Implementing Access Lists and Prefix Lists

An access control list (ACL) consists of one or more access control entries (ACE) that collectively define the network traffic profile. This profile can then be referenced by Cisco IOS XR softwarefeatures such as traffic filtering, route filtering, QoS classification, and access control. Each ACL includes an action element (permit or deny) and a filter element based on criteria such as source address, destination address, protocol, and protocol-specific parameters.

Prefix lists are used in route maps and route filtering operations and can be used as an alternative to access lists in many Border Gateway Protocol (BGP) route filtering commands. A prefix is a portion of an IP address, starting from the far left bit of the far left octet. By specifying exactly how many bits of an address belong to a prefix, you can then use prefixes to aggregate addresses and perform some function on them, such as redistribution (filter routing updates).

This module describes the new and revised tasks required to implement access lists and prefix lists on the Cisco ASR 9000 Series Router

Note For a complete description of the access list and prefix list commands listed in this module, refer to the *IP Addresses and Services Command Reference for Cisco ASR 9000 Series Routers*To locate documentation of other commands that appear in this chapter, use the command reference master index, or search online.

Feature History for Implementing Access Lists and Prefix Lists

Release	Modification
Release 3.7.2	This feature was introduced.
Release 4.2.1	IPv6 ACL over BVI interface feature was added.
Release 4.2.1	ACL in Class map feature was added.
Release 5.3.2	Multi-level ACL Chaining feature was added for Cisco ASR 9000 High Density 100GE Ethernet Line Card.
Release 6.0.1	Feature to disable automic ACL updates was introduced.

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:

- From Release 5.3.2 onward, on Cisco ASR 9000 High Density 100GE Ethernet Line Cards, up to 4095 unique IPv4 and Ethernet Service (ES) ACLs and up to 4095 unique IPv6 ACLs are supported.
- Layer 2/Layer 3 ACLs are not supported on Layer 2 interfaces.
- IPv4 ACLs are not supported for loopback and interflex interfaces.
- IPv4 and IPv6 ACLs are not supported on service application and service infrastructure interfaces.
- 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:
  1. Remove the ACL from the interface.
  2. Reconfigure the ACL.
  3. Reapply the ACL to the interface.

Note 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.

  If the packet comes on an ASR 9000 Ethernet Line Card, and is labeled as part of an MPLS flow, then the ingress ASR 9000 Ethernet Line Card cannot apply ACL. Also, for ASR 9000 Ethernet Line Cards, if the label is popped because it is routed to an attached customer edge (CE), then the egress line card (LC) sees a plain IP. But, it still cannot apply an egress (outbound) ACL on the IP packet. Whereas, an ASR 9000 Enhanced Ethernet Line Card can perform an egress IP ACL on this packet before sending it to the directly attached CE.

- Video Monitoring is not supported through ACLs on IPv6 interfaces.

Restrictions for Implementing ACL-Based Forwarding

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

- ABF is not supported for **for-us** packets (packets destined for the router).
- The following nexthop configurations are not supported: attaching ACL having a nexthop option in the egress direction, modifying an ACL attached in the egress direction having nexthop, denying an ACE with a nexthop.
- The A9K-SIP-700 LC and ASR 9000 Enhanced Ethernet LC support ABFv4 and ABFv6 in Release 4.2.0. ASR 9000 Ethernet LC does not support ABFv6 in Release 4.2.0, it only supports ABFv4.
- ABFv4 is supported on BVI interfaces for ASR 9000 Enhanced Ethernet line card. It is not supported for ASR 9000 Ethernet line card.

Note Nexthop egress over A9K-SIP-700 line card, ASR 9000 Ethernet line card, or virtual interfaces like GRE or BVI is supported when ABFv4 is configured for a BVI interface.

- ABFv6 is supported on IRB/BVI interfaces for ASR 9000 Enhanced Ethernet line card. It is not supported for ASR 9000 Ethernet line card.

Note There is one exception to this. In case of IP to TAG, the label is imposed by the ingress LC (based on ABF nexthop), and the packet crosses the fabric as a tag packet. These packets are handled by A9K-SIP-700 without any issue.

- Packets punted in the ingress direction from the NPU to the LC CPU are not subjected to ABF treatment due to lack of ABF support in the slow path.

- IP packet(s) needing fragmentation are not subjected to ABF. The packet is forwarded in the traditional way. Fragmented packets received are handled by ABF.

Hardware Limitations

- Support for ABF is only for IPv4 and Ethernet line cards. IPv6 and other interfaces are not supported.
- ABF is an ingress line card feature and the egress line card must be ABF aware.

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.

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.
- 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.

From Release 5.2.2, you can supply a wildcard mask with a source or destination address in an access list statement. The wildcard masking feature now supports IPv6 ACL with wildcard masking. This feature is supported in ASR 9000 Enhanced Ethernet Line Card.

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 and for IPv6 address 2001:db8:abcd:0012:0000:0000:0000:0000 corresponds to 2001:db8:abcd:0012::0/64.

Transport Layer Information

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

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. 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.

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 is not 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 is when a threshold is not specified.

Note The logging facility might drop some logging message packets if there are too many to be handled or if 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 should not be used as a billing tool or as an accurate source of the number of matches to an access list.

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 16" 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.

Note Hardware counters are enabled by default on 100Gigabit ethernet interfaces, Cisco ASR 9000 Ethernet line cards, and Cisco ASR 9000 Enhanced Ethernet line cards.

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).

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.

ABF-OT

To provide flexibility to the user in selecting the suitable next hop, the ABF functionality is enhanced to interact with object-tracking (OT), which impacts:

- Tracking prefix in CEF
- Tracking the line-state protocol
- IPSLA (IP Service Level Agreement)

IPv6 ACL Based Forwarding Object Tracking

The IPv6 ACL based forwarding (ABF) object tracking feature enables ABF to decide which next hop address to use, based on the state of the object being tracked for the next hop. IPv6 SLA echos are used to determine reachability to the next hop address. If the primary route is unreachable, the secondary route is used to forward traffic. IPv6 ABF object tracking is supported on ASR 9000 Enhanced Ethernet line cards only.

For information about the **object** command which is used to configure an object for tracking, see the *System Management Command Reference for Cisco ASR 9000 Series Routers*.

IPSLA support for Object tracking

The OT-module interacts with the IPSLA-module to get reachability information. With IPSLA, the routers perform periodic measurements

ACL Counters Using SNMP

Apart from viewing the access control list counters using commands, you can also get the ACL counter information using SNMP. When the router receives an SNMP request for ACE counters, the router responds by sending the packet count that matches each access control entry along with the byte count to the SNMP server.

You can use the **counter** *counter-name* command to aggregate several ACEs into a single counter.

The following features are supported when you retrieve ACL counters using SNMP:

- Hardware counters for interface ACLs applied with the **interface-statistics** command.
- Hardware ACL statistics on GigabitEthernet, TenGigabitEthernet, HundredGigabitEthernet, Bundle Ethernet interfaces, and subinterfaces.
- ACE label counter statistics.

The following features are not supported when you retrieve ACL counters using SNMP:

- Software counters.
- Counter names cannot be configured on ABF ACLs.
- Common ACLs. If an interface has both common ACL and interface ACL, statistics pertaining to ACEs from the common ACL are not returned.
- Hardware statistics for subscriber interfaces.
- Hardware statistics for ACEs with the same counter name.

Only Cisco ASR 9000 Enhanced Ethernet Line Cards support this feature. We recommend that you do not enable more than 50 unique counters in an ACL.

How to Implement Access Lists and Prefix Lists

IPv6 ACL support is available on the Cisco ASR 9000 SIP 700 linecard and the ASR 9000 Ethernet linecards. The relevant scale is:

- ACL enabled interfaces - 1000 (500 in each direction); for ASR 9000 Ethernet linecards- 4000
- Unique ACLs - 512 (with 5 ACEs each); for ASR 9000 Ethernet linecards- 2000
- Maximum ACEs per ACL - 8000 (for ASR 9000 Ethernet lincards, ACEs could be 16000, 8000, 4000- based on the LC model)
- IPv6 ACL log will also be supported.

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/RSP0/CPU0:router(config)# ipv4 access-list acl_1`
	or
	`RP/0/RSP0/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/RSP0/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**] • [*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* |	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. or

	Command or Action	Purpose

	protocol-port}] [**dscp** *value*] [*routing*] [**authen**] [**destopts**] [**fragments**] [**log** | **log-input**]
	Example:
	`RP/0/RSP0/CPU0:router(config-ipv4-acl)# 10 permit 172.16.0.0 0.0.255.255`
	`RP/0/RSP0/CPU0:router(config-ipv4-acl)# 20 deny 192.168.34.0 0.0.0.255`
	or
	`RP/0/RSP0/CPU0:router(config-ipv6-acl)# 20 permit icmp any any`
	`RP/0/RSP0/CPU0:router(config-ipv6-acl)# 30 deny tcp any any gt 5000`
		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 address list has two implicit permits used for neighbor advertisement and solicitation: Implicit Neighbor Discovery–Neighbor Advertisement (NDNA) permit, and Implicit Neighbor Discovery–Neighbor Solicitation (NDNS) permit.
		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/RSP0/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 16 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/RSP0/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/RSP0/CPU0:router(config-if)# ipv4 access-group p-in-filter in`
	`RP/0/RSP0/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/RSP0/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/RSP0/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/RSP0/CPU0:router(config)# ipv4 prefix-list pfx_1`
	or
	`RP/0/RSP0/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/RSP0/CPU0:router(config-ipv4_pfx)# 10 remark Deny all routes with a prefix of 10/8`
	`RP/0/RSP0/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/RSP0/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/RSP0/CPU0:router# show prefix-list ipv4 pfx_1`
	or
	`RP/0/RSP0/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/RSP0/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/RSP0/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/RSP0/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/RSP0/CPU0:router(config-ipv4-acl)# 20 permit 172.16.0.0 0.0.255.255`
	or
	`RRP/0/RSP0/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/RSP0/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 16" 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/RSP0/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/RSP0/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.

Note When an ACL is configured under an interface and its resequenced and rolled back, the interface experiences traffic loss for a short period of time.

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/RSP0/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/RSP0/CPU0:router(config)# ipv4 access-list acl_1`
	or
	`RP/0/RSP0/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/RSP0/CPU0:router(config-ipv4-acl)# 10 permit 172.16.0.0 0.0.255.255`
	`RP/0/RSP0/CPU0:router(config-ipv4-acl)# 20 deny 192.168.34.0 0.0.0.255`
	or
	`RP/0/RSP0/CPU0:router(config-ipv6-acl)# 20 permit`
		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
	`icmp any any` `RP/0/RSP0/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/RSP0/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 16" 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/RSP0/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/RSP0/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/RSP0/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/RSP0/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/RSP0/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/RSP0/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** [**ipv4** *ipv4-address1*] **nexthop2**[**ipv4** *ipv4-address2*] **nexthop3**[**ipv4** *ipv4-address3*]] [**dscp** *dscp*] [**fragments**] [**log** | **log-input**] [[**track** *track-name*] [**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/RSP0/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** [**ipv4** *ipv4-address1*] **nexthop2**[**ipv4** *ipv4-address2*] **nexthop3**[**ipv4** *ipv4-address3*]] [**dscp** *dscp*] [**fragments**] [**log**	**log-input**] [[**track** *track-name*] [**ttl** *ttl* [*value1 ... value2*]]
		Example:
		`RP/0/RSP0/CPU0:router(config-ipv4-acl)# 10 permit ipv4 10.0.0.0 0.255.255.255 any nexthop 50.1.1.2`
		`RP/0/RSP0/CPU0:router(config-ipv4-acl)# 15 permit ipv4 30.2.1.0 0.0.0.255 any`
		`RP/0/RSP0/CPU0:router(config-ipv4-acl)# 20 permit ipv4 30.2.0.0 0.0.255.255 any nexthop 40.1.1.2`
		`RP/0/RSP0/CPU0:router(config-ipv4-acl)# 25 permit ipv4 any any`
			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:** `RP/0/RSP0/CPU0:router# show access-lists ipv4 security-abf-acl`	Displays the information for ACL software.

Implementing IPSLA-OT

In this section, the following procedures are discussed:

Note When a large number of IPSLA instances need to be configured, it's more convenient to create a configuration file with all the configurations and then load the configuration file. The configuration statements in the configuration file should be properly indented including the exit statements, otherwise the configuration won't work when loading the configuration file.

Enabling track mode

SUMMARY STEPS

1. configure
2. **track** *track-name*
3. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**track** *track-name* **Example:** `RP/0/RSP0/CPU0:router(config)# track t1`	Enters track configuration mode.
Step 3	**commit**	

Configuring track type

There are different mechanisms to track the availability of the next-hop device. The tracking type can be of four types, using:

- line protocol

- list
- route
- IPSLA

Configuring tracking type (line protocol)

Line protocol is one of the object types the object tracker component can track. This object type provides an option for tracking state change notification from an interface. Based on the interface state change notification, it decides whether the track state should be UP or DOWN.

SUMMARY STEPS

1. configure
2. **track** *track-name*
3. **type line-protocol state interface** *type interface-path-id*
4. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**track** *track-name* **Example:** `RP/0/RSP0/CPU0:router(config)# track t1`	Enters track configuration mode.
Step 3	**type line-protocol state interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config-track)# type line-protocol state interface tengige 0/4/4/0`	Sets the interface which needs to be tracked for state change notifications.
Step 4	**commit**	

Configuring track type (list)

List is a boolen object type. Boolean refers to the capability of performing a boolean AND or boolean OR operation on combinations of different object types supported by object tracker.

SUMMARY STEPS

1. configure
2. **track** *track-name*
3. **type list boolean and**
4. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**track** *track-name* **Example:** `RP/0/RSP0/CPU0:router(config)# track t1`	Enters track configuration mode.
Step 3	**type list boolean and** **Example:** `RP/0/RSP0/CPU0:router(config-track)# type list boolean and`	Sets the list of track objects on which boolean AND or boolean OR operations could be performed.
Step 4	**commit**	

Configuring tracking type (route)

Route is a route object type. The object tracker tracks the fib notification to determine the route reachability and the track state.

SUMMARY STEPS

1. configure
2. **track** *track-name*
3. **type route reachability**
4. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**track** *track-name* **Example:** `RP/0/RSP0/CPU0:router(config)# track t1`	Enters track configuration mode.
Step 3	**type route reachability** **Example:** `RP/0/RSP0/CPU0:router(config-track)# type route reachability`	Sets the route on which reachability state needs to be learnt dynamically.
Step 4	**commit**	

Configuring tracking type (rtr)

IPSLA is an ipsla object type. The object tracker tracks the return code of ipsla operation to determine the track state changes.

SUMMARY STEPS

1. configure
2. **track** *track-name*
3. **type rtr** *ipsla operation id* **reachability**
4. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	configure **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**track** *track-name* **Example:** `RP/0/RSP0/CPU0:router(config)# track t1`	Enters track configuration mode.
Step 3	**type rtr** *ipsla operation id* **reachability** **Example:** `RP/0/RSP0/CPU0:routertype rtr 100 reachability`	Sets the ipsla operation id which needs to be tracked for reachability.
Step 4	**commit**	

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*] [**dscp** *dscp*] [**fragments**] [**log** | **log-input**]] [**ttl** *ttl value* [*value1 ... value2*]][**default] nexthop1** [**track** *track-name-1*] [**vrf** *vrf-name1*][**ipv6** *ipv6-address1*] [**nexthop2** [**track** *track-name-2*] [**vrf** *vrf-name2*] [**ipv6** *ipv6-address2*] [**nexthop3** [**track** *track-name-3*] [**vrf** *vrf-name3*] [**ipv6***ipv6-address3*]]]
4. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	

	Command or Action	Purpose
Step 2	{**ipv6** } **access-list** *name* **Example:** `RP/0/RSP0/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*] [**dscp** *dscp*] [**fragments**] [**log** | **log-input**]] [**ttl** *ttl value* [*value1 ... value2*]][**default] nexthop1** [**track** *track-name-1*] [**vrf** *vrf-name1*][**ipv6** *ipv6-address1*] [**nexthop2** [**track** *track-name-2*] [**vrf** *vrf-name2*] [**ipv6** *ipv6-address2*] [**nexthop3** [**track** *track-name-3*] [**vrf** *vrf-name3*] [**ipv6***ipv6-address3*]]] **Example:** `RP/0/RSP0/CPU0:router(config-ipv6-acl)# 10 permit ipv6 any any default nexthop1 vrf vrf_A ipv6 11::1 nexthop2 vrf vrf_B ipv6 nexthop3 vrf vrf_C ipv6 33::3`	Sets the conditions for an IPv6 access list. The configuration example shows how to configure pure ACL-based forwarding for ACL. • Forwards the specified next hop for this entry. • The track option specifies object tracking name for the corresponding next hop.
Step 4	**commit**	

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. This feature is supported on A9K-SIP-700 Line Card and ASR 9000 Enhanced Ethernet Line Card only.

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.
- The IPv4 and IPv6 common ACL is limited to 200 Ternary Content Addressable Memory(TCAM) entries for the ASR 9000 Enhanced Ethernet line card and A9K-SIP-700 line card. Although, A9K-SIP-700 line card may support more.
- Common ACL is not supported on Cisco ASR 9000 12-Port 100GE Line Card, ASR 9000 Ethernet Line Card and ASR 9000 Enhanced Ethernet-TR Line Card.
- 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.
- Object-groups are not supported with common ACLs.
- The **interface-statistics** and **hardware-count** options are not supported for ACLs on the A9K-SIP-700 line card.

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**	

	Command or Action	Purpose
Step 2	**interface** *type interface-path-id* **Example:** `RP/0/RSP0/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/RSP0/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. **Note** The **interface-statistics** and **hardware-count** options are not supported for ACLs on the A9K-SIP-700 line card.
Step 4	**commit**	

Configuring an Interface to Accept Multiple ACLs on Cisco ASR 9000 High Density 100GE Ethernet Line Cards

You can configure an interface on Cisco ASR 9000 High Density 100GE Ethernet line cards (such as A9K-8x100G-LB-SE and A9K-8x100G-LB-TR) to accept up to five IPv4 and/or IPv6 ACLs. This feature extends the ACL chaining from two ACLs to a maximum of five ACLs on Cisco ASR 9000 High Density 100GE Ethernet line cards only.

The following restrictions apply while configuring multiple ACLs on Cisco ASR 9000 High Density 100GE Ethernet line cards:

- This feature is not supported on Cisco ASR 9000 12-Port 100GE Line Card.
- Multi-level ACL is supported only in the ingress direction and for L3 interfaces only.
- The multi-level ACL feature is not available for Ethernet Service (ES) ACLs.
- ACLs with obj-groups are not supported.
- Compression is not supported.
- Access List Based Forwarding (ABF) enabled rules for ACLs are not supported.
- An ACL already applied in per-ace mode cannot be applied elsewhere in interface-stats mode.
- Accessing the ACL counters using SNMP query is not supported.

Perform this task to configure an interface on Cisco ASR 9000 High Density 100GE Ethernet line cards to accept up to five IPv4 and/or IPv6 ACLs:

SUMMARY STEPS

1. **configure**

2. **interface** *type interface-path-id*
3. [**ipv4 | ipv6**] **access-group common** *acl-c1* **common** *acl-c2 acl-i2 acl-i4 acl-i5* **ingress**
4. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config)# interface GigabitEthernet 0/1/0/0`	This command configures an interface (in this case a GigabitEthernet interface) and enters the interface configuration mode.
Step 3	[**ipv4 | ipv6**] **access-group common** *acl-c1* **common** *acl-c2 acl-i2 acl-i4 acl-i5* **ingress** **Example:** `RP/0/RSP0/CPU0:router(config-if)# ipv4 access-group common acl-a common acl-b acl-x acl-y acl-z ingress`	Configures the interface to accept five ACLs in the inbound direction. There can be any combination of common and/or interface ACLs up to a total of five ACLs. In this command: • "*acl_c1* and "*acl_c2*" are common ACLs, each preceded by the "**common**" keyword • "*acl_i2*", "*acl_i4*," and "*acl_i5*" are interface ACLs
Step 4	**commit**	

ACL Scale Enhancements

The Access Control List (ACL) Scale enhancements feature enables you to define ACL rules as a set of several rules (super-set of ACEs (Access Control Entry)). This is achieved with object-groups of prefixes and ports, which are referred by ACE in the same way as single source address or destination address prefix and ports are referred.

Note The ACL Scale enhancements feature is not supported on first generation ASR 9000 Ethernet Line Card.

ACL Scale Enhancements: Backward Compatibility

With the support of object-groups, configuring ACE in the existing way in which one ACE entry uses object groups, while another ACE entry does not use object groups is supported.

Note From Release 4.3.1, object group is only supported on ASR 9000 Enhanced Ethernet Line Card.

```
ipv4 access-list acl1
     10 permit tcp  net-group  group1  host 10.10.10.1  eq 2200
```

```
     20 permit tcp 10.10.10.3/32  host 1.1.1.2   eq  2000
!
```

It is possible that a user configures a host or prefix in an ACE entry, where the same host or prefix is added to an existing source group, eliminating the need to configure a separate ACE entry. However, such an optimization is not automated. A user could intentionally configure a particular prefix in a separate ACE for the purpose of separate counter or accounting for that prefix.

Configuring a Network Object-Group

Perform this task to configure a network object group and to enter the network object group configuration mode.

SUMMARY STEPS

1. **configure**
2. **object-group network** { **ipv4 | ipv6** } *object-group-name*
3. **description** *description*
4. **host** *address*
5. *address { mask | prefix }*
6. **range** *address address*
7. **object-group** *name*
8. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**object-group network** { **ipv4 | ipv6** } *object-group-name* **Example:** `RP/0/RSP0/CPU0:router(config)# object-group network ipv4 ipv4_type5_obj1`	Configures a network object group and enters the network object group configuration mode.
Step 3	**description** *description* **Example:** `RP/0/RSP0/CPU0:router(config-object-group-ipv4)# description network-object-group`	Describes the object group.
Step 4	**host** *address* **Example:** `RP/0/RSP0/CPU0:router(config-object-group-ipv4)# host 10.20.2.3`	Configures the host IPv4 address for the object group.

	Command or Action	Purpose
Step 5	*address { mask | prefix }* **Example:** `RP/0/RSP0/CPU0:router(config-object-group-ipv4)# 10.20.20.3 255.255.255.0`	Configures the host address mask or prefix.
Step 6	**range** *address address* **Example:** `RP/0/RSP0/CPU0:router(config-object-group-ipv4)# range 10.20.20.10 10.20.20.40`	Configures the range of host IPv4 address for the object group.
Step 7	**object-group** *name* **Example:** `RP/0/RSP0/CPU0:router(config-object-group-ipv4)# object-group`	Specifies the name of the nested object group.
Step 8	**commit**	

Configuring a Port Object-Group

Perform this task to configure a port object group and to enter the port object group configuration mode.

SUMMARY STEPS

1. **configure**
2. **object-group port** *object-group-name*
3. **description** *description*
4. **{ eq | lt | gt }***{ protocol | number }*
5. **range** *range range*
6. **object-group** *name*
7. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**object-group port** *object-group-name* **Example:**	Configures a port object group and enters the port object group configuration mode.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config)# object-group port ipv4_type5_obj1`	
Step 3	**description** *description* **Example:** `RP/0/RSP0/CPU0:router(config-object-group-port)# description port-object-group`	Configures the description for the object group.

Step 4	**{ eq | lt | gt }***{ protocol | number }*
	Example:
	`RP/0/RSP0/CPU0:router(config-object-group-port)# eq ftp`
	`or`
	`RP/0/RSP0/CPU0:router(config-object-group-port)# eq 21`
		Matches packets on ports equal to, less than, or greater than the specified port number or protocol.

Step 5	**range** *range range* **Example:** `RP/0/RSP0/CPU0:router(config-object-group-port)# range 1000 2000`	Configures the range of host ports for the object group.
Step 6	**object-group** *name* **Example:** `RP/0/RSP0/CPU0:router(config-object-group-port)# object-group port-group2`	Specifies the name of the nested object group.
Step 7	**commit**	

Configuring ACL with Object-Groups

You must be aware of the following information that apply to object-group ACLs:

- You can configure ACLs that contain both conventional and object-group ACEs.
- You can modify the objects in an object group dynamically without redefining the object group or the ACE that references the object group.
- You can configure an object-group ACL multiple times with a source group, or a destination group, or both source and destination groups.

Configuring object-group ACLs involves the following restrictions:

- Object-group ACLs can only be configured to an interface. They cannot be used or referenced by applications like SSH, SNMP, NTP.
- To delete an object-group, you must first delete it from all ACLs.
- You cannot configure object-group ACLs along with QoS policies.
- Object-group ACLs are not supported in any policy based configuration.

Perform this task to configure ACL with object groups.

SUMMARY STEPS

1. **configure**
2. **{ ipv4 | ipv6 } access-list** *name*
3. [*sequence-number*] **permit** *protocol* **net-group** *source-net-object-group-name* **port-group** *source-port-object-group-name* **net-group** *destination-net-object-group-name* **port-group** *destination-port-object-group-name* [**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 range** *dscp dscp*] [**fragments**] [*packet-length operator packet-length value*] [**log** | **log-input**] [[**track** *track-name*] [**ttl** *ttl* [*value1 ... value2*]]
4. **exit**
5. **interface** *type interface-path-id*
6. **ipv4 access-group** *access-list-name* {**ingress** | **egress** } **compress level** *level* [**hardware-count**] [**interface-statistics**]
7. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**{ ipv4 | ipv6 } access-list** *name* **Example:** `RP/0/RSP0/CPU0:router(config)# ipv4 access-list acl1`	Configures the specified access list, and enters IPv4 or IPv6 access list configuration mode.

	Command or Action	Purpose

Step 3	[*sequence-number*] **permit** *protocol* **net-group** *source-net-object-group-name* **port-group** *source-port-object-group-name* **net-group** *destination-net-object-group-name* **port-group** *destination-port-object-group-name* [**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 range** *dscp dscp*] [**fragments**] [*packet-length operator packet-length value*] [**log** | **log-input**] [[**track** *track-name*] [**ttl** *ttl* [*value1 ... value2*]]
	Example:
	`RP/0/RSP0/CPU0:router(config-ipv4-acl)# 10 permit tcp net-group network-group-west net-group network-group-east port-group`
	`RP/0/RSP0/CPU0:router(config-ipv4-acl)# 20 permit ipv4 net-group network-group-west1 net-group network-group-east1`
		Configures ACL with object groups.
		Note You must configure network object groups and port object groups before configuring ACL. For more information about configuring network object groups, see Configuring a Network Object-Group, on page 36. For more information about configuring port object groups, see Configuring a Port Object-Group, on page 37.
		When a network or port object-group is part of an ACL attached to an interface, you can add or remove members from the corresponding network or port object-group.
		When a network or port object-group is part of an ACL attached to an interface, adding or removing object-groups which are part of inherited or nested object-groups is not supported.
		A member is either an IPv4/IPv6 address/prefix or port.

Step 4	**exit** **Example:** `RP/0/RSP0/CPU0:router(config-ipv4-acl)# exit`	Returns to global configuration mode.
Step 5	**interface** *type interface-path-id* **Example:** `RP/0/RSP0/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. • The *interface-path-id* argument specifies either a physical interface instance or a virtual instance.The *interface-path-id* argument specifies either a physical interface instance or a virtual instance.

Step 6	**ipv4 access-group** *access-list-name* {**ingress** | **egress** } **compress level** *level* [**hardware-count**] [**interface-statistics**]
	Example:
	`RP/0/RSP0/CPU0:router(config-if)# ipv4 access-group acl1 ingress compress level 1`
	`RP/0/RSP0/CPU0:router(config-if)# ipv4 access-group`
		Controls access to an interface. Use the **compress level** keyword to specify ACL compression in the hardware.
		• level 0 indicates no compression
		• level 1 indicates source compression
		• level 3 indicates all compression

	Command or Action	Purpose
	`acl1 engress compress level 3`	
Step 7	**commit**	

Atomic ACL Updates By Using the Disable Option

Atomic ACL updates involve the insertion, modification, or removal of Access List Entries (ACEs) on an interface that is in operation. Such atomic updates consume up to 50% of TCAM resources. There can be an instance where multiple modifications are required and the available resources are not sufficient. The solution to this problem is to disable atomic ACL updates such that the old ACEs are deleted before the new ACEs are added.

Note When you configure the **atomic-disable** statement in an ACL, any ACE modification detaches the ACL, until the modification is complete. In addition to this, the ACL rules are not applied during the modification process. Hence, it is recommended to configure to either permit or deny all traffic until the modification is complete.

Configuration for Disabling Atomic ACL Updates

To disable atomic updates on the hardware, by permitting all packets, use the following configuration.

```
RP/0/RSP0/CPU0:router# hardware access-list atomic-disable
```

Modifying ACLs when Atomic ACL Updates are Disabled

On disabling atomic ACL updates on the hardware, use the steps in this section to modify ACLs.

Add an ACE

Use the following steps to add an ACE.

1. Locate the ACL you want to modify.

```
RP/0/RSP0/CPU0:router(config)# do show access-lists
...
!
ipv4 access-list list1
 10 permit ipv4 10.1.1.0/24 any
 20 permit ipv4 20.1.1.0/24 any
!
```

2. Add the ACE to the ACL.

```
RP/0/RSP0/CPU0:router(config)# ipv4 access-list list1
RP/0/RSP0/CPU0:router(config-ipv4-acl)# 30 permit ipv4 30.1.1.0/24 any
RP/0/RSP0/CPU0:router(config-ipv4-acl)# commit
```

3. Verify if your ACE was added successfully.

```
RP/0/RSP0/CPU0:router(config)# do show access-lists
...
!
ipv4 access-list list1
 10 permit ipv4 10.1.1.0/24 any
 20 permit ipv4 20.1.1.0/24 any
 30 permit ipv4 30.1.1.0/24 any
!
```

You have successfully added an ACE.

Delete an ACE

Use the steps in this section to delete an ACE.

1. Locate the ACL containing the ACE that you want deleted.

```
RP/0/RSP0/CPU0:router(config)# do show access-lists
...
!
ipv4 access-list list1
 10 permit ipv4 10.1.1.0/24 any
 20 permit ipv4 20.1.1.0/24 any
 30 permit ipv4 30.1.1.0/24 any
!
```

2. Delete the ACE.

```
RP/0/RSP0/CPU0:router(config)# ipv4 access-list list1
RP/0/RSP0/CPU0:router(config-ipv4-acl)# no 30
RP/0/RSP0/CPU0:router(config-ipv4-acl)# commit
```

3. Verify if the ACE has been removed from the ACL.

```
RP/0/RSP0/CPU0:router(config-ipv4-acl)# do show access-lists
...
ipv4 access-list list1
 10 permit ipv4 10.1.1.0 0.0.0.255 any
 20 permit ipv4 20.1.1.0 0.0.0.255 any
```

You have successfully deleted an ACE.

Replace an ACE

Use the steps in this section to replace an ACE.

1. Locate the ACL you want to modify.

```
RP/0/RSP0/CPU0:router(config-ipv4-acl)#do show access-lists
...
ipv4 access-list list1
 10 permit ipv4 10.1.1.0 0.0.0.255 any
 20 permit ipv4 20.1.1.0 0.0.0.255 any
```

2. Configure the new ACE to replace the existing ACE.

```
RP/0/RSP0/CPU0:router(config)# ipv4 access-list list1
RP/0/RSP0/CPU0:router(config-ipv4-acl)# 10 permit ipv4 11.1.1.0/24 any
RP/0/RSP0/CPU0:router(config-ipv4-acl)# commit
```

3. Verify if the ACE replacement is successful.

```
RP/0/RSP0/CPU0:router(config)# do show access-lists
...
```

```
ipv4 access-list list1
 10 permit ipv4 11.1.1.0 0.0.0.255 any
 20 permit ipv4 20.1.1.0 0.0.0.255 any
```

You have successfully replaced an ACE.

Delete an ACE and Add a New ACE

Use the following steps to delete an ACE and add a new ACE.

1. Locate the ACL you want to modify.

```
RP/0/RSP0/CPU0:router(config)# do show access-lists
...
ipv4 access-list list1
 10 permit ipv4 11.1.1.0 0.0.0.255 any
 20 permit ipv4 20.1.1.0 0.0.0.255 any
```

2. Delete the required ACE, and add the new ACE.

```
RP/0/RSP0/CPU0:router(config)# ipv4 access-list list1
RP/0/RSP0/CPU0:router(config-ipv4-acl)# no 20
RP/0/RSP0/CPU0:router(config-ipv4-acl)# permit ipv4 12.1.1.0/24 any
RP/0/RSP0/CPU0:router(config-ipv4-acl)# commit
```

3. Verify if the modification is successful.

```
RP/0/RSP0/CPU0:router(config)# do show access-lists
...
ipv4 access-list list1
 10 permit ipv4 11.1.1.0 0.0.0.255 any
 20 permit ipv4 12.1.1.0 0.0.0.255 any
```

You have successfully deleted an ACE, and added a new ACE.

Similarly, you can combine the addition, removal, and replacement of ACEs.

Configuring ACL Counters for SNMP Query

You can configure ACL counters and access the counters using SNMP query. This section explains how to configure ACL counters for SNMP query.

SUMMARY STEPS

1. **configure**
2. {**ipv4** | **ipv6**} **access-list** *name*
3. Do one of the following:
   - [*sequence-number*] {**permit** | **deny**} *source* {[*source source-wildcard*] | [*destination destination-wildcard*]} **counter** *counter-name*
   - [*sequence-number*] {**permit** | **deny**} *protocol* {[*source-ipv6-prefix*/*prefix-length* | **any** | **host** *source-ipv6-address* | [*destination-ipv6-prefix*/*prefix-length* | **any** | **host** *destination-ipv6-address*]} **counter** *counter-name*
4. Repeat Step 3 as necessary, adding statements by sequence number where you planned. Use the **no** *sequence-number* command to delete an entry.
5. **commit**

6. **show access-lists** {**ipv4** | **ipv6**} [*access-list-name*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	

Step 2	{**ipv4** | **ipv6**} **access-list** *name*
	Example:
	`RP/0/RSP0/CPU0:router(config)# ipv4 access-list acl_1`
	or
	`RP/0/RSP0/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	Do one of the following:
	• [*sequence-number*] {**permit** | **deny**} *source* {[*source source-wildcard*] | [*destination destination-wildcard*]} **counter** *counter-name*
	• [*sequence-number*] {**permit** | **deny**} *protocol* {[*source-ipv6-prefix*/*prefix-length* | **any** | **host** *source-ipv6-address* | [*destination-ipv6-prefix*/*prefix-length* | **any** | **host** *destination-ipv6-address*]} **counter** *counter-name*
	Example:
	`RP/0/RSP0/CPU0:router(config-ipv4-acl)# 10 permit 172.16.0.0 0.0.255.255 counter counter1`
	`RP/0/RSP0/CPU0:router(config-ipv4-acl)# 20 deny 192.168.34.0 0.0.0.255 counter counter2`
	or
	`RP/0/RSP0/CPU0:router(config-ipv6-acl)# 20 permit icmp any any counter counter3`
	`RP/0/RSP0/CPU0:router(config-ipv6-acl)# 30 deny tcp any any gt 5000 counter counter4`
		Specifies one or more conditions allowed or denied in IPv4 access list acl_1 or IPv6 access list acl_2.
		The **counter** *counter-name* keyword enables ACL counters which you can access using SNMP query.

Step 4	Repeat Step 3 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 5	**commit**	
Step 6	**show access-lists** {**ipv4** | **ipv6**} [*access-list-name*] **Example:** `RP/0/RSP0/CPU0:router# show access-lists ipv4 acl_1`	(Optional) Displays the contents of current IPv4 or IPv6 access lists.

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 10, and increment value is 20. 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
 ipv4 access-list acl_1
 end
resequence ipv4 access-list acl_1 10 20

show access-lists ipv4 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 permit ip host 172.16.2.2 host 10.3.3.12
110 permit ip host 10.3.3.3 any log
130 permit tcp host 10.3.3.3 host 10.1.2.2
150 permit ip any any


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 10 .1.1.1 0.0.0.255
permit 10 .2.2.2 0.0.0.255
permit 10 .3.3.3 0.0.0.255
end

ipv4 access-list acl_10
 10 permit ip 10 .1.1.0 0.0.0.255 any
 20 permit ip 10 .2.2.0 0.0.0.255 any
 30 permit ip 10 .3.3.0 0.0.0.255 any

configure
ipv4 access-list acl_10
permit 10 .4.4.4 0.0.0.255
end

ipv4 access-list acl_10
 10 permit ip 10 .1.1.0 0.0.0.255 any
 20 permit ip 10 .2.2.0 0.0.0.255 any
 30 permit ip 10 .3.3.0 0.0.0.255 any
 40 permit ip 10 .4.4.0 0.0.0.255 any
```

Atomic ACL Updates By Using the Disable Option

Atomic ACL updates involve the insertion, modification, or removal of Access List Entries (ACEs) on an interface that is in operation. Such atomic updates consume up to 50% of TCAM resources. There can be an instance where multiple modifications are required and the available resources are not sufficient. The solution to this problem is to disable atomic ACL updates such that the old ACEs are deleted before the new ACEs are added.

Note When you configure the **atomic-disable** statement in an ACL, any ACE modification detaches the ACL, until the modification is complete. In addition to this, the ACL rules are not applied during the modification process. Hence, it is recommended to configure to either permit or deny all traffic until the modification is complete.

Configuration for Disabling Atomic ACL Updates

To disable atomic updates on the hardware, by permitting all packets, use the following configuration.

```
RP/0/RSP0/CPU0:router# hardware access-list atomic-disable
```

Modifying ACLs when Atomic ACL Updates are Disabled

On disabling atomic ACL updates on the hardware, use the steps in this section to modify ACLs.

Add an ACE

Use the following steps to add an ACE.

1. Locate the ACL you want to modify.

```
RP/0/RSP0/CPU0:router(config)# do show access-lists
...
!
ipv4 access-list list1
 10 permit ipv4 10.1.1.0/24 any
 20 permit ipv4 20.1.1.0/24 any
!
```

2. Add the ACE to the ACL.

```
RP/0/RSP0/CPU0:router(config)# ipv4 access-list list1
RP/0/RSP0/CPU0:router(config-ipv4-acl)# 30 permit ipv4 30.1.1.0/24 any
RP/0/RSP0/CPU0:router(config-ipv4-acl)# commit
```

3. Verify if your ACE was added successfully.

```
RP/0/RSP0/CPU0:router(config)# do show access-lists
...
!
ipv4 access-list list1
 10 permit ipv4 10.1.1.0/24 any
 20 permit ipv4 20.1.1.0/24 any
 30 permit ipv4 30.1.1.0/24 any
!
```

You have successfully added an ACE.

Delete an ACE

Use the steps in this section to delete an ACE.

1. Locate the ACL containing the ACE that you want deleted.

```
RP/0/RSP0/CPU0:router(config)# do show access-lists
...
!
ipv4 access-list list1
 10 permit ipv4 10.1.1.0/24 any
 20 permit ipv4 20.1.1.0/24 any
 30 permit ipv4 30.1.1.0/24 any
!
```

2. Delete the ACE.

```
RP/0/RSP0/CPU0:router(config)# ipv4 access-list list1
RP/0/RSP0/CPU0:router(config-ipv4-acl)# no 30
RP/0/RSP0/CPU0:router(config-ipv4-acl)# commit
```

3. Verify if the ACE has been removed from the ACL.

```
RP/0/RSP0/CPU0:router(config-ipv4-acl)# do show access-lists
...
ipv4 access-list list1
 10 permit ipv4 10.1.1.0 0.0.0.255 any
 20 permit ipv4 20.1.1.0 0.0.0.255 any
```

You have successfully deleted an ACE.

Replace an ACE

Use the steps in this section to replace an ACE.

1. Locate the ACL you want to modify.

```
RP/0/RSP0/CPU0:router(config-ipv4-acl)#do show access-lists
...
ipv4 access-list list1
 10 permit ipv4 10.1.1.0 0.0.0.255 any
 20 permit ipv4 20.1.1.0 0.0.0.255 any
```

2. Configure the new ACE to replace the existing ACE.

```
RP/0/RSP0/CPU0:router(config)# ipv4 access-list list1
RP/0/RSP0/CPU0:router(config-ipv4-acl)# 10 permit ipv4 11.1.1.0/24 any
RP/0/RSP0/CPU0:router(config-ipv4-acl)# commit
```

3. Verify if the ACE replacement is successful.

```
RP/0/RSP0/CPU0:router(config)# do show access-lists
...
ipv4 access-list list1
 10 permit ipv4 11.1.1.0 0.0.0.255 any
 20 permit ipv4 20.1.1.0 0.0.0.255 any
```

You have successfully replaced an ACE.

Delete an ACE and Add a New ACE

Use the following steps to delete an ACE and add a new ACE.

1. Locate the ACL you want to modify.

```
RP/0/RSP0/CPU0:router(config)# do show access-lists
...
ipv4 access-list list1
 10 permit ipv4 11.1.1.0 0.0.0.255 any
 20 permit ipv4 20.1.1.0 0.0.0.255 any
```

2. Delete the required ACE, and add the new ACE.

```
RP/0/RSP0/CPU0:router(config)# ipv4 access-list list1
RP/0/RSP0/CPU0:router(config-ipv4-acl)# no 20
RP/0/RSP0/CPU0:router(config-ipv4-acl)# permit ipv4 12.1.1.0/24 any
RP/0/RSP0/CPU0:router(config-ipv4-acl)# commit
```

3. Verify if the modification is successful.

```
RP/0/RSP0/CPU0:router(config)# do show access-lists
...
ipv4 access-list list1
 10 permit ipv4 11.1.1.0 0.0.0.255 any
 20 permit ipv4 12.1.1.0 0.0.0.255 any
```

   You have successfully deleted an ACE, and added a new ACE.

Similarly, you can combine the addition, removal, and replacement of ACEs.

IPv4/IPv6 ACL over BVI interface

In Release 4.2.1, IPv4/IPv6 ACL is enabled over BVI interfaces on the ASR 9000 Enhanced Ethernet Line Cards.

For ACL over BVI interfaces, the defined direction is:

- L2 interface - ingress direction
- L3 interface - egress direction

On the A9K-SIP-700 and ASR 9000 Ethernet Line Cards, ACLs on BVI interfaces are not supported.

Note For ASR 9000 Ethernet linecards, ACL can be applied on the EFP level (IPv4 L3 ACL can be applied on an L2 interface).

Configuring IPv4 ACL over BVI interface - An Example

This example shows how to configure IPv4 ACL over a BVI interface:

```
ipv4 access-list bvi-acl
10 permit ipv4 any any ttl eq 70
```

```
20 deny ipv4 any any ttl eq 60
```

Configuring ABFv4/v6 over IRB/BVI interface

Perform this task to configure ABF (access-list based forwarding) v4/v6 over Integrated Routing and Bridging(IRB) or Bridge-Group Virtual Interface (BVI) interface:

SUMMARY STEPS

1. **configure**
2. **ipv4 access-list** *access-list-name*
3. [*sequence-number*] **permit** *protocol source source-wildcard destination destination-wildcard* **nexthop1** [**vrf** *vrf-name*][**ipv4** *ipv4-address1*] **nexthop2** [**vrf** *vrf-name*] [**ipv4** *ipv4-address2*] **nexthop3** [**vrf** *vrf-name*] [**ipv4** *ipv4-address3*]
4. **exit**
5. **ipv6 access-list** *access-list-name*
6. [*sequence-number*] **permit** *protocol source source-wildcard destination destination-wildcard* **nexthop1** [**vrf** *vrf-name*][**ipv6** *ipv6-address1*] **nexthop2** [**vrf** *vrf-name*] [**ipv6** *ipv6-address2*] **nexthop3** [**vrf** *vrf-name*] [**ipv6** *ipv6-address3*]
7. **exit**
8. **interface** *type interface-path-id*
9. { **ipv4** | **ipv6**} **address** *address {network-mask* | *ipv6-prefix}*
10. { **ipv4** | **ipv6**} **access-group** *access-list-name* **{ingress** | **egress}**
11. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**ipv4 access-list** *access-list-name* **Example:** `RP/0/RSP0/CPU0:router(config)# ipv4 acess-list abf-v4`	Enters the IPv4 access list configuration mode, and configures the named access list.
Step 3	[*sequence-number*] **permit** *protocol source source-wildcard destination destination-wildcard* **nexthop1** [**vrf** *vrf-name*][**ipv4** *ipv4-address1*] **nexthop2** [**vrf** *vrf-name*] [**ipv4** *ipv4-address2*] **nexthop3** [**vrf** *vrf-name*] [**ipv4** *ipv4-address3*] **Example:** `RP/0/RSP0/CPU0:router(config-ipv4-acl)# 10 permit ipv4 any any nexthop1 ipv4 192.168.1.20 nexthop2`	Configures the permit conditions for an IPv4 access list.

	Command or Action	Purpose
	ipv4 192.168.9.2 nexthop3 ipv4 192.168.10.2	

Step 4	**exit**
	Example:
	`RP/0/RSP0/CPU0:router(config-ipv4-acl)# exit`
	`RP/0/RSP0/CPU0:router(config)#`
		Returns to global configuration mode.

Step 5	**ipv6 access-list** *access-list-name* **Example:** `RP/0/RSP0/CPU0:router(config)# ipv6 acess-list abf-v6`	Enters the IPv6 access list configuration mode, and configures the named access list.
Step 6	[*sequence-number*] **permit** *protocol source source-wildcard destination destination-wildcard* **nexthop1** [**vrf** *vrf-name*][**ipv6** *ipv6-address1*] **nexthop2** [**vrf** *vrf-name*] [**ipv6** *ipv6-address2*] **nexthop3** [**vrf** *vrf-name*] [**ipv6** *ipv6-address3*] **Example:** `RP/0/RSP0/CPU0:router(config-ipv6-acl)# 10 permit ipv6 any any nexthop1 ipv6 5001:5001::2 nexthop2 ipv6 9001:9001::2 nexthop3 ipv6 1901:1901::2`	Configures the permit conditions for an IPv6 access list.

Step 7	**exit**
	Example:
	`RP/0/RSP0/CPU0:router(config-ipv4-acl)# exit`
	`RP/0/RSP0/CPU0:router(config)#`
		Returns to global configuration mode.

Step 8	**interface** *type interface-path-id* **Example:** `RP/0/RSP0/CPU0:router(config)# interface BVI 18`	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. • 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 (/)

	Command or Action	Purpose
		between values is required as part of the notation. • The number range for a virtual interface instance varies depending on the interface type.

Step 9	{ **ipv4 | ipv6**} **address** *address {network-mask | ipv6-prefix}*
	Example:
	`RP/0/RSP0/CPU0:router(config-if)#ipv4 address 192.168.18.1 255.255.255.0`
	`or`
	`RP/0/RSP0/CPU0:router(config-if)#ipv6 address 1801:1801::1/64`
		Configures the primary IPv4 address or IPv6 address for an interface.
		The network mask can be specified in either of two ways:
		• The network mask can be a four-part dotted decimal address. For example, 255.0.0.0 indicates that each bit equal to 1 means the corresponding address bit belongs to the network address.
		• The network mask can be indicated as a slash (/) and number. For example, /8 indicates that the first 8 bits of the mask are ones, and the corresponding bits of the address are network address.
		This ipv6-prefix must be in the form documented in RFC 2373 where the address is specified between colons in hexadecimal using 16-bit values.

Step 10	{ **ipv4 | ipv6**} **access-group** *access-list-name* **{ingress | egress}**
	Example:
	`RP/0/RSP0/CPU0:router(config-if)# ipv4 access-group abfv4 ingress`
	`or`
	`RP/0/RSP0/CPU0:router(config-if)# ipv6 access-group abfv6 ingress`
		Controls access to an interface. The **ipv6 access-group** command is similar to the **ipv4 access-group** command, except that it is IPv6-specific. Use the *access-list-name* to specify a particular IPv6 access list. Use the **ingress** keyword to filter on inbound packets or the **egress** keyword to filter on outbound packets.

Step 11	**commit**	

Configuring ABFv4 over IRB/BVI interface: Example

This example shows how to configure ABFv4 over Integrated Routing and Bridging (IRB)/Bridge-Group Virtual Interface (BVI) interface:

```
interface BVI18
 ipv4 address 192.168.18.1 255.255.255.0
 ipv4 access-group abfv4 ingress
!


l2vpn
 bridge group bg18
```

```
  bridge-domain bd18
   interface GigabitEthernet0/0/1/18
   !
   routed interface BVI18
  !
 !
!

ipv4 access-list abfv4
 10 permit ipv4 any any nexthop1 ipv4 192.168.1.20 nexthop2 ipv4 192.168.9.2 nexthop3 ipv4
 192.168.10.2
!
```

Configuring ABFv6 over IRB/BVI interface: Example

This example shows how to configure ABFv6 over Integrated Routing and Bridging (IRB) or Bridge-Group Virtual Interface (BVI) interface:

```
interface BVI18
 ipv4 address 192.168.18.1 255.255.255.0
 ipv6 address 1801:1801::1/64
 ipv4 access-group abfv4 ingress
 ipv6 access-group abfv6 ingress
!

l2vpn
 bridge group bg18
  bridge-domain bd18
   interface GigabitEthernet0/0/1/18
   !
   routed interface BVI18
  !
 !
!
ipv4 access-list abfv4
 10 permit ipv4 any any nexthop1 ipv4 192.168.1.20 nexthop2 ipv4 192.168.9.2 nexthop3 ipv4
 192.168.10.2
!
ipv4 access-list ipv4-abf
 10 permit ipv4 any any nexthop1 vrf 1 ipv4 45.45.45.2
!

ipv6 access-list ipv6-abf
 10 permit ipv6 any any nexthop1 vrf 1 ipv6 2040::2
!
ipv6 access-list ipv6-vrf
 10 permit ipv6 2001::1/64 any nexthop1 ipv6 2075::2
!
ipv6 access-list abfv6-bvi
 10 permit ipv6 any any nexthop1 ipv6 5001:5001::2 nexthop2 ipv6 9001:9001::2 nexthop3 ipv6
 1901:1901::2
!
ipv6 access-list ipv6-thor
 10 permit ipv6 any any nexthop1 vrf 4 ipv6 2001::2
!
```

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
```

Configuring ACL Counters for SNMP Query: Example

The following example shows how to configure IPv4 ACL counters for SNMP query.

```
configure
ipv4 access-list CounterExample
permit any ?
  counter    Count matches on this entry
  log        Log matches against this entry
  log-input  Log matches against this entry, including input interface
permit any counter ?
  WORD  Name of counter
permit any counter TestCounter
show configuration

Building configuration...
!! IOS XR Configuration 0.0.0
ipv4 access-list CounterExample
 10 permit ipv4 any any counter TestCounter
permit tcp any any counter TestCounter2

show configuration
Building configuration...
!! IOS XR Configuration 0.0.0
ipv4 access-list CounterExample
 10 permit ipv4 any any counter TestCounter
 20 permit tcp any any counter TestCounter2

commit

show access-lists ipv4 CounterExample

ipv4 access-list CounterExample
 10 permit ipv4 any any counter TestCounter
 20 permit tcp any any counter TestCounter2
```

The following example shows how to configure IPv6 ACL counters for SNMP query.

```
conf igure
ipv6 access-list V6CounterExample
permit tcp any any counter ?
  WORD  Name of counter
permit tcp any any counter TestCounter6
```

```
show configconfiguration
Building configuration...
!! IOS XR Configuration 0.0.0
ipv6 access-list V6CounterExample
 10 permit tcp any any counter TestCounter6

commit

show access-lists ipv6 V6CounterExample

ipv6 access-list V6CounterExample
 10 permit tcp any any counter TestCounter6
```

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 ASR 9000 Series 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 ASR 9000 Series 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 ASR 9000 Series 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