingress** | **egress**} [**hardware-count**] [**interface-statistics**]
   - **ipv6 access-group** *access-list-name* {**ingress** | **egress**} [**interface-statistics**]
4. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config)# interface GigabitEthernet 0/2/0/2`	Configures an interface and enters interface configuration mode. • The *type* argument specifies an interface type. For more information on interface types, use the question mark (?) online help function.

	Command or Action	Purpose

		• The *instance* argument specifies either a physical interface instance or a virtual instance.
		  • The naming notation for a physical interface instance is *rack/slot/module/port*. The slash (/) between values is required as part of the notation.
		  • The number range for a virtual interface instance varies depending on the interface type.

Step 3	Do one of the following:
	• **ipv4 access-group** *access-list-name* {**ingress** | **egress**} [**hardware-count**] [**interface-statistics**]
	• **ipv6 access-group** *access-list-name* {**ingress** | **egress**} [**interface-statistics**]
	Example:
	`RP/0/RP0/CPU0:router(config-if)# ipv4 access-group p-in-filter in`
	`RP/0/RP0/CPU0:router(config-if)# ipv4 access-group p-out-filter out`
		Controls access to an interface.
		• Use the *access-list-name* argument to specify a particular IPv4 or IPv6 access list.
		• Use the **in** keyword to filter on inbound packets or the **out** keyword to filter on outbound packets.
		• Use the **hardware-count** keyword to enable hardware counters for the IPv4 access group.
		  • Hardware counters are automatically enabled for IPv6 access groups.
		• Use the **interface-statistics** keyword to specify per-interface statistics in the hardware.
		This example applies filters on packets inbound and outbound from GigabitEthernet interface 0/2/0/2.

Step 4	**commit**	

Controlling Access to a Line

This task applies an access list to a line to control access to that line.

SUMMARY STEPS

1. **configure**
2. **line** {**aux** | **console** | **default** | **template** *template-name*}
3. **access-class** *list-name*{**ingress** | **egress**}
4. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	

	Command or Action	Purpose

Step 2	**line** {**aux** | **console** | **default** | **template** *template-name*}
	Example:
	`RP/0/RP0/CPU0:router(config)# line default`
		Specifies either the auxiliary, console, default, or a user-defined line template and enters line template configuration mode.
		• Line templates are a collection of attributes used to configure and manage physical terminal line connections (the console and auxiliary ports) and vty connections. The following templates are available in Cisco IOS XR software:
		  • Aux line template—The line template that applies to the auxiliary line.
		  • Console line template— The line template that applies to the console line.
		  • Default line template—The default line template that applies to a physical and virtual terminal lines.
		  • User-defined line templates—User-defined line templates that can be applied to a range of virtual terminal lines.

Step 3	**access-class** *list-name*{**ingress** | **egress**} **Example:** `RP/0/RP0/CPU0:router(config-line)# access-class acl_2 out`	Restricts incoming and outgoing connections using an IPv4 or IPv6 access list. • In the example, outgoing connections for the default line template are filtered using the IPv6 access list acl_2.
Step 4	**commit**	

Configuring Prefix Lists

This task configures an IPv4 or IPv6 prefix list.

SUMMARY STEPS

1. **configure**
2. {**ipv4** | **ipv6**} **prefix-list** *name*
3. [*sequence-number*] **remark** *remark*
4. [*sequence-number*] {**permit** | **deny**} *network/length* [**ge** *value*] [**le** *value*] [**eq** *value*]
5. Repeat Step 4 as necessary. Use the **no** *sequence-number* command to delete an entry.
6. **commit**
7. Do one of the following:
   - **show prefix-list ipv4** [*name*] [*sequence-number*]
   - **show prefix-list ipv6** [*name*] [*sequence-number*] [*summary*]

8. **clear** {**ipv4** | **ipv6**} **prefix-list** *name* [*sequence-number*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	

Step 2	{**ipv4** | **ipv6**} **prefix-list** *name*
	Example:
	`RP/0/RP0/CPU0:router(config)# ipv4 prefix-list pfx_1`
	or
	`RP/0/RP0/CPU0:router(config)# ipv6 prefix-list pfx_2`
		Enters either IPv4 or IPv6 prefix list configuration mode and configures the named prefix list.
		• To create a prefix list, you must enter at least one **permit** or **deny** clause.
		• Use the **no** {**ipv4** | **ipv6**} **prefix-list** *name* command to remove all entries in a prefix list.

Step 3	[*sequence-number*] **remark** *remark*
	Example:
	`RP/0/RP0/CPU0:router(config-ipv4_pfx)# 10 remark Deny all routes with a prefix of 10/8`
	`RP/0/RP0/CPU0:router(config-ipv4_pfx)# 20 deny 10.0.0.0/8 le 32`
		(Optional) Allows you to comment about the following **permit** or **deny** statement in a named prefix list.
		• The remark can be up to 255 characters; anything longer is truncated.
		• Remarks can be configured before or after **permit** or **deny** statements, but their location should be consistent.

Step 4	[*sequence-number*] {**permit** | **deny**} *network/length* [**ge** *value*] [**le** *value*] [**eq** *value*] **Example:** `RP/0/RP0/CPU0:router(config-ipv6_pfx)# 20 deny 128.0.0.0/8 eq 24`	Specifies one or more conditions allowed or denied in the named prefix list. • This example denies all prefixes matching /24 in 128.0.0.0/8 in prefix list pfx_2.
Step 5	Repeat Step 4 as necessary. Use the **no** *sequence-number* command to delete an entry.	Allows you to revise a prefix list.
Step 6	**commit**	

Step 7	Do one of the following:
	• **show prefix-list ipv4** [*name*] [*sequence-number*]
	• **show prefix-list ipv6** [*name*] [*sequence-number*] [*summary*]
	Example:
	`RP/0/RP0/CPU0:router# show prefix-list ipv4 pfx_1`
	or
	`RP/0/RP0/CPU0:router# show prefix-list ipv6 pfx_2 summary`
		(Optional) Displays the contents of current IPv4 or IPv6 prefix lists.
		• Use the *name* argument to display the contents of a specific prefix list.
		• Use the *sequence-number* argument to specify the sequence number of the prefix-list entry.
		• Use the **summary** keyword to display summary output of prefix-list contents.

	Command or Action	Purpose
Step 8	**clear** {**ipv4** | **ipv6**} **prefix-list** *name* [*sequence-number*] **Example:** `RP/0/RP0/CPU0:router# clear prefix-list ipv4 pfx_1 30`	(Optional) Clears the hit count on an IPv4 or IPv6 prefix list. **Note** The *hit count* is a value indicating the number of matches to a specific prefix-list entry.

Configuring Standard Access Lists

This task configures a standard IPv4 access list.

Standard access lists use source addresses for matching operations.

SUMMARY STEPS

1. **configure**
2. **ipv4 access-list** *name*
3. [*sequence-number*] **remark** *remark*
4. [*sequence-number*] {**permit** | **deny**} *source* [*source-wildcard*] [**log** | **log-input**]
5. Repeat Step 4 as necessary, adding statements by sequence number where you planned. Use the **no** *sequence-number* command to delete an entry.
6. **commit**
7. **show access-lists** [**ipv4** | **ipv6**] [*access-list-name* **hardware** {**ingress** | **egress**} [**interface** *type interface-path-id*] {**sequence** *number* | **location** *node-id*} | **summary** [*access-list-name*] | *access-list-name* [*sequence-number*] | **maximum** [**detail**] [**usage** {**pfilter location** *node-id*}]]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**ipv4 access-list** *name* **Example:** `RP/0/RP0/CPU0:router# ipv4 access-list acl_1`	Enters IPv4 access list configuration mode and configures access list acl_1.
Step 3	[*sequence-number*] **remark** *remark* **Example:** `RP/0/RP0/CPU0:router(config-ipv4-acl)# 10 remark Do not allow user1 to telnet out`	(Optional) Allows you to comment about the following **permit** or **deny** statement in a named access list. • The remark can be up to 255 characters; anything longer is truncated. • Remarks can be configured before or after **permit** or **deny** statements, but their location should be consistent.
Step 4	[*sequence-number*] {**permit** | **deny**} *source* [*source-wildcard*] [**log** | **log-input**] **Example:**	Specifies one or more conditions allowed or denied, which determines whether the packet is passed or dropped.

	Command or Action	Purpose

	`RP/0/RP0/CPU0:router(config-ipv4-acl)# 20 permit 172.16.0.0 0.0.255.255`
	or
	`RRP/0/RP0/CPU0:routerrouter(config-ipv4-acl)# 30 deny 192.168.34.0 0.0.0.255`
		• Use the *source* argument to specify the number of network or host from which the packet is being sent.
		• Use the optional *source-wildcard* argument to specify the wildcard bits to be applied to the source.
		• The optional **log** keyword causes an information logging message about the packet that matches the entry to be sent to the console.
		• The optional **log-input** keyword provides the same function as the **log** keyword, except that the logging message also includes the input interface.

Step 5	Repeat Step 4 as necessary, adding statements by sequence number where you planned. Use the **no** *sequence-number* command to delete an entry.	Allows you to revise an access list.
Step 6	**commit**	
Step 7	**show access-lists** [**ipv4** | **ipv6**] [*access-list-name* **hardware** {**ingress** | **egress**} [**interface** *type interface-path-id*] {**sequence** *number* | **location** *node-id*} | **summary** [*access-list-name*] | *access-list-name* [*sequence-number*] | **maximum** [**detail**] [**usage** {**pfilter location** *node-id*}]] **Example:** `RP/0/RP0/CPU0:router# show access-lists ipv4 acl_1`	(Optional) Displays the contents of the named IPv4 access list. • The contents of an IPv4 standard access list are displayed in extended access-list format.

What to do next

After creating a standard access list, you must apply it to a line or interface. See the " Applying Access Lists, on page 14" section for information about how to apply an access list.

Copying Access Lists

This task copies an IPv4 or IPv6 access list.

SUMMARY STEPS

1. **copy access-list** {**ipv4** | **ipv6**}*source-acl destination-acl*
2. **show access-lists** {**ipv4** | **ipv6**}[*access-list-name* **hardware** {**ingress** | **egress**} [**interface** *type interface-path-id*] {**sequence** *number* | **location** *node-id*} | **summary** [*access-list-name*] | *access-list-name* [*sequence-number*] | **maximum** [**detail**] [**usage** {**pfilter location** *node-id*}]]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**copy access-list** {**ipv4** | **ipv6**}*source-acl destination-acl*	Creates a copy of an existing IPv4 or IPv6 access list.

	Command or Action	Purpose
	Example: `RP/0/RP0/CPU0:router# copy ipv6 access-list list-1 list-2`	• Use the *source-acl* argument to specify the name of the access list to be copied. • Use the *destination-acl* argument to specify where to copy the contents of the source access list. • The *destination-acl* argument must be a unique name; if the *destination-acl* argument name exists for an access list, the access list is not copied.
Step 2	**show access-lists** {**ipv4** | **ipv6**}[*access-list-name* **hardware** {**ingress** | **egress**} [**interface** *type interface-path-id*] {**sequence** *number* | **location** *node-id*} | **summary** [*access-list-name*] | *access-list-name* [*sequence-number*] | **maximum** [**detail**] [**usage** {**pfilter location** *node-id*}]] **Example:** `RP/0/RP0/CPU0:router# show access-lists ipv4 list-2`	(Optional) Displays the contents of a named IPv4 or IPv6 access list. For example, you can verify the output to see that the destination access list list-2 contains all the information from the source access list list-1.

Sequencing Access-List Entries and Revising the Access List

This task shows how to assign sequence numbers to entries in a named access list and how to add or delete an entry to or from an access list. It is assumed that a user wants to revise an access list. Resequencing an access list is optional.

SUMMARY STEPS

1. **resequence access-list** {**ipv4** | **ipv6**} *name* [*base* [*increment*]]
2. **configure**
3. {**ipv4** | **ipv6**} **access-list** *name*
4. Do one of the following:
   - [*sequence-number*] {**permit** | **deny**} *source source-wildcard destination destination-wildcard* [**precedence** *precedence*] [**dscp** *dscp*] [**fragments**] [**log** | **log-input**]
   - [*sequence-number*] {**permit** | **deny**} *protocol* {*source-ipv6-prefix/prefix-length* | **any** | **host** *source-ipv6-address*} [*operator* {*port* | *protocol-port*}] {*destination-ipv6-prefix/prefix-length* | **any** | **host** *destination-ipv6-address*} [*operator* {*port* | *protocol-port*}] [**dscp** *value*] [**routing**] [**authen**] [**destopts**] [**fragments**] [**log** | **log-input**]
5. Repeat Step 4 as necessary, adding statements by sequence number where you planned. Use the **no** *sequence-number* command to delete an entry.
6. **commit**
7. **show access-lists** [**ipv4** | **ipv6**] [*access-list-name* **hardware** {**ingress** | **egress**} [**interface** *type interface-path-id*] {**sequence** *number* | **location** *node-id*} | **summary** [*access-list-name*] | *access-list-name* [*sequence-number*] | **maximum** [**detail**] [**usage** {**pfilter location** *node-id*}]]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**resequence access-list** {**ipv4** | **ipv6**} *name* [*base* [*increment*]] **Example:** `RP/0/RP0/CPU0:router# resequence access-list ipv4 acl_3 20 15`	(Optional) Resequences the specified IPv4 or IPv6 access list using the starting sequence number and the increment of sequence numbers. • This example resequences an IPv4 access list named acl_3. The starting sequence number is 20 and the increment is 15. If you do not select an increment, the default increment 10 is used.
Step 2	**configure**	

Step 3	{**ipv4** | **ipv6**} **access-list** *name*
	Example:
	`RP/0/RP0/CPU0:router(config)# ipv4 access-list acl_1`
	or
	`RP/0/RP0/CPU0:router(config)# ipv6 access-list acl_2`
		Enters either IPv4 or IPv6 access list configuration mode and configures the named access list.

Step 4	Do one of the following:
	• [*sequence-number*] {**permit** | **deny**} *source source-wildcard destination destination-wildcard* [**precedence** *precedence*] [**dscp** *dscp*] [**fragments**] [**log** | **log-input**]
	• [*sequence-number*] {**permit** | **deny**} *protocol* {*source-ipv6-prefix/prefix-length* | **any** | **host** *source-ipv6-address*} [*operator* {*port* | *protocol-port*}] {*destination-ipv6-prefix/prefix-length* | **any** | **host** *destination-ipv6-address*} [*operator* {*port* | *protocol-port*}] [**dscp** *value*] [**routing**] [**authen**] [**destopts**] [**fragments**] [**log** | **log-input**]
	Example:
	`RP/0/RP0/CPU0:router(config-ipv4-acl)# 10 permit 172.16.0.0 0.0.255.255`
	`RP/0/RP0/CPU0:router(config-ipv4-acl)# 20 deny 192.168.34.0 0.0.0.255`
	or
	`RP/0/RP0/CPU0:router(config-ipv6-acl)# 20 permit icmp any any`
		Specifies one or more conditions allowed or denied in IPv4 access list acl_1.
		• The optional **log** keyword causes an information logging message about the packet that matches the entry to be sent to the console.
		• The optional **log-input** keyword provides the same function as the **log** keyword, except that the logging message also includes the input interface.
		• 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.
		or
		Specifies one or more conditions allowed or denied in IPv6 access list acl_2.
		• Refer to the **permit** (IPv6) and **deny** (IPv6) commands for more information on filtering IPv6 traffic based on IPv6 option headers and upper-layer protocols such as ICMP, TCP, and UDP.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-ipv6-acl)# 30 deny tcp any any gt 5000`	**Note** Every IPv6 access list has an implicit **deny ipv6 any any** statement as its last match condition. An IPv6 access list must contain at least one entry for the implicit **deny ipv6 any any** statement to take effect.
Step 5	Repeat Step 4 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 6	**commit**	
Step 7	**show access-lists** [**ipv4** | **ipv6**] [*access-list-name* **hardware** {**ingress** | **egress**} [**interface** *type interface-path-id*] {**sequence** *number* | **location** *node-id*} | **summary** [*access-list-name*] | *access-list-name* [*sequence-number*] | **maximum** [**detail**] [**usage** {**pfilter location** *node-id*}]] **Example:** `RP/0/RP0/CPU0:router# show access-lists ipv4 acl_1`	(Optional) Displays the contents of a named IPv4 or IPv6 access list. • Review the output to see that the access list includes the updated information.

What to do next

If your access list is not already applied to an interface or line or otherwise referenced, apply the access list. See the "Applying Access Lists, on page 14" section for information about how to apply an access list.

Copying Prefix Lists

This task copies an IPv4 or IPv6 prefix list.

SUMMARY STEPS

1. **copy prefix-list** {**ipv4** | **ipv6**} *source-name destination-name*
2. Do one of the following:
   - **show prefix-list ipv4** [*name*] [*sequence-number*] [**summary**]
   - **show prefix-list ipv6** [*name*] [*sequence-number*] [**summary**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**copy prefix-list** {**ipv4** | **ipv6**} *source-name destination-name* **Example:** `RP/0/RP0/CPU0:router# copy prefix-list ipv6 list_1 list_2`	Creates a copy of an existing IPv4 or IPv6 prefix list. • Use the *source-name* argument to specify the name of the prefix list to be copied and the *destination-name* argument to specify where to copy the contents of the source prefix list.

	Command or Action	Purpose
		• The *destination-name* argument must be a unique name; if the *destination-name* argument name exists for a prefix list, the prefix list is not copied.
Step 2	Do one of the following: • **show prefix-list ipv4** [*name*] [*sequence-number*] [**summary**] • **show prefix-list ipv6** [*name*] [*sequence-number*] [**summary**] **Example:** `RP/0/RP0/CPU0:router# show prefix-list ipv6 list_2`	(Optional) Displays the contents of current IPv4 or IPv6 prefix lists. • Review the output to see that prefix list list_2 includes the entries from list_1.

Sequencing Prefix List Entries and Revising the Prefix List

This task shows how to assign sequence numbers to entries in a named prefix list and how to add or delete an entry to or from a prefix list. It is assumed a user wants to revise a prefix list. Resequencing a prefix list is optional.

Before you begin

Note Resequencing IPv6 prefix lists is not supported.

SUMMARY STEPS

1. **resequence prefix-list ipv4** *name* [*base* [*increment*]]
2. **configure**
3. {**ipv4** | **ipv6**} **prefix-list** *name*
4. [*sequence-number*] {**permit** | **deny**} *network/length* [**ge** *value*] [**le** *value*] [**eq** *value*]
5. Repeat Step 4 as necessary, adding statements by sequence number where you planned. Use the **no** *sequence-number* command to delete an entry.
6. **commit**
7. Do one of the following:
   - **show prefix-list ipv4** [*name*] [*sequence-number*]
   - **show prefix-list ipv6** [*name*] [*sequence-number*] [**summary**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**resequence prefix-list ipv4** *name* [*base* [*increment*]] **Example:** `RP/0/RP0/CPU0:router# resequence prefix-list ipv4 pfx_1 10 15`	(Optional) Resequences the named IPv4 prefix list using the starting sequence number and the increment of sequence numbers. • This example resequences a prefix list named pfx_1. The starting sequence number is 10 and the increment is 15.
Step 2	**configure**	
Step 3	{**ipv4** | **ipv6**} **prefix-list** *name* **Example:** `RP/0/RP0/CPU0:router(config)# ipv6 prefix-list pfx_2`	Enters either IPv4 or IPv6 prefix list configuration mode and configures the named prefix list.
Step 4	[*sequence-number*] {**permit** | **deny**} *network/length* [**ge** *value*] [**le** *value*] [**eq** *value*] **Example:** `RP/0/RP0/CPU0:router(config-ipv6_pfx)# 15 deny 128.0.0.0/8 eq 24`	Specifies one or more conditions allowed or denied in the named prefix list.
Step 5	Repeat Step 4 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 prefix list.
Step 6	**commit**	
Step 7	Do one of the following: • **show prefix-list ipv4** [*name*] [*sequence-number*] • **show prefix-list ipv6** [*name*] [*sequence-number*] [**summary**] **Example:** `RP/0/RP0/CPU0:router# show prefix-list ipv6 pfx_2`	(Optional) Displays the contents of current IPv4 or IPv6 prefix lists. • Review the output to see that prefix list pfx_2 includes all new information.

How to Implement ACL-based Forwarding

This section contains the following procedures:

Configuring ACL-based Forwarding with Security ACL

Perform this task to configure ACL-based forwarding with security ACL.

SUMMARY STEPS

1. **configure**
2. **ipv4 access-list** *name*
3. [*sequence-number*] **permit** *protocol source source-wildcard destination destination-wildcard* [**precedence** *precedence*] [**[default] nexthop1** [**vrf** *vrf-name*][**ipv4** *ipv4-address1*] **nexthop2**[**vrf** *vrf-name*][**ipv4** *ipv4-address2*] **nexthop3**[**vrf** *vrf-name*][**ipv4** *ipv4-address3*]] [**dscp** *dscp*] [**fragments**] [**log** | **log-input**] [[**ttl** *ttl* [*value1 ... value2*]]
4. **commit**
5. **show access-list ipv4** [[*access-list-name* **hardware** {**ingress** | **egress**} [**interface** *type interface-path-id*] {**sequence** *number* | **location** *node-id*} | **summary** [*access-list-name*] | *access-list-name* [*sequence-number*] | **maximum** [**detail**] [**usage** {**pfilter location** *node-id*}]]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**ipv4 access-list** *name* **Example:** `RP/0/RP0/CPU0:router(config)# ipv4 access-list security-abf-acl`	Enters IPv4 access list configuration mode and configures the specified access list.

Step 3	[*sequence-number*] **permit** *protocol source source-wildcard destination destination-wildcard* [**precedence** *precedence*] [**[default] nexthop1** [**vrf** *vrf-name*][**ipv4** *ipv4-address1*] **nexthop2**[**vrf** *vrf-name*][**ipv4** *ipv4-address2*] **nexthop3**[**vrf** *vrf-name*][**ipv4** *ipv4-address3*]] [**dscp** *dscp*] [**fragments**] [**log**	**log-input**] [[**ttl** *ttl* [*value1 ... value2*]]
		Example:
		Example:
		`RP/0/RP0/CPU0:router# show access-lists ipv4 v4_acl`
		`ipv4 access-list v4_acl`
		` 10 permit ipv4 any host 172.1.1.1 nexthop1 vrf vrf_A ipv4 1.1.1.1 nexthop2 vrf vrf_B ipv4 2.2.2.2 nexthop3 vrf vrf_C ipv4 3.3.3.3`
			Sets the conditions for an IPv4 access list. The configuration example shows how to configure ACL-based forwarding with security ACL.
			• The **nexthop1, nexthop2, nexthop3** keywords forward the specified next hop for this entry.
			• If the **default** keyword is configured, ACL-based forwarding action is taken only if the results of the PLU lookup for the destination of the packets determine a default route; that is, no specified route is determined to the destination of the packet.

Step 4	**commit**	
Step 5	**show access-list ipv4** [[*access-list-name* **hardware** {**ingress**	**egress**} [**interface** *type interface-path-id*] {**sequence** *number*	**location** *node-id*}	**summary** [*access-list-name*]	*access-list-name* [*sequence-number*]	**maximum** [**detail**] [**usage** {**pfilter location** *node-id*}]] **Example:**	Displays the information for ACL software.

	Command or Action	Purpose

	`RP/0/RP0/CPU0:router# show access-lists ipv4 security-abf-acl`
	
	`RP/0/RP0/CPU0:router# show access-lists ipv4 v4_acl`
	
	`ipv4 access-list v4_acl`
	` 10 permit ipv4 any host 172.1.1.1 nexthop1 vrf vrf_A ipv4 1.1.1.1 nexthop2 vrf vrf_B ipv4 2.2.2.2 nexthop3 vrf vrf_C ipv4 3.3.3.3`	

Configuring Pure ACL-Based Forwarding for IPv6 ACL

SUMMARY STEPS

1. **configure**
2. {**ipv6** } **access-list** *name*
3. [*sequence-number*] **permit** *protocol source source-wildcard destination destination-wildcard* [**precedence** *precedence*] [**default nexthop** [*ipv6-address1*] [*ipv6-address2*] [*ipv6-address3*]] [**dscp** *dscp*] [**fragments**] [**log** | **log-input**] [**nexthop** [*ipv6-address1*] [*ipv6-address2*] [*ipv6-address3*]] [**ttl** *ttl value* [*value1 ... value2*]][**vrf** *vrf-name* [*ipv6-address1*] [*ipv6-address2*] [*ipv6-address3*]]
4. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	{**ipv6** } **access-list** *name* **Example:** `RP/0/RP0/CPU0:router(config)# ipv6 access-list security-abf-acl`	Enters IPv6 access list configuration mode and configures the specified access list.
Step 3	[*sequence-number*] **permit** *protocol source source-wildcard destination destination-wildcard* [**precedence** *precedence*] [**default nexthop** [*ipv6-address1*] [*ipv6-address2*] [*ipv6-address3*]] [**dscp** *dscp*] [**fragments**] [**log**	**log-input**] [**nexthop** [*ipv6-address1*] [*ipv6-address2*] [*ipv6-address3*]] [**ttl** *ttl value* [*value1 ... value2*]][**vrf** *vrf-name* [*ipv6-address1*] [*ipv6-address2*] [*ipv6-address3*]] **Example:** `RP/0/RP0/CPU0:router(config-ipv6-acl)# 10 permit ipv6 host 100:1:1:2:3::1 host 10:11:12::2 nexthop1`	Sets the conditions for an IPv6 access list. The configuration example shows how to configure pure ACL-based forwarding for ACL. • The **nexthop** keyword forwards the specified next hop for this entry.

	Command or Action	Purpose
	`ipv6 195:1:1:200:5ff:fe00:0`	
Step 4	**commit**	

Configuring Pure ACL-based Forwarding for ACL

Perform this task to configure pure ACL-based forwarding for ACL.

SUMMARY STEPS

1. **configure**
2. {**ipv4** } **access-list** *name*
3. [*sequence-number*] **permit** *protocol source source-wildcard destination destination-wildcard* [**precedence** *precedence*] [**default nexthop** [*ipv4-address1*] [*ipv4-address2*] [*ipv4-address3*]] [**dscp** *dscp*] [**fragments**] [**log** | **log-input**] [**nexthop** [*ipv4-address1*] [*ipv4-address2*] [*ipv4-address3*]] [**ttl** *ttl value* [*value1 ... value2*]][**vrf** *vrf-name* [*ipv4-address1*] [*ipv4-address2*] [*ipv4-address3*]]
4. **commit**
5. **show access-list ipv4** [*access-list-name* **hardware** {**ingress** | **egress**} [**interface** *type interface-path-id*] {**sequence** *number* | **location** *node-id*} | **summary** [*access-list-name*] | *access-list-name* [*sequence-number*] | **maximum** [**detail**] [**usage** {**pfilter location** *node-id*}]]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	{**ipv4** } **access-list** *name* **Example:** `RP/0/RP0/CPU0:router(config)# ipv4 access-list security-abf-acl`	Enters IPv4 access list configuration mode and configures the specified access list.

Step 3	[*sequence-number*] **permit** *protocol source source-wildcard destination destination-wildcard* [**precedence** *precedence*] [**default nexthop** [*ipv4-address1*] [*ipv4-address2*] [*ipv4-address3*]] [**dscp** *dscp*] [**fragments**] [**log**	**log-input**] [**nexthop** [*ipv4-address1*] [*ipv4-address2*] [*ipv4-address3*]] [**ttl** *ttl value* [*value1 ... value2*]][**vrf** *vrf-name* [*ipv4-address1*] [*ipv4-address2*] [*ipv4-address3*]]
		Example:
		`RP/0/RP0/CPU0:router(config-ipv4-acl)# 10 permit ipv4 10.0.0.0 0.255.255.255 any nexthop 50.1.1.2`
		`RP/0/RP0/CPU0:router(config-ipv4-acl)# 15 permit ipv4 30.2.1.0 0.0.0.255 any`
		`RP/0/RP0/CPU0:router(config-ipv4-acl)# 20 permit ipv4 30.2.0.0 0.0.255.255 any nexthop 40.1.1.2`
			Sets the conditions for an IPv4 or an IPv6 access list. The configuration example shows how to configure pure ACL-based forwarding for ACL.
			• The **nexthop** keyword forwards the specified next hop for this entry.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-ipv4-acl)# 25 permit ipv4 any any`	
Step 4	**commit**	
Step 5	**show access-list ipv4** [*access-list-name* **hardware** {**ingress** | **egress**} [**interface** *type interface-path-id*] {**sequence** *number* | **location** *node-id*} | **summary** [*access-list-name*] | *access-list-name* [*sequence-number*] | **maximum** [**detail**] [**usage** {**pfilter location** *node-id*}]] **Example:** `RP/0/RP0/CPU0:router# show access-lists ipv4 security-abf-acl`	Displays the information for ACL software.

ACL-Chaining

ACL-Chaining also known as Multi-ACL enables customers to apply two IPv4 or IPv6 (common and interface) ACLs on an interface for packet filtering at the router. One ACL is common across multiple interfaces on the line card. This provides Ternary Content Addressable Memory(TCAM)/HW scalability.

ACL-Chaining Overview

Currently, the packet filter process (pfilter_ea) supports only one ACL to be applied per direction and per protocol on an interface. This leads to manageability issues if there are common ACL entries needed on most interfaces. Duplicate ACEs are configured for all those interfaces, and any modification to the common ACEs needs to be performed for all ACLs.

A typical ACL on the edge box for an ISP has two sets of ACEs:

- common ISP specific ACEs (ISP protected address block)
- customer/interface specific ACEs (Customer source address block)

The purpose of these address blocks is to deny access to ISP's protected infrastructure networks and anti-spoofing protection by allowing only customer source address blocks. This results in configuring unique ACL per interface and most of the ACEs being common across all the ACLs on a box. ACL provisioning and modification is very cumbersome. Any changes to the ACE impacts every customer interface. (This also wastes the HW/TCAM resources as the common ACEs are being replicated in all ACLs).

The ACL chaining feature also known as Multi-ACL allows you to configure more than one ACL that can be applied to a single interface. The goal is to separate various types of ACLs for management, and also allow you to apply both of them on the same interface, in a defined order.

Restrictions for Common ACL

The following restrictions apply while implementing Common ACL:

- Common ACL is supported in only ingress direction and for L3 interfaces only.

- The **interface-statistics** option is not available for common ACLs.
- The **hardware-count** option is available for only IPv4 ACLs.
- Only one common IPv4 and IPv6 ACL is supported on each line card.
- The common ACL option is not available for Ethernet Service (ES) ACLs.
- You can specify only common ACL or only interface ACL or both common and interface ACL in this command.
- The **compress** option is not supported for common ACLs.

Configuring an Interface to accept Common ACL

Perform this task to configure the interface to accept a common ACL along with the interface specific ACL:

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-path-id*
3. { **ipv4 | ipv6** } **access-group** { **common** *access-list-name* { [*access-list-name* **ingress** [**interface-statistics**]] | **ingress** } | *access-list-name* { **ingress | egress** } [**interface-statistics**] } [**hardware-count**]
4. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config)# interface TenGigE 0/2/0/1`	This command configures an interface (in this case a TenGigabitEThernet interface) and enters the interface configuration mode.
Step 3	{ **ipv4 | ipv6** } **access-group** { **common** *access-list-name* { [*access-list-name* **ingress** [**interface-statistics**]] | **ingress** } | *access-list-name* { **ingress | egress** } [**interface-statistics**] } [**hardware-count**] **Example:** `RP/0/RP0/CPU0:router(config-if)# ipv4 access-group common acl-p acl1 ingress`	Configures the interface to accept a common ACL along with the interface specific ACL.
Step 4	**commit**	

Configuration Examples for Implementing Access Lists and Prefix Lists

This section provides the following configuration examples:

Resequencing Entries in an Access List: Example

The following example shows access-list resequencing. The starting value in the resequenced access list is 1 , and increment value is 2 . The subsequent entries are ordered based on the increment values that users provide, and the range is from 1 to 2147483646.

When an entry with no sequence number is entered, by default it has a sequence number of 10 more than the last entry in the access list.

```
ipv4 access-list acl_1
10 permit ip host 10.3.3.3 host 172.16.5.34
20 permit icmp any any
30 permit tcp any host 10.3.3.3
40 permit ip host 10.4.4.4 any
60 permit ip host 172.16.2.2 host 10.3.3.12
70 permit ip host 10.3.3.3 any log
80 permit tcp host 10.3.3.3 host 10.1.2.2
100 permit ip any any

configure
ipv6 access-list acl_1
end
resequence ipv6 access-list acl_1 10 20

ipv4 access-list acl_1
10 permit ip host 10.3.3.3 host 172.16.5.34
30 permit icmp any any
50 permit tcp any host 10.3.3.3
70 permit ip host 10.4.4.4 any
90 Dynamic test permit ip any any
110 permit ip host 172.16.2.2 host 10.3.3.12
130 permit ip host 10.3.3.3 any log
150 permit tcp host 10.3.3.3 host 10.1.2.2
170 permit ip host 10.3.3.3 any
190 permit ip any any
```

Adding Entries with Sequence Numbers: Example

In the following example, an new entry is added to IPv4 access list acl_5.

```
ipv4 access-list acl_5
 2 permit ipv4 host 10.4.4.2 any
 5 permit ipv4 host 10.0.0.44 any
 10 permit ipv4 host 10.0.0.1 any
 20 permit ipv4 host 10.0.0.2 any
configure
ipv4 access-list acl_5
 15 permit 10.5.5.5 0.0.0.255
end
ipv4 access-list acl_5
```

```
 2 permit ipv4 host 10.4.4.2 any
 5 permit ipv4 host 10.0.0.44 any
 10 permit ipv4 host 10.0.0.1 any
 15 permit ipv4 10.5.5.5 0.0.0.255 any
 20 permit ipv4 host 10.0.0.2 any
```

Adding Entries Without Sequence Numbers: Example

The following example shows how an entry with no specified sequence number is added to the end of an access list. When an entry is added without a sequence number, it is automatically given a sequence number that puts it at the end of the access list. Because the default increment is 10, the entry will have a sequence number 10 higher than the last entry in the existing access list.

```
configure
ipv4 access-list acl_10
permit  1.1.1.1 0.0.0.255
permit  2.2.2.2 0.0.0.255
permit  3.3.3.3 0.0.0.255
end

ipv4 access-list acl_10
 10 permit ip  1.1.1.0 0.0.0.255 any
 20 permit ip  2.2.2.0 0.0.0.255 any
 30 permit ip  3.3.3.0 0.0.0.255 any

configure
ipv4 access-list acl_10
permit  4.4.4.4 0.0.0.255
end

ipv4 access-list acl_10
 10 permit ip  1.1.1.0 0.0.0.255 any
 20 permit ip  2.2.2.0 0.0.0.255 any
 30 permit ip  3.3.3.0 0.0.0.255 any
 40 permit ip  4.4.4.0 0.0.0.255 any
```

Configuration Examples for Implementing ACL-based Forwarding

This section provides the following configuration examples:

All configuration examples include a forwarded action **nexthop** keyword. If the **default nexthop** keyword is configured, ABF action is taken only if the pointer lookup (PLU) of the destination of the packets results in hitting a default route; for example, no specific route is specified to the packet destination.

ACL with Security and ACL-based Forwarding Access Control Entry: Example

The following example shows how to configure ACL with security and an ACL-based forwarding access control entry (ACE):

```
configure
    ipv4 access-list security-abf-acl
    10 permit ipv4 10.0.0.0 0.255.255.255 any
    15 permit ipv4 10.2.0.0 0.0.255.255 any next-hop 10.1.1.2
```

```
    20 deny ipv4 10.1.0.0 0.0.255.255 any
    25 permit ipv4 10.0.0.0 0.255.255.255 any
    end
```

Note For ACL-based forwarding, the following command is programmed in the hardware after access list entry (ACE) 25:

```
deny ipv4 any any
```

The following methods are used to attach the ACL for both security and ACL-based forwarding ACE to an interface in ingress direction:

- Packets entering an interface with source address 10.*x.x.x* are forwarded using a traditional forwarding lookup.
- Packets entering an interface with source address 30.2.*x.x* are forwarded to next hop 40.1.1.2 (if reachable through FIB).
- Packets entering an interface with source address 30.1.*x.x* are dropped by security ACE 20.
- All other packets that are entering an interface are dropped by security ACL.

Pure ACL-based Forwarding for ACL Example

The following example shows how to configure pure ABF for ACL:

```
configure
    ipv4 access-list security-abf-acl
    10 permit ipv4 10.0.0.0 0.255.255.255 any next-hop 10.1.1.2
    15 permit ipv4 10.2.1.0 0.0.0.255 any
    20 permit ipv4 10.2.0.0 0.0.255.255 any next-hop 10.1.1.2
    25 permit ipv4 any any
    end
```

IPv6 ACL-based Forwarding Example

The following example shows how to configure IPv6 supported ABF:

```
configure
ipv6 access-list v6_abf
    10 permit ipv6 host 100:1:1:2:3::1 host 10:11:12::2 nexthop1 ipv6 195:1:1:1:200:5ff:fe00:0

    20 permit ipv6 host 100:1:1:2:3::1 host 10:11:12::2 nexthop1 ipv6 195:1:1:1:200:5ff:fe00:0
  nexthop2 ipv6 192:3:2:2:200:3ff:fe00:0 nexthop3 ipv6 192:4:2:2:200:3ff:fe00:0
    30 permit ipv6 host 100:1:1:2:3::1 host 10:11:12::2 nexthop1 vrf VRF1
    40 permit ipv6 host 100:1:1:2:3::1 host 10:11:12::3 nexthop1 vrf VRF1 ipv6
192:2:2:2:200:3ff:fe00:0
    50 permit ipv6 host 100:1:1:2:3::1 host 10:11:12::2 default nexthop1 ipv6
195:1:1:1:200:5ff:fe00:0
    60 permit ipv6 any any nexthop1 vrf VRF1 ipv6 192:2:2:2:200:3ff:fe00:0 nexthop2 vrf
VRF1 ipv6 192:3:2:2:200:3ff:fe00:0 nexthop3 vrf VRF1 ipv6 192:4:2:2:200:3ff:fe00:0
end
```

Note For ACL-based forwarding, the following command is programmed in hardware of the ACL:F

```
deny ipv4 any any
```

Therefore, the following ACE command must be issued to let other traffic through:

```
25 permit ipv4 any any
```

The following methods are used to attach the ACL, which is used only for an ACL-based forwarding ACE, to an interface in the ingress direction:

- Packets entering an interface with source address 10.*x*.*x*.*x* are forwarded to next hop 10.1.1.2 (if reachable through FIB).
- Packets entering an interface with source address 10.2.1.*x* are forwarded using traditional forwarding lookup.
- Packets entering an interface with source address 19.2.*x*.*x*, but not in 30.2.1.*x*, are forwarded to next hop 10.1.1.2 (if reachable through FIB).
- All other packets entering an interface are permitted by ACE 25 and are forwarded by using a traditional forwarding lookup.
- ACE 25 ensures that all packets not matching this ACL-based forwarding ACE are forwarded and are not dropped due to an implicit deny ACE that is installed after ACE 25 by the software.

IPv6 ACL in Class Map

In Release 4.2.1, Quality of Service (Qos) features on ASR 9000 Ethernet line card and ASR 9000 Enhanced Ethernet line card are enhanced to support these:

- ASR 9000 Enhanced Ethernet LC:
  - Support on L2 and L3 interface and sub-interface
  - Support on bundle L2 and L3 interface and sub-interface
  - Support for both ingress and egress directions
  - ICMP code and type for IPv4/IPv6
- ASR 9000 Ethernet LC:
  - Support on only L3 interface and sub-interface
  - Support on L3 bundle interface and sub-interface
  - Support for both ingress and egress directions
  - ICMP code and type for IPv4/IPv6
- IPv6-supported match fields:

- IPv6 Source Address
- IPv6 Destination Address
- IPv6 Protocol
- Time to live (TTL) or hop limit
- Source Port
- Destination Port
- TCP Flags
- IPv6 Flags (Routing Header(RH), Authentication Header(AH) and Destination Option Header(DH))

- Class map with IPv6 ACL that also supports:
  - IPv4 ACL
  - Discard class
  - QoS Group
  - Outer CoS
  - Inner CoS
  - Outer VLAN (ASR 9000 Enhanced Ethernet LC only)
  - Inner VLAN (ASR 9000 Enhanced Ethernet LC only)
  - match-not option
  - type of service (TOS) support

- Policy-map with IPv6 ACL supports:
  - hierarchical class-map

Configuring IPv6 ACL QoS - An Example

This example shows how to configure IPv6 ACL QoS with IPv4 ACL and other fields :

```
ipv6 access-list aclv6
10 permit ipv6 1111:6666::2/64 1111:7777::2/64 authen
30 permit tcp host 1111:4444::2 eq 100 host 1111:5555::2 ttl eq 10
!

ipv4 access-list aclv4
10 permit ipv4 host 10.6.10.2 host 10.7.10.2
!

class-map match-any c.aclv6
match access-group ipv6 aclv6
match access-group ipv4 aclv4
match cos 1
end-class-map
```

```
!

policy-map p.aclv6
class c.aclv6
  set precedence 3
!
class class-default
!
end-policy-map
!



show qos-ea km policy p.aclv6 vmr interface tenGigE 0/1/0/6.10 hw

================================================================================
B : type & id      E : ether type     VO : vlan outer     VI : vlan inner
Q : tos/exp/group  X : Reserved       DC : discard class  Fl : flags
F2: L2 flags       F4: L4 flags       SP/DP: L4 ports
T : IP TTL         D : DFS class#      L : leaf class#
Pl: Protocol       G : QoS Grp         M : V6 hdr ext.     C : VMR count
--------------------------------------------------------------------------------
policy name p.aclv6 and km format type 4
Total Egress TCAM entries: 5
|B    F2 VO   VI   Q  G  DC T  F4 Pl SP   DP   M  IPv4/6 SA                          IPv4/6
 DA
================================================================================
V|3019 00 0000 0000 00 00 00 00 00 00 0000 0000 80 11116666:00000000:00000000:00000000
11117777:00000000:00000000:00000000
M|0000 FF FFFF FFFF FF FF FF FF FF FF FFFF FFFF 7F 00000000:00000000:FFFFFFFF:FFFFFFFF
00000000:00000000:FFFFFFFF:FFFFFFFF
R| C=0 03080200 000000A6 F06000FF 0000FF00 0002FF00 00FF0000 FF000000 00000000
V|3019 00 0000 0000 00 00 00 0A 01 00 0064 0000 00 11114444:00000000:00000000:00000002
11115555:00000000:00000000:00000002
M|0000 FF FFFF FFFF FF FF FF 00 FE FF 0000 FFFF FF 00000000:00000000:00000000:00000000
00000000:00000000:00000000:00000000
R| C=1 03080200 000000A6 F06000FF 0000FF00 0002FF00 00FF0000 FF000000 00000000
V|3018 00 0000 0000 00 00 00 00 00 00 0000 0000 00 0A060A02 -------- -------- --------
0A070A02 -------- -------- --------
M|0000 FF FFFF FFFF FF FF FF FF FF FF FFFF FFFF FF 00000000 -------- -------- --------
00000000 -------- -------- --------
R| C=2 03080200 000000A6 F06000FF 0000FF00 0002FF00 00FF0000 FF000000 00000000
V|3018 00 2000 0000 00 00 00 00 00 00 0000 0000 00 00000000:00000000:00000000:00000000
00000000:00000000:00000000:00000000
M|0003 FF 1FFF FFFF FF FF FF FF FF FF FFFF FFFF FF FFFFFFFF:FFFFFFFF:FFFFFFFF:FFFFFFFF
FFFFFFFF:FFFFFFFF:FFFFFFFF:FFFFFFFF
R| C=3 03080200 000000A6 F06000FF 0000FF00 0002FF00 00FF0000 FF000000 00000000
V|3018 00 0000 0000 00 00 00 00 00 00 0000 0000 00 00000000:00000000:00000000:00000000
00000000:00000000:00000000:00000000
M|0003 FF FFFF FFFF FF FF FF FF FF FF FFFF FFFF FF FFFFFFFF:FFFFFFFF:FFFFFFFF:FFFFFFFF
FFFFFFFF:FFFFFFFF:FFFFFFFF:FFFFFFFF
R| C=4 03000200 00010002 FF0000FF 0000FF00 0002FF00 00FF0000 FF000000 00000000
```

This example shows how to configure hierarchical policy map:

```
ipv6 access-list aclv6.p
10 permit ipv6 1111:1111::/8 2222:2222::/8

ipv6 access-list aclv6.c
10 permit ipv6 host 1111:1111::2 host 2222:2222::3
```

```
class-map match-any c.aclv6.c
match not access-group ipv6 aclv6.c
end-class-map
!

class-map match-any c.aclv6.p
match access-group ipv6 aclv6.p
end-class-map
!

policy-map child
class c.aclv6.c
  set precedence 7
!

policy-map parent
class c.aclv6.p
  service-policy child
  set precedence 1
```

```
(config)#do show qos-ea km policy parent vmr interface tenGigE 0/1/0/6 hw

================================================================================
B : type & id      E : ether type     VO : vlan outer     VI : vlan inner
Q : tos/exp/group  X : Reserved       DC : discard class  Fl : flags
F2: L2 flags       F4: L4 flags       SP/DP: L4 ports
T : IP TTL         D : DFS class#      L : leaf class#
Pl: Protocol       G : QoS Grp         M : V6 hdr ext.     C : VMR count

================================================================================
policy name parent and format type 4
Total Ingress TCAM entries: 3
|B    F2 VO   VI   Q  G  DC T  F4 Pl SP   DP   M  IPv4/6 SA                          IPv4/6
 DA
================================================================================
V|200D 00 0000 0000 00 00 00 00 00 00 0000 0000 00 11111111:00000000:00000000:00000002
22222222:00000000:00000000:00000003
M|0000 FF FFFF FFFF FF FF FF FF FF FF FFFF FFFF FF 00000000:00000000:00000000:00000000
00000000:00000000:00000000:00000000
R| C=0 11800200 00020000 29000000 80004100 00000000 00000000 00000000 00000000
V|200D 00 0000 0000 00 00 00 00 00 00 0000 0000 00 11000000:00000000:00000000:00000000
22000000:00000000:00000000:00000000
M|0000 FF FFFF FFFF FF FF FF FF FF FF FFFF FFFF FF 00FFFFFF:FFFFFFFF:FFFFFFFF:FFFFFFFF
00FFFFFF:FFFFFFFF:FFFFFFFF:FFFFFFFF
R| C=1 11800200 00010000 29000000 80004700 00000000 00000000 00000000 00000000
V|200C 00 0000 0000 00 00 00 00 00 00 0000 0000 00 00000000:00000000:00000000:00000000
00000000:00000000:00000000:00000000
M|0003 FF FFFF FFFF FF FF FF FF FF FF FFFF FFFF FF FFFFFFFF:FFFFFFFF:FFFFFFFF:FFFFFFFF
FFFFFFFF:FFFFFFFF:FFFFFFFF:FFFFFFFF
R| C=2 11000200 00030000 00000000 00000000 00000000 00000000 00000000 00000000
```

Configuring an Interface to accept Common ACL - Examples

This section provides configuration examples of common ACL.

This example shows how to replace an ACL configured on the interface without explicitly deleting the ACL:

```
Interface Pos0/2/0/0
ipv4 access-group common  C_acl ACL1 ingress
commit
replace Interface acl ACL1 by ACL2
Interface Pos0/2/0/0
ipv4 access-group common  C_acl ACL2 ingress
commit
```

This example shows how common ACL cannot be replaced on interfaces without deleting it explicitly from the interface:

```
Interface Pos0/2/0/0
ipv4 access-group common  C_acl1 ACL1 ingress
commit
change the common acl to C_acl2
Interface Pos0/2/0/0
no ipv4 access-group common  C_acl1  ACL1 ingress
commit
Interface Pos0/2/0/0
ipv4 access-group common  C_acl2  ACL1 ingress
commit
```

Note When reconfiguring common ACL, you must ensure that no other interface on the line card is attached to the common ACL. In other words, atomic replacement of common ACL is not possible.

Note If both common ACL and interface ACL are attached to an interface and only one of the above is reconfigured on the interface, then the other will be removed automatically.

```
Interface Pos0/2/0/0
ipv4 access-group common  C_acl1 ACL1 ingress
commit

Interface Pos0/2/0/0
ipv4 access-group  ACL1 ingress
commit
This removes the common acl.

Interface Pos0/2/0/0
ipv4 access-group common  C_acl1 ACL1 ingress
commit

Interface Pos0/2/0/0
ipv4 access-group common C_acl1  ingress
commit
```

This example shows how the interface ACL is removed:

```
Interface Pos0/2/0/0
ipv4 access-group common  C_acl1 ACL1 ingress
commit

Interface Pos0/2/0/0
no ipv4 access-group common  acl  acl ingress
Commit
```

Additional References

The following sections provide references related to implementing access lists and prefix lists.

Related Documents

Related Topic	Document Title
Access list commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*Access List Commands* module in *IP Addresses and Services Command Reference for Cisco CRS Routers*
Prefix list commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*Prefix List Commands* module in *IP Addresses and Services Command Reference for Cisco CRS Routers*
Terminal services commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*Terminal Services Commands* module in *System Management Command Reference for Cisco CRS Routers*

Standards

Standards	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	—

MIBs

MIBs	MIBs Link
—	To locate and download MIBs, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: https://mibs.cloudapps.cisco.com/ITDIT/MIBS/servlet/index

RFCs

RFCs	Title
No new or modified RFCs are supported by this feature, and support for existing RFCs has not been modified by this feature.	—

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/techsupport