Implementing Access Lists and Prefix Lists

Understanding 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:

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 software features such as traffic filtering, route filtering, QoS classification, and access control. There are 2 types of ACLs:

- Standard ACLs- Verifies only the source IP address of the packets. Traffic is controlled by the comparison of the address or prefix configured in the ACL, with the source address found in the packet.
- Extended ACLs- Verifies more than just the source address of the packets. Attributes such as destination address, specific IP protocols, UDP or TCP port numbers, DSCP, and so on are validated. Traffic is controlled by a comparison of the attributes stated in the ACL with those in the incoming or outgoing packets.

Cisco IOS XR does not differentiate between standard and extended access lists. Standard access list support is provided for backward compatibility.

Purpose of IP Access Lists

- Filter incoming or outgoing packets on an interface.
- Filter packets for mirroring.
- Redirect traffic as required.
- 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.

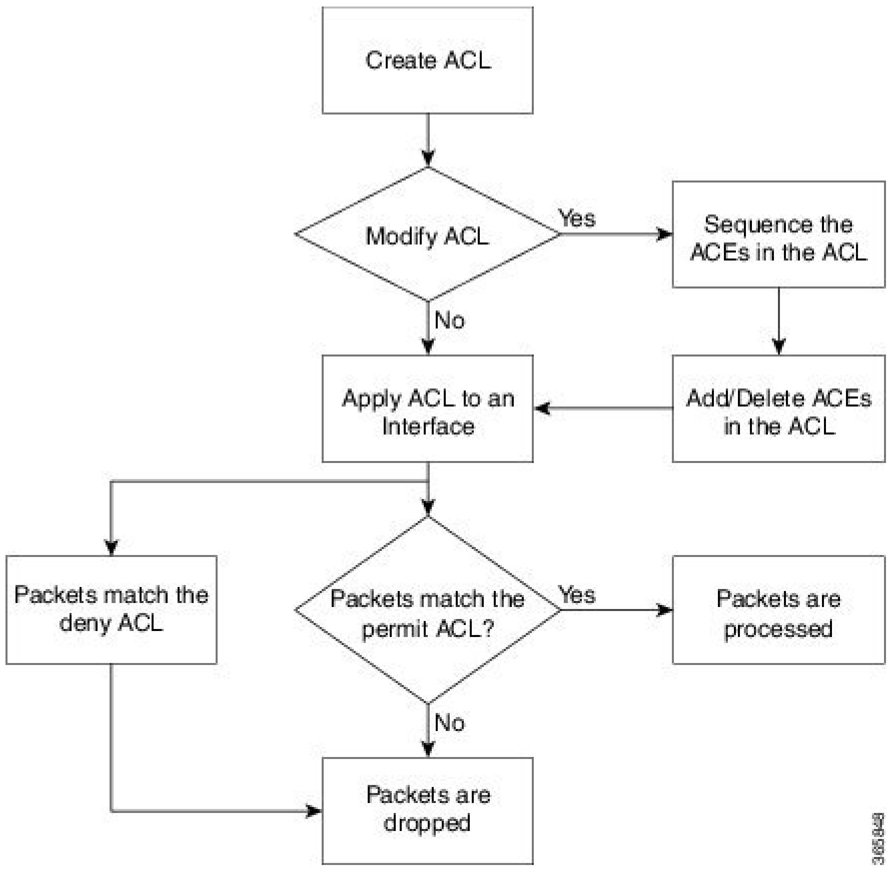
An access list can be configured and named, but it is not in effect until the access list is referenced by a command that accepts an access list. Multiple commands can reference the same access list. An access list can control traffic arriving at the router or leaving the router, but not traffic originating at the router.

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.

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

ACL Workflow

The following image illustrates the workflow of an ACL.


Create ACL
Modify ACL
Yes
Sequence the ACEs in the ACL
No
Apply ACL to an Interface
Add/Delete ACEs in the ACL
Packets match the deny ACL
Packets match the permit ACL?
Yes
Packets are processed
No
Packets are dropped
365848

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.
- Before removing an interface, which is configured with an ACL that denies certain traffic, you must remove the ACL and commit your configuration. If this is not done, then some packets are leaked through the interface as soon as the **no interface <interface-name>** command is configured and committed.
- An access list must exist before you can use the **ipv4 access group** command.

ACL Filtering by Wildcard Mask and Implicit Wildcard Mask

Address filtering uses wildcard masking to indicate whether the software checks or ignores corresponding IP address bits when comparing the address bits in an access-list entry to a packet being submitted to the access list. By carefully setting wildcard masks, an administrator can select a single or several IP addresses for permit or deny tests.

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

- A wildcard mask bit 0 means *check* the corresponding bit value.
- A wildcard mask bit 1 means *ignore* that corresponding bit value.

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

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

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.

Including Comments 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.

Configuring IPv4 ACLs

This section describes the basic configuration of IPv4 ingress and egress ACLs.

Notes and Restrictions for Configuring IPv4 Ingress ACLs

IPv4 ingress ACLs are characterized by the following behavior.

- Ingress IPv4 ACLs are supported on all interfaces except management interfaces.
- ACL-based Forwarding (ABF) is supported only in the ingress direction.
- The total number of ACLs allowed by default per NPU is 31.
- The number of attached ACEs allowed per line card is 4000.
- Packet Length (using the **pkt-length** keyword) is supported only for ingress IPv4 ACLs. The **pkt-length** filtering values can only be specified in increments of 16 bytes by default.
- ACL logging with input interface (using the **log-input** keyword) is not supported.

Notes and Restrictions for Configuring IPv4 Egress ACLs

IPv4 egress ACLs are characterized by the following behavior.

- Egress IPv4 ACLs are supported on main physical interfaces and bundle interfaces.

  Note Egress ACLs are not directly supported on sub-interfaces. However, If you configure an egress ACL on a main interface that has sub-interfaces, the ACL action is also applied to the sub-interface traffic. This egress ACL behavior holds true even if the sub-interfaces are configured after the ACL is applied to the main interface.

- ACL is not supported on Management interface on egress direction.
- The number of attached ACEs allowed per line card is 4000.
- ACL logging (using the **log** command) and ACL logging with input interface (using the **log-input** command) is not supported.

Configuring an Ingress IPv4 ACL on a Gigabit Ethernet Interface

Use the following configuration to configure an ingress IPv4 ACL on a GigE interface.

```
/* Configure a GigE interface with an IPv4 address */
Router(config)# interface gigabitEthernet 0/0/0/0
Router(config-if)# ipv4 address 10.1.1.1 255.255.255.0
Router(config-if)# no shut
Router(config-if)# commit
Thu Jan 25 10:07:54.700 IST
Router(config-if)# exit

/* Verify if the interface is up */
Router(config)# do show ipv4 interface brief
Thu Jan 25 10:08:49.087 IST

Interface                      IP-Address      Status          Protocol Vrf-Name
GigabitEthernet0/0/0/0         10.1.1.1        Up              Up       default

/* Configure an IPv4 ingress ACL */
Router(config)# ipv4 access-list V4-ACL-INGRESS
Router(config-ipv4-acl)# 10 permit tcp 10.2.1.1 0.0.0.255 any
Router(config-ipv4-acl)# 20 deny udp any any
Router(config-ipv4-acl)# 30 permit ipv4 10.2.0.0 0.255.255.255 any
Router(config-ipv4-acl)# commit
Thu Jan 25 10:16:11.473 IST

/* Verify the ingress ACL creation */
Router(config)# do show access-lists ipv4
Thu Jan 25 10:25:19.896 IST
...
ipv4 access-list V4-ACL-INGRESS
 10 permit tcp 10.2.1.0 0.0.0.255 any
 20 deny udp any any
 30 permit ipv4 10.0.0.0 0.255.255.255 any


/* Apply the ingress ACL to the GigE interface */
Router(config)# interface GigabitEthernet0/0/0/0
Router(config-if)# ipv4 access-group V4-ACL-INGRESS ingress
Router(config-if)# commit
Thu Jan 25 10:28:19.671 IST
Router(config-if)# exit

/* Verify if the ingress ACL has been successfully applied to the interface */
Router(config)# do show ipv4 interface
Thu Jan 25 10:29:44.944 IST
GigabitEthernet0/0/0/0 is Up, ipv4 protocol is Up
  Vrf is default (vrfid 0x60000000)
  Internet address is 10.1.1.1/24
  MTU is 1514 (1500 is available to IP)
  Helper address is not set
  Directed broadcast forwarding is disabled
  Outgoing access list is not set
  Inbound  common access list is not set, access list is V4-ACL-INGRESS
  Proxy ARP is disabled
  ICMP redirects are never sent
  ICMP unreachables are always sent
  ICMP mask replies are never sent
  Table Id is 0xe0000000
```

You have successfully configured an IPv4 ingress ACL on a Gigabit Ethernet interface.

Configuring an Egress IPv4 ACL on a Gigabit Ethernet Interface

Use the following configuration to configure an egress IPv4 ACL on a GigE interface.

```
/* Configure a GigE interface with an IPv4 address */
Router(config)# interface gigabitEthernet 0/0/0/0
Router(config-if)# ipv4 address 20.1.1.1 255.255.255.0
Router(config-if)# no shut
Router(config-if)# commit
Thu Jan 25 10:08:38.767 IST
Router(config-if)# exit

/* Verify if the interface is up */
Router(config)# do show ipv4 interface brief
Thu Jan 25 10:08:49.087 IST

Interface                      IP-Address      Status          Protocol Vrf-Name
GigabitEthernet0/0/0/0         10.1.1.1        Up              Up       default
GigabitEthernet0/0/0/0         20.1.1.1        Up              Up       default

/* Configure an IPv4 egress ACL */
Router(config)# ipv4 access-list V4-ACL-EGRESS
Router(config-ipv4-acl)# 10 permit ipv4 10.2.0.0 0.255.255.255 20.2.0.0 0.255.255.255
Router(config-ipv4-acl)# 20 deny ipv4 any any
Router(config-ipv4-acl)# commit
Thu Jan 25 10:25:04.655 IST

/* Verify the egress ACL creation */
Router(config)# do show access-lists ipv4
Thu Jan 25 10:25:19.896 IST
ipv4 access-list V4-ACL-EGRESS
 10 permit ipv4 10.0.0.0 0.255.255.255 20.0.0.0 0.255.255.255
 20 deny ipv4 any any
...

/* Apply the egress ACL to the GigE interface */
Router(config)# interface gigabitEthernet 0/0/0/1
Router(config-if)# ipv4 access-group V4-ACL-EGRESS egress
Router(config-if)# commit
Thu Jan 25 10:28:45.937 IST
Router(config-if)# exit

/* Verify if the egress ACL has been successfully applied to the interface */
Router(config)# do show ipv4 interface
Thu Jan 25 10:29:44.944 IST
GigabitEthernet 0/0/0/1 is Up, ipv4 protocol is Up
  Vrf is default (vrfid 0x60000000)
  Internet address is 20.1.1.1/24
  MTU is 1514 (1500 is available to IP)
  Helper address is not set
  Directed broadcast forwarding is disabled
  Outgoing access list is V4-ACL-EGRESS
  Inbound  common access list is not set, access list is not set
  Proxy ARP is disabled
  ICMP redirects are never sent
  ICMP unreachables are always sent
  ICMP mask replies are never sent
  Table Id is 0xe0000000
...
```

You have successfully configured an IPv4 egress ACL on a Gigabit Ethernet interface.

Modifying ACLs

This section describes a sample configuration for modification of ACLs.

```
*/ Create an Access List*/
Router(config)#ipv4 access-list acl_1

*/Add entries (ACEs) to the ACL*/
Router(config-ipv4-acl)#10 permit ip host 10.3.3.3 host 172.16.5.34
Router(config-ipv4-acl)#20 permit icmp any any
Router(config-ipv4-acl)#30 permit tcp any host 10.3.3.3
Router(config-ipv4-acl)#end

*/Verify the entries of the ACL*/:
Router#show access-lists ipv4 acl_1
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

*/Add new entries, one with a sequence number "15" and another without a sequence number
to the ACL. Delete an entry with the sequence number "30":*/
Router(config)#ipv4 access-list acl_1
Router(config-ipv4-acl)# 15 permit 10.5.5.5 0.0.0.255
Router(config-ipv4-acl)# no 30
Router(config-ipv4-acl)# permit 10.4.4.4 0.0.0.255
Router(config-ipv4-acl)# commit


*/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*/

*/Verify the entries of the ACL:*/
Router(config)#show access-lists ipv4 acl_1
ipv4 access-list acl_1
 10 permit ipv4 host 10.3.3.3 host 172.16.5.34

15 permit 10.5.5.5 0.0.0.255---*/newly added ACE (with the sequence number)*/
20 permit icmp any any
30 permit ipv4 10.4.4.0 0.0.0.255 any ---*/newly added ACE (without the sequence number)*/

*/The entry with the sequence number 30, that is, "30 permit tcp any host 10.3.3.3" is
deleted from the ACL*/
```

You have successfully modified ACLs in operation.

Configuring ACL-based Forwarding

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.

Feature Highlights

- ABF is only supported on ingress ACL.
- ABF supports nexthop modifications. You can modify a nexthop, remove a nexthop, or make changes between existing nexthops.
- VRF-aware ABF is supported for IPv4 and IPv6 with up to three next hops.
- As ABF is ACL-based, packets that do not match an existing rule (ACE) in the ACL are subject to the default ACL rule (drop all). If the ACL is being used for ABF-redirect only (not for security), then include an explicit ACE rule at the end of the ACL (lowest user priority) to match and "permit" all traffic. This ensures that all traffic that does not match an ABF rule is permitted and forwarded as normal.
- ABF is supported on permit rules only.
- VRF-select (where only the VRF is configured for the nexthop) is not supported.
- ABF with object-tracking is not supported.
- ABF default route is not supported.
- Packets punted in the ingress direction from the NPU to the linecard CPU are not subjected to ABF treatment due to lack of ABF support in the slow path. These packets will be forwarded normally based on destination-address lookup by the software dataplane. Some examples of these types of packets are (but are not limited to) packets with IPv4 options, IPv6 extension headers, and packets destined for glean (unresolved/incomplete) adjacencies.
- Packets destined to the local IP interface ("for-us" packets) are subjected to redirect if they match the rule containing the ABF action. This can be avoided by either designing the rule to be specific enough to avoid matching the “for-us” packets or placing an explicit permit ACE rule (with higher priority) into the ACL before the matching ABF rule.

Configuration Example

To configure ACL-based forwarding, perform this task:

```
/* Enter IPv4 access list configuration mode and configure an ACL: */
router# configure
router(config)# ipv4 access-list abf-acl

/* Set the conditions for the ACL and configure ABF: */
/* The next hop for this entry is specified. */
router(config-ipv4-acl)# 10 permit ipv4 192.168.18.0 0.255.255.255 any nexthop1 ipv4
192.168.20.2
router(config-ipv4-acl)# 15 permit ipv4 192.168.21.0 0.0.0.255 any
router(config-ipv4-acl)# 20 permit ipv4 192.168.22.0 0.0.255.255 any nexthop1 ipv4
192.168.23.2
/* More than two nexthops */
router(config-ipv4-acl)# 25 permit tcp any range 2000 3000 any range 4000 5000 nexthop1
ipv4 192.168.23.1 nexthop2 ipv4 192.168.24.1 nexthop3 ipv4 192.168.25.1

/* VRF support on ABF */
router(config-ipv4-acl)# 30 permit tcp any eq www host 192.168.12.2 precedence immediate
nexthop1 vrf vrf1_ipv4 ipv4 192.168.13.2 nexthop2 vrf vrf1_ipv4 ipv4 192.168.14.2
```

```
router(config-ipv4-acl)# 35 permit ipv4 any any

router(config-ipv4-acl)# commit

/* (Optional) Display ACL information: */
router# show access-lists ipv4 abf-acl
```

Running Configuration

```
ipv4 access-list abf-acl
10 permit ipv4 192.168.18.0 0.255.255.255 any nexthop1 192.168.20.2
15 permit ipv4 192.168.21.0 0.0.0.255 any
20 permit ipv4 192.168.22.0 0.0.255.255 any nexthop1 192.168.23.2
25 permit tcp any range 2000 3000 any range 4000 5000 nexthop1 ipv4 192.168.23.1 nexthop2
ipv4 192.168.24.1 nexthop3 ipv4 192.168.25.1
30 permit tcp any eq www host 192.168.12.2 precedence immediate nexthop1 vrf vrf1_ipv4 ipv4
 192.168.13.2 nexthop2 vrf vrf1_ipv4 ipv4 192.168.14.2
35 permit ipv4 any any
commit
!
```

Verification

Use the following command to verify the IP nexthop state in ABF to ensure that the expected nexthop is up:

```
router# show access-lists ipv4 abf nexthops client pfilter_ea location 0/3/CPU0
Tue May 17 22:25:05.940 UTC

 ACL name : abf-acl
  ACE seq.         NH-1                  NH-2                  NH-3
 ---------  ---------------------  ---------------------  ---------------------
        20    Global 192.168.23.2    Not present        Not present
    status                     UP    Not present            Not present
     exist                     No           Not present            Not present
    pd ctx                Present          Not present            Not present
              Track not present      Track not present                       --
        25    Global 192.168.23.1    Global 192.168.24.1        Global 192.168.25.1
    status                     UP                     UP             UP
     exist                    Yes                    Yes              Yes
    pd ctx                Present                Present            Present
              Track not present      Track not present            Track not present
```

Use the following command to verify if ABF is currently attached to any interfaces at any linecard:

```
show access-lists usage pfilter location all
```

ACLs on Bridge Virtual Interfaces

Bridge Virtual Interfaces (BVIs) provide a bridge between the routing and bridging domains on a router. A BVI is configured with an IP address and operates as a regular routed interface. You can configure an ACL on a BVI to filter the traffic for the network that uses the interface.

Note Do not delete an ACL attached to a BVI interface when the BVI interface is not part of a bridge domain. Later, if you add the BVI interface to the bridge domain then the traffic is dropped.

Increased TCAM Consumption with Configuring ACLs on BVIs

The consumption of TCAM resources is impacted in the following manner when ACLs are configured on BVIs.

- When an ACL is attached to a BVI interface, TCAM entries are programmed on all line cards regardless of physical interface membership. This leads to greater consumption of TCAM resources even on line cards that do not have BVI member interfaces.
- When an ACL is attached to a BVI interface, TCAM entries are programmed on all NPUs in a line card, regardless of physical interface membership. This leads to greater consumption of TCAM resources even on NPUs that do not have BVI member interfaces.
- For ingress ACLs, the TCAM entries for the same ACL are shared across interfaces on the same NPU.
- For egress ACLs, the TCAM entries for the same ACL are unique for all interfaces. This leads to greater consumption of TCAM resources.

Restrictions for Configuring ACLs on BVIs

You must be aware of the following restrictions before proceeding to configure ACLs on BVIs.

- Egress IPv6 ACLs are not supported on BVIs.
- When an egress IPv4 ACL is enabled on a BVI through the **hw-module** command, no other interface types are supported for the ACL (non-BVI interfaces are not supported for the ACL in this mode).
- A reboot of the line card is required after using the **hw-module profile acl egress layer3 interface-based** command to enable IPv4 egress ACL on BVI and to enable permit statistics for the ACL.
- QoS stats are not supported (disabled) when "acl-permit" stats are enabled.

Prerequisites for Configuring IPv4 Egress ACLs on BVIs

By default, an IPv4 egress ACL on a BVI is disabled, and ACL filtering does not take place even when the ACL is attached to the BVI. Hence, we use the **hw-module** command, which enables the ACL when the line cards are reloaded.

Note IPv4 and IPv6 ingress ACLs do not require this configuration.

Use the following configuration to enable an IPv4 egress ACL on a BVI on the hardware and reload the line cards.

```
/* Enable an IPv4 egress ACL on BVI */
RP/0/RP0/CPU0:router(config)# hw-module profile acl egress layer3 interface-based
/* Enable permit statistics for the egress ACL (by default, only deny statistics are shown)*/
RP/0/RP0/CPU0:router(config)# hw-module profile stats acl-permit
RP/0/RP0/CPU0:router(config)# commit
```

```
RP/0/RP0/CPU0:router(config)# end
RP/0/RP0/CPU0:router# reload location all
Wed Apr 5 23:05:46.193 UTC
Proceed with reload? [confirm]
```

Configuration

The following section describes the procedure for configuring IPv4 ingress and egress ACLs on BVIs.

To configure IPv4 ingress and egress ACLs on a BVI, use the following procedure with sample configuration.

1. Enter the Global Configuration mode, and configure an IPv4 ingress ACL.

   ```
   RP/0/RP0/CPU0:router(config)# ipv4 access-list v4-acl-ingress
   RP/0/RP0/CPU0:router(config-ipv4-acl)# 10 permit tcp any 10.1.1.0/24 dscp cs6
   RP/0/RP0/CPU0:router(config-ipv4-acl)# 20 deny udp any any eq ssh
   RP/0/RP0/CPU0:router(config-ipv4-acl)# 30 permit ipv4 any any
   RP/0/RP0/CPU0:router(config-ipv4-acl)# commit
   RP/0/RP0/CPU0:router(config-ipv4-acl)# exit
   ```

2. Configure an IPv4 egress ACL.

   ```
   RP/0/RP0/CPU0:router(config)# ipv4 access-list v4-acl-egress
   RP/0/RP0/CPU0:router(config-ipv4-acl)# 10 deny ipv4 any any fragments log
   RP/0/RP0/CPU0:router(config-ipv4-acl)# 20 deny tcp any any ack
   RP/0/RP0/CPU0:router(config-ipv4-acl)# 30 permit ipv4 any any
   RP/0/RP0/CPU0:router(config-ipv4-acl)# commit
   RP/0/RP0/CPU0:router(config-ipv4-acl)# exit
   ```

3. Configure the Gigabit Ethernet interface that must be mapped to the BVI, and enable it for Layer 2 transport.

   ```
   RP/0/RP0/CPU0:router(config)# interface GigabitEthernet 0/0/0/0
   RP/0/RP0/CPU0:router(config-if)# l2transport
   RP/0/RP0/CPU0:router(config-if-l2)# commit
   ```

4. Attach the ingress and egress ACLs to the BVI.

   ```
   RP/0/RP0/CPU0:router(config)# interface BVI1
   RP/0/RP0/CPU0:router(config-if)# ipv4 access-group v4-acl-ingress ingress
   RP/0/RP0/CPU0:router(config-if)# ipv4 access-group v4-acl-egress egress
   RP/0/RP0/CPU0:router(config-if)# commit
   RP/0/RP0/CPU0:router(config-if)# exit
   ```

5. Configure the bridge domain with the Gigabit Ethernet interface and BVI.

   ```
   RP/0/RP0/CPU0:router(config)# l2vpn
   RP/0/RP0/CPU0:router(config-l2vpn)# bridge group BG1
   RP/0/RP0/CPU0:router(config-l2vpn-bg)# bridge-domain B1
   RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# interface GigabitEthernet 0/0/0/0
   RP/0/RP0/CPU0:router(config-l2vpn-bg-bd-ac)# routed interface BVI1
   RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# commit
   RP/0/RP0/CPU0:router(config-l2vpn-bg-bd)# exit
   RP/0/RP0/CPU0:router(config-l2vpn-bg)# exit
   RP/0/RP0/CPU0:router(config-l2vpn)# exit
   ```

6. Confirm that your configuration has been successfully committed.

```
RP/0/RP0/CPU0:router(config)# show run
...
!
ipv4 access-list v4-acl-egress
 10 deny ipv4 any any fragments log
 20 deny tcp any any ack
 30 permit ipv4 any any
!
ipv4 access-list v4-acl-ingress
 10 permit tcp any 10.1.1.0/24 dscp cs6
 20 deny udp any any eq ssh
 30 permit ipv4 any any
!
interface GigabitEthernet0/0/0/0
 l2transport
 !
!
interface BVI1
ipv4 address 209.165.200.224/27
 ipv4 access-group v4-acl-ingress ingress
 ipv4 access-group v4-acl-egress egress

!
l2vpn
 bridge group BG1
  bridge-domain B1
   interface GigabitEthernet0/0/0/0
   !
   routed interface BVI1
  !
 !
!
end
```

7. Exit to the Executive Privileged mode and confirm that the ACLs are in operation.

```
RP/0/RP0/CPU0:router# show access-lists interface bvi1
Tue May 9 10:01:25.732 EDT
Input ACL (common): GigabitEthernet 0/0/0/0 (interface): v4-acl-ingress
Output ACL: v4-acl-egress

RP/0/RP0/CPU0:router# show access-lists summary
Tue May 9 10:02:01.167 EDT
ACL Summary:
Total ACLs configured: 2
Total ACEs configured: 6


RP/0/RP0/CPU0:router# show access-lists ipv4 v4-acl-egress hardware egress location
0/0/CPU0
ipv4 access-list v4-acl-egress
10 deny ipv4 any any fragments log (15214 matches)
20 deny tcp any any ack (15214 matches)
30 permit ipv4 any any (15214 matches)
```

The output clearly shows the configured ACLs, the total number of ACEs (three per ACL), and also the ACE matches in hardware.

You have successfully configured and enabled IPv4 ingress and egress ACL on a BVI.

Configuring ACLs with Fragment Control

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** and **permit** . 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** 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.

Configuring IPv6 Extension Header

When the following IPv6 extensions headers are present in the ACL, by default, the control plane CPU filters the packets and applies security ACLs:

- Hop-by-Hop
- Destination-Options
- Routing
- Fragment
- Mobility
- Host-Identity
- SHIM6

Filtering of these packets in CPU reduces the packet rate to 100 packets/sec and later leads to packet drop. Any extension headers that are not identified are not detected as extension header and ACLs may not work properly.

You can use the following command to verify the number of packets dropped using the following command:

```
Router # show controllers npu stats traps-all instance all location 0/2/cpu0 |  i EXT_HDR
```

You can add ACL permit and deny rules with right priority to process the packets at full rate. Adding the rules avoids the packets being sent to the CPU. But bypasses the security ACLs at layer 4.

Note You can use Layer 3 information to filter these packets.

For example, to permit AH packets, you can add the following ACL rule:

```
10 permit ahp any
Or
10 deny ahp any
```

To disable the default behaviour of filtering the packets in the CPU and yet permit the extension headers, configure the Ipv6 extension header.

With this, you need not include permit rules in each ACL. All the packets with extension headers bypass the security ACLs and permit the extension header.

You can enable or disable the Ipv6 extension header option anytime without restarting the device.

Configuration Example

```
Router# configure
Router(config)# hw-module profile acl IPv6 ext-header permit

Router(config)# commit
```

Configuring Extended Access Lists

Configuration Example

Creates an IPv4 named access list "acl_1". This access list permits ICMP protocol packets with any source and destination IPv4 address and denies TCP protocol packets with any source and destination IPv4 address and port greater than 5000.

```
Router#configure
Router(config)#ipv4 access-list acl_1


Router(config-ipv4-acl)#20 permit icmp any any
Router(config-ipv4-acl)#30 deny tcp any any gt 5000
Router(config-ipv4-acl)#commit
```

Running Configuration

```
Router#  show running-config ipv4 access-list acl_1
ipv4 access-list acl_1
 20 permit icmp any any
 30 deny tcp any any gt 5000
!
```

Verification

Verify that the permit and deny settings are according to the set configuration.

```
Router# show access-lists acl_1
ipv4 access-list acl_1
```

```
 20 permit icmp any any
 30 deny tcp any any gt 5000
Router#
```

Associated Commands

- ipv4 access-list
- ipv6 access-list
- permit (IPv4)
- permit (IPv6)
- remark (IPv4)
- remark (IPv6)
- deny (IPv4)
- deny (IPv6)

What to Do Next

After creating an access list, you must apply it to a line or an interface. 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 ACL configuration mode and re-enter it before adding the first ACE.

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.

Understanding Prefix Lists

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).

BGP Filtering Using Prefix Lists

Prefix lists can be used as an alternative to access lists in many BGP route filtering commands. It is configured under the Global configurations of the BGP protocol. 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 IPv4 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.

Configuring Prefix Lists

Configuration Example

Creates a prefix-list "pfx_2" with a remark "Deny all routes with a prefix of 10/8". This prefix-list denies all prefixes matching /24 in 128.0.0.0/8.

```
Router#configure
Router(config)#ipv4 prefix-list pfx_2


Router(config-ipv4_pfx)#10 remark Deny all routes with a prefix of 10/8
Router(config-ipv4_pfx)#20 deny 128.0.0.0/8 eq 24
/* Repeat the above step as necessary. Use the no sequence-number command to delete an
entry. */

Router(config-ipv4_pfx)#commit
```

Running Configuration

```
Router#show running-config ipv4 prefix-list pfx_2
ipv4 prefix-list pfx_2
 10 remark Deny all routes with a prefix of 10/8
 20 deny 128.0.0.0/8 eq 24
!
```

Verification

Verify that the permit and remark settings are according to the set configuration.

```
Router# show prefix-list pfx_2
ipv4 prefix-list pfx_2
 10 remark Deny all routes with a prefix of 10/8
 20 deny 128.0.0.0/8 eq 24
RP/0/RP0/CPU0:ios#
```

Associated Commands

- ipv4 prefix-list
- ipv6 prefix-list
- show prefix-list ipv4
- show prefix-list ipv6

Sequencing Prefix List Entries and Revising the Prefix List

Configuration Example

Assigns 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.

```
Router#config
Router(config)#ipv4 prefix-list cl_1
Router(config)#10 permit 172.16.0.0 0.0.255.255
/* Repeat the above step as necessary adding statements by sequence number where you planned;
 use the no sequence-number command to delete an entry */

Router(config)#commit
end
Router#resequence prefix-list ipv4 cl_1 20 15
```

Running Configuration

```
/*Before resequencing/*
Router#show running-config ipv4 prefix-list cl_1
ipv4 prefix-list cl_1
 10 permit 172.16.0.0/16
!
/* After resequencing using the resequence prefix-list ipv4 cl_1 20 15 command: */
Router#show running-config ipv4 prefix-list cl_1
ipv4 prefix-list cl_1
 35 permit 172.16.0.0/16
!
```

Verification

Verify that the prefix list has been resequenced:

```
Router#show prefix-list cl_1
ipv4 prefix-list cl_1
 35 permit 172.16.0.0/16
```

Associated Commands

- resequence prefix-list ipv4
- resequence prefix-list ipv6
- ipv4 prefix-list
- ipv6 prefix-list
- show prefix-lists ipv4
- show prefix-lists ipv6