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.

Traditional ACLs do not support commpression. Object-group ACLs use compression to accommodate the large number of ACEs.

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.

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-based Forwarding (ABF) is not supported in common ACLs.
- Filtering of MPLS packets with the explicit-null or de-aggregation label is supported on the ingress direction.
- **ACL policy on an MPLS interface** - An IP access control list (IP ACL) is in effect when applied on an MPLS interface on the egress PE (NCS 5000 Series) router. When an MPLS tunnel terminates on this router, the ACL policy is applied on the packets. So, ensure that correct entries are in the ACL **permit** and **deny** commands, so that the traffic is forwarded. If not, the traffic might be dropped.

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.

Implementing Access Lists

Implementing ACLs involve:

1. Creating Standard or Extended ACLs-

   Create an ACL that 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.

2. Applying the ACL to specific interfaces-

   After configuring an access list, you must reference the access list to make it work. Access lists can be applied on inbound interfaces. 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.

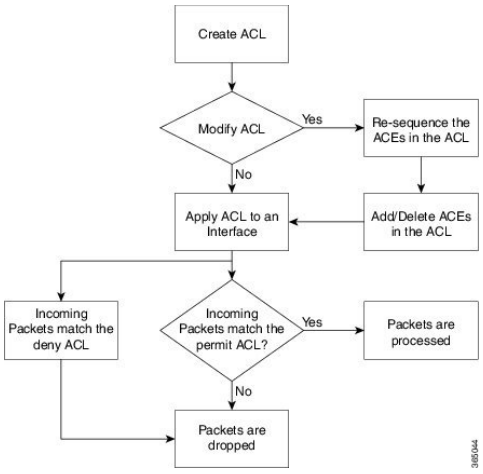
   When you apply an access list that has not yet been defined to an interface, the software acts as if the access list has not been applied to the interface and accepts all packets. Note this behavior if you use undefined access lists as a means of security in your network.

The other actions that can be performed on an ACL are:

- Copying Access Lists—

  Users can create a copy of an existing access list using the **copy access-list ipv4 | ipv6** command.

- Adding or Deleting Access-List Entries



Note If there is no match to either permit or deny the packets, then the default action is to drop the packet.

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 Commands:

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

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.

Configuring Standard Access Lists

Configuration Example

Creates an IPv4 named access list "acl_1" with a remark "Do not allow user1 to telnet out". This access list permits packets from the source address of "172.16.0.0" with a wild-card mask of "0.0.255.255"

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


Router(config-ipv4-acl)# 10 remark Do not allow user1 to telnet out
Router(config-ipv4-acl)# 20 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-ipv4-acl)# commit
```

Running Configuration

```
Router# show running-config ipv4 access-list acl_1
ipv4 access-list acl_1
 10 remark Do not allow user1 to telnet out
 20 permit ipv4 172.16.0.0 0.0.255.255 any
!
```

Verification

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

```
Router# show access-lists ipv4 acl_1
ipv4 access-list acl_1
 10 remark Do not allow user1 to telnet out
 20 permit ipv4 172.16.0.0 0.0.255.255 any
```

Associated Commands

IPv4 Commands:

- ipv4 access-list
- remark(IPv4)
- permit (IPv4)
- deny (IPv4)
- show access-lists ipv4

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 config-ipv4-acl or config-ipv6-acl mode to configuration mode and re-enter the same mode before adding the first ACE.

Applying Access Lists

Configuration Example

Applies the access lists on the interface, which acts as filters on packets inbound from TenGigE interface 0/0/0/2.

```
Router# configure
Router(config)# interface TenGigE 0/0/0/2
Router(config-if)# ipv4 access-group acl_1 ingress


Router(config-ipv4-acl)# commit
```

Verification

Verify that the ACL applied (acl_1) on the interface is listed:

```
Router #show access-lists interface TenGigE 0/0/0/2
Input ACL
(common): N/A      (interface): acl_1
Output ACL: N/A
Router#
```

Associated Commands

IPv4 Commands:

- ipv4 access-group
- show access-lists ipv4

Sequencing Access List Entries and Revising the Access List

Configuration Example

Assigns sequence numbers to entries in an access list and explains how to add or delete an entry to or from an access list. In this configuration, it is assumed that a user wants to revise an access list.

Resequences entries in the access list “acl_1”. The starting value in the resequenced access list is 20, and increment value is 15.

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.

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

Router(config-ipv4-acl)#10 permit 10.1.1.1
*/Repeat the above step as necessary adding statements by sequence number where you planned*/
Router(config-ipv4-acl)#20 permit 10.2.0.0 0.0.255.25
Router(config-ipv4-acl)#no 25
*/Use the no sequence-number command to delete an entry with that specific sequence number*/
Router(config-ipv4-acl)#30 permit tcp host 10.2.2.2 255.255.0.0 any eq telnet
```

```
Router(config-ipv4-acl)#commit
end
Router#resequence access-list ipv4 acl_1 20 15
```

Running Configuration

```
Before resequencing:
Router#show running-config ipv4 access-list acl_1
ipv4 access-list acl_1
 10 remark Do not allow user1 to telnet out
 20 permit ipv4 172.16.0.0 0.0.255.255 any
!
*/After resequencing using the resequence access-list ipv4 acl_1 20 15 command*/:

Router# show running-config ipv4 access-list acl_1
ipv4 access-list acl_1
 20 permit ipv4 host 10.1.1.1 any
 35 permit ipv4 10.2.0.0 0.0.255.25 any
 50 permit ipv4 10.1.1.1/24 any
 65 permit tcp host 10.2.2.2 any eq telnet
!
```

Verification

Verify that the access list (acl_1) contains all permit and deny options (after resequencing) that were configured:

```
Router#show access-lists ipv4 acl_1
ipv4 access-list acl_1
20 permit ipv4 host 10.1.1.1 any
35 permit ipv4 10.2.0.0 0.0.255.25 any
50 permit ipv4 10.1.1.1/24 any
65 permit tcp host 10.2.2.2 any eq telnet
```

Associated Commands

IPv4 Commands:

- resequence access-list ipv4
- ipv4 access-list
- permit (IPv4)
- show access-lists ipv4

Use case

This use case explains how to create, resequence, add new entries, delete an entry and verify the ACL:

```
*/ 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

/* Resequence the ACL */
Router(config)#resequence ipv4 access-list acl_1 10 20
/* 10 indicates the starting value in the resequenced ACL and the subsequent entries in the
 original ACL are incremented by 20 */

/* Verify the entries of the ACL */
Router#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

/* 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

/* 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
10 permit ip 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) */
50 permit tcp any host 10.3.3.3
60 permit 10 .4.4.4 0.0.0.255---/* newly added ACE (without the sequence number) */
/* The entry with the sequence number 30, that is, "30 permit icmp any any" is deleted from
 the ACL */
```

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 Commands:

- ipv4 prefix-list
- show prefix-list ipv4

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.

Note It is possible to resequence ACLs for prefix-list but not for security ACLs.

```
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
 20 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
 20 permit 172.16.0.0/16
```

Associated Commands

IPv4 Commands:

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

Understanding Access Lists and Prefix Lists

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 packets that match the ACE rule, use the following configuration.

```
RP/0/RP0/CPU0:router# hardware access-list atomic-disable default-action permit
```

To disable atomic updates on the hardware, by denying all packets until the modification is complete, use the following configuration.

```
RP/0/RP0/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/RP0/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/RP0/CPU0:router(config)# ipv4 access-list list1
RP/0/RP0/CPU0:router(config-ipv4-acl)# 30 permit ipv4 30.1.1.0/24 any
RP/0/RP0/CPU0:router(config-ipv4-acl)# commit
```

3. Verify if your ACE was added successfully.

```
RP/0/RP0/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/RP0/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/RP0/CPU0:router(config)# ipv4 access-list list1
RP/0/RP0/CPU0:router(config-ipv4-acl)# no 30
RP/0/RP0/CPU0:router(config-ipv4-acl)# commit
```

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

```
RP/0/RP0/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/RP0/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/RP0/CPU0:router(config)# ipv4 access-list list1
RP/0/RP0/CPU0:router(config-ipv4-acl)# 10 permit ipv4 11.1.1.0/24 any
RP/0/RP0/CPU0:router(config-ipv4-acl)# commit
```

3. Verify if the ACE replacement is successful.

```
RP/0/RP0/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/RP0/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/RP0/CPU0:router(config)# ipv4 access-list list1
RP/0/RP0/CPU0:router(config-ipv4-acl)# no 20
RP/0/RP0/CPU0:router(config-ipv4-acl)# permit ipv4 12.1.1.0/24 any
RP/0/RP0/CPU0:router(config-ipv4-acl)# commit
```

3. Verify if the modification is successful.

```
RP/0/RP0/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.