Access Control Lists

Access control lists (ACLs) are used by many different features. When applied to interfaces or globally as access rules, they permit or deny traffic that flows through the appliance. For other features, the ACL selects the traffic to which the feature will apply, performing a matching service rather than a control service.

The following sections explain the basics of ACLs and how to configure and monitor them. Access rules, ACLs applied globally or to interfaces, are explained in more detail in Access Rules.

About ACLs

Access control lists (ACLs) identify traffic flows by one or more characteristics, including source and destination IP address, IP protocol, ports, EtherType, and other parameters, depending on the type of ACL. ACLs are used in a variety of features. ACLs are made up of one or more access control entries (ACEs).

ACL Types

The ASA uses the following types of ACLs:

- Extended ACLs—Extended ACLs are the main type that you will use. These ACLs are used for access rules to permit and deny traffic through the device, and for traffic matching by many features, including service policies, AAA rules, WCCP, Botnet Traffic Filter, and VPN group and DAP policies. See Configure Extended ACLs, on page 8.
- EtherType ACLs—EtherType ACLs apply to non-IP layer-2 traffic on bridge group member interfaces only. You can use these rules to permit or drop traffic based on the EtherType value in the layer-2 packet. With EtherType ACLs, you can control the flow of non-IP traffic across the device. See Configure EtherType ACLs, on page 18.
- Webtype ACLs—Webtype ACLs are used for filtering clientless SSL VPN traffic. These ACLs can deny access based on URLs or destination addresses. See Configure Webtype ACLs, on page 15.

- Standard ACLs—Standard ACLs identify traffic by destination address only. There are few features that use them: route maps and VPN filters. Because VPN filters also allow extended access lists, limit standard ACL use to route maps. See Configure Standard ACLs, on page 14.

The following table lists some common uses for ACLs and the type to use.

Table 1: ACL Types and Common Uses

ACL Use	ACL Type	Description
Control network access for IP traffic (routed and transparent mode)	Extended	The ASA does not allow any traffic from a lower security interface to a higher security interface unless it is explicitly permitted by an extended ACL. In routed mode, you must use an ACL to permit traffic between a bridge group member interface and an interface outside same the bridge group. **Note** To access the ASA interface for management access, you do not also need an ACL allowing the host IP address. You only need to configure management access according to the general operations configuration guide.
Identify traffic for AAA rules	Extended	AAA rules use ACLs to identify traffic.
Augment network access control for IP traffic for a given user	Extended, downloaded from a AAA server per user	You can configure the RADIUS server to download a dynamic ACL to be applied to the user, or the server can send the name of an ACL that you already configured on the ASA.
VPN access and filtering	Extended Standard	Group policies for remote access and site to site VPNs use standard or extended ACLs for filtering. Remote access VPNs also use extended ACLs for client firewall configurations and dynamic access policies.
Identify traffic in a traffic class map for Modular Policy Framework	Extended	ACLs can be used to identify traffic in a class map, which is used for features that support Modular Policy Framework. Features that support Modular Policy Framework include TCP and general connection settings, and inspection.
For bridge group member interfaces, control network access for non-IP traffic	EtherType	You can configure an ACL that controls traffic based on its EtherType for any interface that is a member of a bridge group.
Identify route filtering and redistribution	Standard Extended	Various routing protocols use standard ACLs for route filtering and redistribution (through route maps) for IPv4 addresses, and extended ACLs for IPv6.
Filtering for clientless SSL VPN	Webtype	You can configure a webtype ACL to filter URLs and destinations.

ACL Names

Each ACL has a name or numeric ID, such as outside_in, OUTSIDE_IN, or 101. Limit the names to 241 characters or fewer.Consider using all uppercase letters to make it easier to find the name when viewing a running configuration.

Develop a naming convention that will help you identify the intended purpose of the ACL. For example, ASDM uses the convention *interface-name_purpose_direction*, such as “outside_access_in”, for an ACL applied to the “outside” interface in the inbound direction.

Traditionally, ACL IDs were numbers. Standard ACLs were in the range 1-99 or 1300-1999. Extended ACLs were in the range 100-199 or 2000-2699. The ASA does not enforce these ranges, but if you want to use numbers, you might want to stick to these conventions to maintain consistency with routers running IOS Software.

Access Control Entry Order

An ACL is made up of one or more ACEs. Unless you explicitly insert an ACE at a given line, each ACE that you enter for a given ACL name is appended to the end of the ACL.

The order of ACEs is important. When the ASA decides whether to forward or drop a packet, the ASA tests the packet against each ACE in the order in which the entries are listed. After a match is found, no more ACEs are checked.

Thus, if you place a more specific rule after a more general rule, the more specific rule might never be hit. For example, if you want to permit network 10.1.1.0/24, but drop traffic from host 10.1.1.15 on that subnet, the ACE that denies 10.1.1.15 must come before the one that permits 10.1.1.0/24. If the permit 10.1.1.0/24 ACE comes first, 10.1.1.15 will be allowed, and the deny ACE will never be matched.

In an extended ACL, use the **line** *number* parameter on the **access-list** command to insert rules at the right location. Use the **show access-list** *name* command to view the ACL entries and their line numbers to help determine the right number to use. For other types of ACL, you must rebuild the ACL (or better, use ASDM) to change the order of ACEs.

Permit/Deny vs. Match/Do Not Match

Access control entries either “permit” or “deny” traffic that matches the rule. When you apply an ACL to a feature that determines whether traffic is allowed through the ASA or is dropped, such as global and interface access rules, “permit” and “deny” mean what they say.

For other features, such as service policy rules, “permit” and “deny” actually mean “match” or “do not match.” In these cases, the ACL is selecting the traffic that should receive the services of that feature, such as application inspection or redirection to a service module. “Denied” traffic is simply traffic that does not match the ACL, and thus will not receive the service.

Access Control Implicit Deny

ACLs that are used for through-the-box access rules have an implicit deny statement at the end. Thus, for traffic controlling ACLs such as those applied to interfaces, if you do not explicitly permit a type of traffic, that traffic is dropped. For example, if you want to allow all users to access a network through the ASA except for one or more particular addresses, then you need to deny those particular addresses and then permit all others.

For management (control plane) ACLs, which control to-the-box traffic, there is no implicit deny at the end of a set of management rules for an interface. Instead, any connection that does not match a management access rule is then evaluated by regular access control rules.

For ACLs used to select traffic for a service, you must explicitly "permit" the traffic; any traffic not "permitted" will not be matched for the service; "denied" traffic bypasses the service.

For EtherType ACLs, the implicit deny at the end of the ACL does not affect IP traffic or ARPs; for example, if you allow EtherType 8037, the implicit deny at the end of the ACL does not now block any IP traffic that you previously allowed with an extended ACL (or implicitly allowed from a high security interface to a low security interface). However, if you *explicitly* deny all traffic with an EtherType ACE, then IP and ARP traffic is denied; only physical protocol traffic, such as auto-negotiation, is still allowed.

IP Addresses Used for Extended ACLs When You Use NAT

When you use NAT or PAT, you are translating addresses or ports, typically mapping between internal and external addresses. If you need to create an extended ACL that applies to addresses or ports that have been translated, you need to determine whether to use the real (untranslated) addresses or ports or the mapped ones. The requirement differs by feature.

Using the real address and port means that if the NAT configuration changes, you do not need to change the ACLs.

Features That Use Real IP Addresses

The following commands and features use real IP addresses in the ACLs, even if the address as seen on an interface is the mapped address:

- Access Rules (extended ACLs referenced by the **access-group** command)
- Service Policy Rules (Modular Policy Framework **match access-list** command)
- Botnet Traffic Filter traffic classification (**dynamic-filter enable classify-list** command)
- AAA Rules (**aaa ... match** commands)
- WCCP (**wccp redirect-list group-list** command)

For example, if you configure NAT for an inside server, 10.1.1.5, so that it has a publicly routable IP address on the outside, 209.165.201.5, then the access rule to allow the outside traffic to access the inside server needs to reference the server's real IP address (10.1.1.5), and not the mapped address (209.165.201.5).

```
hostname(config)# object network server1
hostname(config-network-object)# host 10.1.1.5
hostname(config-network-object)# nat (inside,outside) static 209.165.201.5

hostname(config)# access-list OUTSIDE extended permit tcp any host 10.1.1.5 eq www
hostname(config)# access-group OUTSIDE in interface outside
```

Features That Use Mapped IP Addresses

The following features use ACLs, but these ACLs use the mapped values as seen on an interface:

- IPsec ACLs
- **capture** command ACLs

- Per-user ACLs
- Routing protocol ACLs
- All other feature ACLs.

Time-Based ACEs

You can apply time range objects to extended and webtype ACEs so that the rules are active for specific time periods only. These types of rules let you differentiate between activity that is acceptable at certain times of the day but that is unacceptable at other times. For example, you could provide additional restrictions during working hours, and relax them after work hours or at lunch. Conversely, you could essentially shut your network down during non-work hours.

You cannot create time-based rules that have the exact same protocol, source, destination, and service criteria of a rule that does not include a time range object. The non-time-based rule always overrides the duplicate time-based rule, as they are redundant.

Note Users could experience a delay of approximately 80 to 100 seconds after the specified end time for the ACL to become inactive. For example, if the specified end time is 3:50, because the end time is inclusive, the command is picked up anywhere between 3:51:00 and 3:51:59. After the command is picked up, the ASA finishes any currently running task and then services the command to deactivate the ACL.

Licensing for Access Control Lists

Access control lists do not require a special license.

However, to use **sctp** as the protocol in an entry, you must have a Carrier license.

Guidelines for ACLs

Firewall Mode

- Extended and standard ACLs are supported in routed and transparent firewall modes.
- Webtype ACLs are supported in routed mode only.
- EtherType ACLs are supported for bridge group member interfaces only, in routed and transparent modes.

Failover and Clustering

Configuration sessions are not synchronized across failover or clustered units. When you commit the changes in a session, they are made in all failover and cluster units as normal.

IPv6

- Extended and webtype ACLs allow a mix of IPv4 and IPv6 addresses.

- Standard ACLs do not allow IPv6 addresses.
- EtherType ACLs do not contain IP addresses.

Additional Guidelines

- When you specify a network mask, the method is different from the Cisco IOS software **access-list** command. The ASA uses a network mask (for example, 255.255.255.0 for a Class C mask). The Cisco IOS mask uses wildcard bits (for example, 0.0.0.255).
- Normally, you cannot reference an object or object group that does not exist in an ACL or object group, or delete one that is currently referenced. You also cannot reference an ACL that does not exist in an **access-group** command (to apply access rules). However, you can change this default behavior so that you can "forward reference" objects or ACLs before you create them. Until you create the objects or ACLs, any rules or access groups that reference them are ignored. To enable forward referencing, use the **forward-reference enable** command.
- (Extended ACL only) The following features use ACLs, but cannot accept an ACL with identity firewall (specifying user or group names), FQDN (fully-qualified domain names), or Cisco TrustSec values:
  - VPN **crypto map** command
  - VPN **group-policy** command, except for **vpn-filter**
  - WCCP
  - DAP

Configure ACLs

The following sections explain how to configure the various types of ACL, Read the section on ACL basics to get the big picture, then the sections on specific types of ACL for the details.

Basic ACL Configuration and Management Options

An ACL is made up of one or more access control entries (ACEs) with the same ACL ID or name. To create a new ACL, you simply create an ACE with a new ACL name, and it becomes the first rule in the new ACL.

Working with an ACL, you can do the following things:

Examine the ACL contents and determine line numbers and hit counts

Use the **show access-list** *name* command to view the contents of the ACL. Each row is an ACE, and includes the line number, which you will need to know if you want to insert new entries into an extended ACL. The information also includes a hit count for each ACE, which is how many times the rule was matched by traffic. For example:

```
hostname# show access-list outside_access_in
access-list outside_access_in; 3 elements; name hash: 0x6892a938
access-list outside_access_in line 1 extended permit ip 10.2.2.0 255.255.255.0 any
(hitcnt=0) 0xcc48b55c
access-list outside_access_in line 2 extended permit ip host
2001:DB8::0DB8:800:200C:417A any (hitcnt=0) 0x79797f94
```

```
access-list outside_access_in line 3 extended permit ip user-group
LOCAL\\usergroup any any (hitcnt=0) 0xb0f5b1e1
```

Add an ACE

The command for adding an ACE is **access-list** *name* [**line** *line-num*] *type parameters*. The line number argument works for extended ACLs only. If you include the line number, the ACE is inserted at that location in the ACL, and the ACE that was at that location is moved down, along with the remainder of the ACEs (that is, inserting an ACE at a line number does not replace the old ACE at that line). If you do not include a line number, the ACE is added to the end of the ACL. The parameters available differ based on the ACL type; see the specific topics on each ACL type for details.

Add comments to an ACL (all types except webtype)

Use the **access-list** *name* [**line** *line-num*] **remark** *text* command to add remarks into an ACL to help explain the purpose of an ACE. Best practice is to insert the remark before the ACE; if you view the configuration in ASDM, remarks will be associated with the ACE that follows the remarks. You can enter multiple remarks before an ACE to include an expanded comment. Each remark is limited to 100 characters. You can include leading spaces to help set off the remarks. If you do not include a line number, the remark is added to the end of the ACL. For example, you could add remarks before adding each ACE:

```
hostname(config)# access-list OUT remark - this is the inside admin address
hostname(config)# access-list OUT extended permit ip host 209.168.200.3 any
hostname(config)# access-list OUT remark - this is the hr admin address
hostname(config)# access-list OUT extended permit ip host 209.168.200.4 any
```

Edit or move an ACE or remark

You cannot edit or move an ACE or remark. Instead, you must create a new ACE or remark with the desired values at the right location (using the line number), then delete the old ACE or remark. Because you can insert ACEs in extended ACLs only, you need to rebuild standard, webtype, or EtherType ACLs if you need to edit or move ACEs. It is far easier to reorganize a long ACL using ASDM.

Delete an ACE or remark

Use the **no access-list** *parameters* command to remove an ACE or remark. Use the **show access-list** command to view the parameter string that you must enter: the string must exactly match an ACE or remark to delete it, with the exception of the **line** *line-num* argument, which is optional on the **no access-list** command.

Delete an entire ACL, including remarks

Use the **clear configure access-list** *name* command. USE CAUTION! The command does not ask you for confirmation. If you do not include a name, every access list on the ASA is removed.

Rename an ACL

Use the **access-list** *name* **rename** *new_name* command.

Apply the ACL to a policy

Creating an ACL in and of itself does nothing to traffic. You must apply the ACL to a policy. For example, you can use the **access-group** command to apply an extended ACL to an interface, thus denying or permitting traffic that goes through the interface.

Configure Extended ACLs

An extended ACL is composed of all ACEs with the same ACL ID or name. Extended ACLs are the most complex and feature-rich type of ACL, and you can use them for many features. The most noteworthy use of extended ACLs is as access groups applied globally or to interfaces, which determine the traffic that will be denied or permitted to flow through the box. But extended ACLs are also used to determine the traffic to which other services will be provided.

Because extended ACLs are complex, the following sections focus on creating ACEs to provide specific types of traffic matching. The first sections, on basic address-based ACEs and on TCP/UDP ACEs, build the foundation for the remaining sections.

Add an Extended ACE for IP Address or Fully-Qualified Domain Name-Based Matching

The basic extended ACE matches traffic based on source and destination addresses, including IPv4 and IPv6 addresses and fully-qualified domain names (FQDN), such as www.example.com. In fact, every type of extended ACE must include some specification for source and destination address, so this topic explains the minimum extended ACE.

Tip Tip If you want to match traffic based on FQDN, you must create a network object for each FQDN.

To add an ACE for IP address or FQDN matching, use the following command:

access-list *access_list_name* [**line** *line_number*] **extended** {**deny** | **permit**} *protocol_argument source_address_argument dest_address_argument* [**log** [[*level*] [**interval** *secs*] | **disable** | **default**]] [**time-range** *time_range_name*] [**inactive**]

Example:

```
hostname(config)# access-list ACL_IN extended permit ip any any
hostname(config)# access-list ACL_IN extended permit object service-obj-http any any
```

The options are:

- *access_list_name*—The name of the new or existing ACL.
- Line number—The **line** *line_number* option specifies the line number at which insert the ACE; otherwise, the ACE is added to the end of the ACL.
- Permit or Deny—The **deny** keyword denies or exempts a packet if the conditions are matched. The **permit** keyword permits or includes a packet if the conditions are matched.
- Protocol—The *protocol_argument* specifies the IP protocol:
  - *name* or *number*—Specifies the protocol name or number. Specify **ip** to apply to all protocols.
  - **object-group** *protocol_grp_id*—Specifies a protocol object group created using the **object-group protocol** command.
  - **object** *service_obj_id*—Specifies a service object created using the **object service** command. The object can include port or ICMP type and code specifications if desired.
  - **object-group** *service_grp_id*—Specifies a service object group created using the **object-group service** command.

- Source Address, Destination Address—The *source_address_argument* specifies the IP address or FQDN from which the packet is being sent, and the *dest_address_argument* specifies the IP address or FQDN to which the packet is being sent:
  - **host** *ip_address*—Specifies an IPv4 host address.
  - *ip_address mask*—Specifies an IPv4 network address and subnet mask, such as 10.100.10.0 255.255.255.0.
  - *ipv6-address*/*prefix-length*—Specifies an IPv6 host or network address and prefix.
  - **any**, **any4**, and **any6**—**any** specifies both IPv4 and IPv6 traffic; **any4** specifies IPv4 traffic only; and **any6** specifies IPv6 traffic only.
  - **interface** *interface_name*—Specifies the name of an ASA interface. Use the interface name rather than IP address to match traffic based on which interface is the source or destination of the traffic.
  - **object** *nw_obj_id*—Specifies a network object created using the **object network** command.
  - **object-group** *nw_grp_id*—Specifies a network object group created using the **object-group network** command.
- Logging—**log** arguments set logging options when an ACE matches a connection for network access (an ACL applied with the **access-group** command). If you enter the **log** option without any arguments, you enable syslog message 106100 at the default level (6) and for the default interval (300 seconds). Log options are:
  - *level*—A severity level between 0 and 7. The default is 6 (informational). If you change this level for an active ACE, the new level applies to new connections; existing connections continue to be logged at the previous level.
  - **interval** *secs*—The time interval in seconds between syslog messages, from 1 to 600. The default is 300. This value is also used as the timeout value for deleting an inactive flow from the cache used to collect drop statistics.
  - **disable**—Disables all ACE logging.
  - **default**—Enables logging to message 106023 for denied packets. This setting is the same as not including the **log** option.
- Time Range—The **time-range** *time_range_name* option specifies a time range object, which determines the times of day and days of the week in which the ACE is active. If you do not include a time range, the ACE is always active.
- Activation—Use the **inactive** option to disable the ACE without deleting it. To reenable it, enter the entire ACE without the inactive keyword.

Add an Extended ACE for Port-Based Matching

If you specify service objects in an ACE, the service objects can include protocols with port specifications, such as TCP/80. Alternatively, you can specify the ports directly in the ACE. With port-based matching, you can target certain types of traffic for port-based protocols rather than all traffic for the protocol.

The port-based extended ACE is just the basic address-matching ACE where the protocol is **tcp**, **udp**, or **sctp**. To add port specifications, use the following command:

access-list *access_list_name* [**line** *line_number*] **extended** {**deny** | **permit**} {**tcp** | **udp** | **sctp**} *source_address_argument* [*port_argument*] *dest_address_argument* [*port_argument*] [**log** [[*level*] [**interval** *secs*] | **disable** | **default**] [**time-range** *time-range-name*] [**inactive**]

Example:

```
hostname(config)# access-list ACL_IN extended deny tcp any host 209.165.201.29 eq www
```

The *port_argument* option specifies the source or destination port. If you do not specify ports, all ports are matched. Available arguments include:

- *operator port*—The *port* can be the integer or name of a port. The *operator* can be one of the following:
  - **lt**—less than
  - **gt**—greater than
  - **eq**—equal to
  - **neq**—not equal to
  - **range**—an inclusive range of values. When you use this operator, specify two port numbers, for example, **range 100 200**.

Note DNS, Discard, Echo, Ident, NTP, RPC, SUNRPC, and Talk each require one definition for TCP and one for UDP. TACACS+ requires one definition for port 49 on TCP.

- **object-group** *service_grp_id*—Specifies a service object group created using the **object-group service** {**tcp** | **udp** | **tcp-udp**} command. Note that these object types are no longer recommended.

  You cannot specify the recommended generic service objects, where the protocol and port are defined within the object, as the port argument. You specify these objects as part of the protocol argument, as explained in Add an Extended ACE for IP Address or Fully-Qualified Domain Name-Based Matching, on page 8.

For an explanation of the other keywords, and how to use service objects to specify protocols and ports, see Add an Extended ACE for IP Address or Fully-Qualified Domain Name-Based Matching, on page 8.

Add an Extended ACE for ICMP-Based Matching

If you specify service objects in an ACE, the service objects can include the ICMP/ICMP6 protocols ICMP type and code specifications. Alternatively, you can specify the ICMP type and code directly in the ACE. For example, you can target ICMP Echo Request traffic (pings).

The ICMP extended ACE is just the basic address-matching ACE where the protocol is **icmp** or **icmp6**. Because these protocols have type and code values, you can add type and code specifications to the ACE.

To add an ACE for IP address or FQDN matching, where the protocol is ICMP or ICMP6, use the following command:

access-list *access_list_name* [**line** *line_number*] **extended** {**deny** | **permit**} {**icmp** | **icmp6**} *source_address_argument dest_address_argument* [*icmp_argument*] [**log** [[*level*] [**interval** *secs*] | **disable** | **default**]] [**time-range** *time_range_name*] [**inactive**]

Example:

```
hostname(config)# access-list abc extended permit icmp any any object-group obj_icmp_1
hostname(config)# access-list abc extended permit icmp any any echo
```

The *icmp_argument* option specifies the ICMP type and code.

- *icmp_type* [*icmp_code*]—Specifies the ICMP type by name or number, and the optional ICMP code for that type. If you do not specify the code, then all codes are used.
- **object-group** *icmp_grp_id*—Specifies an object group for ICMP/ICMP6 created using the (deprecated) **object-group icmp-type** command.

  You cannot specify the recommended generic service objects, where the protocol and type are defined within the object, as the ICMP argument. You specify these objects as part of the protocol argument, as explained in Add an Extended ACE for IP Address or Fully-Qualified Domain Name-Based Matching, on page 8.

For an explanation of the other keywords, see Add an Extended ACE for IP Address or Fully-Qualified Domain Name-Based Matching, on page 8.

Add an Extended ACE for User-Based Matching (Identity Firewall)

The user-based extended ACE is just the basic address-matching ACE where you include username or user group to the source matching criteria. By creating rules based on user identity, you can avoid tying rules to static host or network addresses. For example, if you define a rule for user1, and the identity firewall feature maps that user to a host assigned 10.100.10.3 one day, but 192.168.1.5 the next day, the user-based rule still applies.

Because you must still supply source and destination addresses, broaden the source address to include the likely addresses that will be assigned to the user (normally through DHCP). For example, user "LOCAL\user1 any" will match the LOCAL\user1 user no matter what address is assigned, whereas "LOCAL\user1 10.100.1.0 255.255.255.0" matches the user only if the address is on the 10.100.1.0/24 network.

By using group names, you can define rules based on entire classes of users, such as students, teachers, managers, engineers, and so forth.

To add an ACE for user or group matching, use the following command:

access-list *access_list_name* [**line** *line_number*] **extended** {**deny** | **permit**} *protocol_argument* [*user_argument*] *source_address_argument* [*port_argument*] *dest_address_argument* [*port_argument*] [**log** [[*level*] [**interval** *secs*] | **disable** | **default**]] [**time-range** *time_range_name*] [**inactive**]

Example:

```
hostname(config)# access-list v1 extended permit ip user LOCAL\idfw
any 10.0.0.0 255.255.255.0
```

The *user_argument* option specifies the user or group for which to match traffic in addition to the source address. Available arguments include the following:

- **object-group-user** *user_obj_grp_id*—Specifies a user object group created using the **object-group user** command.
- **user** {[*domain_nickname*\]*name* | **any** | **none**}—Specifies a username. Specify **any** to match all users with user credentials, or **none** to match addresses that are not mapped to usernames. These options are especially useful for combining **access-group** and **aaa authentication match** policies.
- **user-group** [*domain_nickname*\\]*user_group_name*—Specifies a user group name. Note the double \\ separating the domain and group name.

For an explanation of the other keywords, see Add an Extended ACE for IP Address or Fully-Qualified Domain Name-Based Matching, on page 8.

Tip You can include both user and Cisco Trustsec security groups in a given ACE.

Add an Extended ACE for Security Group-Based Matching (Cisco TrustSec)

The security group (Cisco TrustSec) extended ACE is just the basic address-matching ACE where you include security groups or tags to the source or destination matching criteria. By creating rules based on security groups, you can avoid tying rules to static host or network addresses. Because you must still supply source and destination addresses, broaden the addresses to include the likely addresses that will be assigned to users (normally through DHCP).

Tip Before adding this type of ACE, configure Cisco TrustSec.

To add an ACE for security group matching, use the following command:

access-list *access_list_name* [**line** *line_number*] **extended** {**deny** | **permit**} *protocol_argument* [*security_group_argument*] *source_address_argument* [*port_argument*] [*security_group_argument*] *dest_address_argument* [*port_argument*] [**log** [[*level*] [**interval** *secs*] | **disable** | **default**]] [**inactive** | **time-range** *time_range_name*]

Example:

```
hostname(config)# access-list INSIDE_IN extended permit ip
security-group name my-group any any
```

The *security_group_argument* option specifies the security group for which to match traffic in addition to the source or destination address. Available arguments include the following:

- **object-group-security** *security_obj_grp_id*—Specifies a security object group created using the **object-group security** command.
- **security-group** {**name** *security_grp_id* | **tag** *security_grp_tag*}—Specifies a security group name or tag.

For an explanation of the other keywords, see Add an Extended ACE for IP Address or Fully-Qualified Domain Name-Based Matching, on page 8.

Tip You can include both user and Cisco Trustsec security groups in a given ACE.

Examples for Extended ACLs

The following ACL allows all hosts (on the interface to which you apply the ACL) to go through the ASA:

```
hostname(config)# access-list ACL_IN extended permit ip any any
```

The following ACL prevents hosts on 192.168.1.0/24 from accessing the 209.165.201.0/27 network for TCP-based traffic. All other addresses are permitted.

```
hostname(config)# access-list ACL_IN extended deny tcp 192.168.1.0 255.255.255.0
209.165.201.0 255.255.255.224
hostname(config)# access-list ACL_IN extended permit ip any any
```

If you want to restrict access to selected hosts only, then enter a limited permit ACE. By default, all other traffic is denied unless explicitly permitted.

```
hostname(config)# access-list ACL_IN extended permit ip 192.168.1.0 255.255.255.0
209.165.201.0 255.255.255.224
```

The following ACL restricts all hosts (on the interface to which you apply the ACL) from accessing a website at address 209.165.201.29. All other traffic is allowed.

```
hostname(config)# access-list ACL_IN extended deny tcp any host 209.165.201.29 eq www
hostname(config)# access-list ACL_IN extended permit ip any any
```

The following ACL that uses object groups restricts several hosts on the inside network from accessing several web servers. All other traffic is allowed.

```
hostname(config-network)# access-list ACL_IN extended deny tcp object-group denied
object-group web eq www
hostname(config)# access-list ACL_IN extended permit ip any any
hostname(config)# access-group ACL_IN in interface inside
```

The following example temporarily disables an ACL that permits traffic from one group of network objects (A) to another group of network objects (B):

```
hostname(config)# access-list 104 permit ip host object-group A object-group B inactive
```

To implement a time-based ACE, use the **time-range** command to define specific times of the day and week. Then use the **access-list extended** command to bind the time range to an ACE. The following example binds an ACE in the “Sales” ACL to a time range named “New_York_Minute.”

```
hostname(config)# access-list Sales line 1 extended deny tcp host 209.165.200.225 host
209.165.201.1 time-range New_York_Minute
```

The following example shows a mixed IPv4/IPv6 ACL:

```
hostname(config)# access-list demoacl extended permit ip 2001:DB8:1::/64 10.2.2.0
255.255.255.0
hostname(config)# access-list demoacl extended permit ip 2001:DB8:1::/64 2001:DB8:2::/64
hostname(config)# access-list demoacl extended permit ip host 10.3.3.3 host 10.4.4.4
```

Example of Converting Addresses to Objects for Extended ACLs

The following normal ACL that does not use object groups restricts several hosts on the inside network from accessing several web servers. All other traffic is allowed.

```
hostname(config)# access-list ACL_IN extended deny tcp host 10.1.1.4 host 209.165.201.29
eq www
hostname(config)# access-list ACL_IN extended deny tcp host 10.1.1.78 host 209.165.201.29
eq www
hostname(config)# access-list ACL_IN extended deny tcp host 10.1.1.89 host 209.165.201.29
eq www
hostname(config)# access-list ACL_IN extended deny tcp host 10.1.1.4 host 209.165.201.16
eq www
hostname(config)# access-list ACL_IN extended deny tcp host 10.1.1.78 host 209.165.201.16
eq www
hostname(config)# access-list ACL_IN extended deny tcp host 10.1.1.89 host 209.165.201.16
eq www
hostname(config)# access-list ACL_IN extended deny tcp host 10.1.1.4 host 209.165.201.78
eq www
hostname(config)# access-list ACL_IN extended deny tcp host 10.1.1.78 host 209.165.201.78
eq www
hostname(config)# access-list ACL_IN extended deny tcp host 10.1.1.89 host 209.165.201.78
eq www
hostname(config)# access-list ACL_IN extended permit ip any any
hostname(config)# access-group ACL_IN in interface inside
```

If you make two network object groups, one for the inside hosts, and one for the web servers, then the configuration can be simplified and can be easily modified to add more hosts:

```
hostname(config)# object-group network denied
hostname(config-network)# network-object host 10.1.1.4
hostname(config-network)# network-object host 10.1.1.78
hostname(config-network)# network-object host 10.1.1.89

hostname(config-network)# object-group network web
hostname(config-network)# network-object host 209.165.201.29
hostname(config-network)# network-object host 209.165.201.16
hostname(config-network)# network-object host 209.165.201.78

hostname(config)# access-list ACL_IN extended deny tcp object-group denied object-group
web eq www
hostname(config)# access-list ACL_IN extended permit ip any any
hostname(config)# access-group ACL_IN in interface inside
```

Configure Standard ACLs

A standard ACL is composed of all ACEs with the same ACL ID or name. Standard ACLs are used for a limited number of features, such as route maps or VPN filters. A standard ACL uses IPv4 addresses only, and defines destination addresses only.

To add a standard access list entry, use the following command:

access-list *access_list_name* **standard** {**deny** | **permit**} {**any4** | **host** *ip_address* | *ip_address mask*}

Example:

```
hostname(config)# access-list OSPF standard permit 192.168.1.0 255.255.255.0
```

The options are:

- Name—The *access_list_name* argument specifies the name of number of an ACL. Traditional numbers for standard ACLs are 1-99 or 1300-1999, but you can use any name or number. You create a new ACL if the ACL does not already exist, otherwise, you are adding the entry to the end of the ACL.
- Permit or Deny—The **deny** keyword denies or exempts a packet if the conditions are matched. The **permit** keyword permits or includes a packet if the conditions are matched.
- Destination Address—The **any4** keyword matches all IPv4 addresses. The **host** *ip_address* argument matches a host IPv4 address. The *ip_address ip_mask* argument matches an IPv4 subnet, for example, 10.1.1.0 255.255.255.0.

Configure Webtype ACLs

Webtype ACLs are used for filtering clientless SSL VPN traffic, constraining user access to specific networks, subnets, hosts, and Web servers. If you do not define a filter, all connections are allowed. A webtype ACL is composed of all ACEs with the same ACL ID or name.

With webtype ACLs, you can match traffic based on URLs or destination addresses. A single ACE cannot mix these specifications. The following sections explain each type of ACE.

Add a Webtype ACE for URL Matching

To match traffic based on the URL the user is trying to access, use the following command;

access-list *access_list_name* **webtype** {**deny** | **permit**} **url** {*url_string* | **any**} [**log** [[*level*] [**interval** *secs*] | **disable** | **default**]] [**time_range** *time_range_name*] [**inactive**]

Example:

```
hostname(config)# access-list acl_company webtype deny url http://*.example.com
```

The options are:

- *access_list_name*—The name of the new or existing ACL. If the ACL already exists, you are adding the ACE to the end of the ACL.
- Permit or Deny—The **deny** keyword denies or exempts a packet if the conditions are matched. The **permit** keyword permits or includes a packet if the conditions are matched.
- URL—The **url** keyword specifies the URL to match. Use **url any** to match all URL-based traffic. Otherwise, enter a URL string, which can include wildcards. Following are some tips and limitations on specifying URLs:
  - Specify **any** to match all URLs.
  - 'Permit url any' will allow all the URLs that have the format protocol://server-ip/path and will block traffic that does not match this pattern, such as port-forwarding. There should be an ACE to allow

connections to the required port (port 1494 in the case of Citrix) so that an implicit deny does not occur.

- Smart tunnel and ica plug-ins are not affected by an ACL with 'permit url any' because they match smart-tunnel:// and ica:// types only.
- You can use these protocols: cifs://, citrix://, citrixs://, ftp://, http://, https://, imap4://, nfs://, pop3://, smart-tunnel://, and smtp://. You can also use wildcards in the protocol; for example, htt* matches http and https, and an asterisk * matches all protocols. For example, *://*.example.com matches any type URL-based traffic to the example.com network.
- If you specify a smart-tunnel:// URL, you can include the server name only. The URL cannot contain a path. For example, smart-tunnel://www.example.com is acceptable, but smart-tunnel://www.example.com/index.html is not.
- An asterisk * matches none or any number of characters. To match any http URL, enter http://*/*.
- A question mark ? matches any one character exactly.
- Square brackets [] are range operators, matching any character in the range. For example, to match both http://www.cisco.com:80/ and http://www.cisco.com:81/, enter **http://www.cisco.com:8[01]/**.

- Logging—**log** arguments set logging options when an ACE matches a packet. If you enter the **log** option without any arguments, you enable syslog message 106102 at the default level (6) and for the default interval (300 seconds). Log options are:
  - *level*—A severity level between 0 and 7. The default is 6.
  - **interval** *secs*—The time interval in seconds between syslog messages, from 1 to 600. The default is 300.
  - **disable**—Disables all ACL logging.
  - **default**—Enables logging to message 106103. This setting is the same as not including the **log** option.
- Time Range—The **time-range** *time_range_name* option specifies a time range object, which determines the times of day and days of the week in which the ACE is active. If you do not include a time range, the ACE is always active.
- Activation—Use the **inactive** option to disable the ACE without deleting it. To reenable it, enter the entire ACE without the inactive keyword.

Add a Webtype ACE for IP Address Matching

You can match traffic based on the destination address the user is trying to access. The webtype ACL can include a mix of IPv4 and IPv6 addresses in addition to URL specifications.

To add a webtype ACE for IP address matching, use the following command:

access-list *access_list_name* **webtype** {**deny** | **permit**} **tcp** *dest_address_argument* [*operator port*] [**log** [[*level*] [**interval** *secs*] | **disable** | **default**]] [**time_range** *time_range_name*]] [**inactive**]]

Example:

```
hostname(config)# access-list acl_company webtype permit tcp any
```

For an explanation of keywords not explained here, see Add a Webtype ACE for URL Matching, on page 15. Keywords and arguments specific to this type of ACE include the following:

- **tcp**—The TCP protocol. Webtype ACLs match TCP traffic only.
- Destination Address—The *dest_address_argument* specifies the IP address to which the packet is being sent:
  - **host** *ip_address*—Specifies an IPv4 host address.
  - *dest_ip_address mask*—Specifies an IPv4 network address and subnet mask, such as 10.100.10.0 255.255.255.0.
  - *ipv6-address/prefix-length*—Specifies an IPv6 host or network address and prefix.
  - **any**, **any4**, and **any6**—**any** specifies both IPv4 and IPv6 traffic; **any4** specifies IPv4 traffic only; and **any6** specifies IPv6 traffic only.
- *operator port*—The destination port. If you do not specify ports, all ports are matched. The *port* can be the integer or name of a TCP port. The *operator* can be one of the following:
  - **lt**—less than
  - **gt**—greater than
  - **eq**—equal to
  - **neq**—not equal to
  - **range**—an inclusive range of values. When you use this operator, specify two port numbers, for example, **range 100 200**.

Examples for Webtype ACLs

The following example shows how to deny access to a specific company URL:

```
hostname(config)# access-list acl_company webtype deny url http://*.example.com
```

The following example shows how to deny access to a specific web page:

```
hostname(config)# access-list acl_file webtype deny url https://www.example.com/dir/file.html
```

The following example shows how to deny HTTP access to any URL on a specific server through port 8080:

```
hostname(config)# access-list acl_company webtype deny url http://my-server:8080/*
```

The following examples show how to use wildcards in webtype ACLs.

- The following example matches URLs such as http://www.example.com/layouts/1033:

  ```
  access-list VPN-Group webtype permit url http://www.example.com/*
  ```

- The following example matches URLs such as http://www.example.com/ and http://www.example.net/:

```
access-list test webtype permit url http://www.example.*
```

- The following example matches URLs such as http://www.example.com and ftp://wwz.example.com:

```
access-list test webtype permit url *://ww?.e*co*/
```

- The following example matches URLs such as http://www.cisco.com:80 and https://www.cisco.com:81:

```
access-list test webtype permit url *://ww?.c*co*:8[01]/
```

  The range operator "[]" in the preceding example specifies that either character **0** or **1** can occur at that location.

- The following example matches URLs such as http://www.example.com and http://www.example.net:

```
access-list test webtype permit url http://www.[a-z]xample?*/
```

  The range operator "[]" in the preceding example specifies that any character in the range from **a** to **z** can occur.

- The following example matches http or https URLs that include "cgi" somewhere in the file name or path.

```
access-list test webtype permit url htt*://*/*cgi?*
```

Note To match any http URL, you must enter **http://*/*** instead of http://*.

The following example shows how to enforce a webtype ACL to disable access to specific CIFS shares.

In this scenario we have a root folder named "shares" that contains two sub-folders named "Marketing_Reports" and "Sales_Reports." We want to specifically deny access to the "shares/Marketing_Reports" folder.

```
access-list CIFS_Avoid webtype deny url cifs://172.16.10.40/shares/Marketing_Reports.
```

However, due to the implicit "deny all" at the end of the ACL, the above ACL makes all of the sub-folders inaccessible ("shares/Sales_Reports" and "shares/Marketing_Reports"), including the root folder ("shares").

To fix the problem, add a new ACL to allow access to the root folder and the remaining sub-folders:

```
access-list CIFS_Allow webtype permit url cifs://172.16.10.40/shares*
```

Configure EtherType ACLs

EtherType ACLs apply to non-IP layer-2 traffic on bridge group member interfaces. You can use these rules to permit or drop traffic based on the EtherType value in the layer-2 packet. With EtherType ACLs, you can

control the flow of non-IP traffic across the bridge group. Note that 802.3-formatted frames are not handled by the ACL because they use a length field as opposed to a type field.

To add an EtherType ACE, use the following command:

access-list *access_list_name* **ethertype** {**deny** | **permit**} {**any** | **bpdu** | **dsap** {*hex_address* | **bpdu** | **ipx** | **isis** | **raw-ipx**} | **eii-ipx** | **ipx** | **isis** | **mpls-multicast** | **mpls-unicast** | *hex_number*}

Example:

```
hostname(config)# access-list ETHER ethertype deny mpls-multicast
```

The options are:

- *access_list_name*—The name of the new or existing ACL. If the ACL already exists, you are adding the ACE to the end of the ACL.
- Permit or Deny—The **deny** keyword denies a packet if the conditions are matched. The **permit** keyword permits a packet if the conditions are matched.
- Traffic Matching Criteria—You can match traffic using the following options:
  - **any**—Matches all layer 2 traffic.
  - **bpdu**—bridge protocol data units (dsap 0x42), which are allowed by default. This keyword is converted to **dsap bpdu**.
  - **dsap** {*hex_address* | **bpdu** | **ipx** | **isis** | **raw-ipx**}—The IEEE 802.2 Logical Link Control (LLC) packet's Destination Service Access Point address. Include the address you want to permit or deny in hexadecimal, from 0x01 to 0xff. You can also use the following keywords to create rules for common values:
    - **bpdu** for 0x42, bridge protocol data units.
    - **ipx** for 0xe0, Internet Packet Exchange (IPX) 802.2 LLC.
    - **isis** for 0xfe, Intermediate System to Intermediate System (IS-IS).
    - **raw-ipx** for 0xff, raw IPX 802.3 format.
  - **eii-ipx**—Ethernet II IPX format, EtherType 0x8137.
  - **ipx**—Internet Packet Exchange (IPX). This keyword is a shortcut for configuring three separate rules, for **dsap ipx**, **dsap raw-ipx**, and **eii-ipx**.
  - **isis**—Intermediate System to Intermediate System (IS-IS). This keyword is converted to **dsap isis**.
  - **mpls-multicast**— MPLS multicast.
  - **mpls-unicast**—MPLS unicast.
  - *hex_number*—Any EtherType that can be identified by a 16-bit hexadecimal number 0x600 to 0xffff. See RFC 1700, “Assigned Numbers,” at http://www.ietf.org/rfc/rfc1700.txt for a list of EtherTypes.

Examples for EtherType ACLs

The following examples shows how to configure EtherType ACLs, including how to apply them to an interface.

For example, the following sample ACL allows common EtherTypes originating on the inside interface:

```
hostname(config)# access-list ETHER ethertype permit ipx
INFO: ethertype ipx is saved to config as ethertype eii-ipx
INFO: ethertype ipx is saved to config as ethertype dsap ipx
INFO: ethertype ipx is saved to config as ethertype dsap raw-ipx
hostname(config)# access-list ETHER ethertype permit mpls-unicast
hostname(config)# access-group ETHER in interface inside
```

The following example allows some EtherTypes through the ASA, but it denies all others:

```
hostname(config)# access-list ETHER ethertype permit 0x1234
hostname(config)# access-list ETHER ethertype permit mpls-unicast
hostname(config)# access-group ETHER in interface inside
hostname(config)# access-group ETHER in interface outside
```

The following example denies traffic with EtherType 0x1256 but allows all others on both interfaces:

```
hostname(config)# access-list nonIP ethertype deny 1256
hostname(config)# access-list nonIP ethertype permit any
hostname(config)# access-group nonIP in interface inside
hostname(config)# access-group nonIP in interface outside
```

Edit ACLs in an Isolated Configuration Session

When you edit an ACL used for access rules or any other purpose, the change is immediately implemented and impacts traffic. With access rules, you can enable the transactional commit model to ensure that new rules become active only after rule compilation is complete, but the compilation happens after each ACE you edit.

If you want to further isolate the impact of editing ACLs, you can make your changes in a "configuration session," which is an isolated mode that allows you to edit several ACEs and objects before explicitly committing your changes. Thus, you can ensure that all of your intended changes are complete before you change device behavior.

Before you begin

- You can edit ACLs that are referenced by an access-group command, but you cannot edit ACLs that are referenced by any other command. You can also edit unreferenced ACLs or create new ones.
- You can create or edit objects and object groups, but if you create one in a session, you cannot edit it in the same session. If the object is not defined as desired, you must commit your changes and then edit the object, or discard the entire session and start over.
- When you edit an ACL that is referenced by an access-group command (access rules), the transactional commit model is used when you commit the session. Thus, the ACL is completely compiled before the new ACL replaces the old version.
- If you enable forward referencing of ACL and object names (the **forward-reference enable** command), you can delete an ACL that is referenced by an access-group command (access rules), and then recreate

the ACL. When you commit changes, the new version of the ACL will be used after compilation is complete. You can also create rules that refer to objects that do not exist, or delete objects that are in use by access rules. However, you will get a commit error if you delete an object used by other rules, such as NAT.

Procedure

Step 1 Start the session.

```
hostname#configure session session_name
hostname(config-s)#
```

If the *session_name* already exists, you open that session. Otherwise, you are creating a new session.

Use the **show configuration session** command to view the existing sessions. You can have at most 3 sessions active at a time. If you need to delete an old unused session, use the **clear configuration session** *session_name* command.

If you cannot open an existing session because someone else is editing it, you can clear the flag that indicates the session is being edited. Do this only if you are certain the session is not actually being edited. Use the **clear session** *session_name* **access** command to reset the flag.

Step 2 (Uncommitted sessions only.) Make your changes. You can use the following basic commands with any of their parameters:

- **access-list**
- **object**
- **object-group**

Step 3 Decide what to do with the session. The commands available depend on whether you have previously committed the session. Possible commands are:

- **exit**—To simply exit the session without committing or discarding changes, so that you can return later.
- **commit** [**noconfirm** [**revert-save** | **config-save**]]—(Uncommitted sessions only.) To commit your changes. You are asked if you want to save the session. You can save the revert session (**revert-save**), which lets you undo your changes using the **revert** command, or the configuration session (**config-save**), which includes all of the changes made in the session (allowing you to commit the same changes again if you would like to). If you save the revert or configuration session, the changes are committed, but the session remains active. You can open the session and revert or recommit the changes. You can avoid the prompt by including the **noconfirm** option and optionally, the desired save option.
- **abort**—(Uncommitted sessions only.) To abandon your changes and delete the session. If you want to keep the session, exit the session and use the **clear session** *session_name* **configuration** command, which empties the session without deleting it.
- **revert**—(Committed sessions only.) To undo your changes, returning the configuration back to what it was before you committed the session, and delete the session.

- **show configuration session** [*session_name*]—To show the changes made in the session.

Monitoring ACLs

To monitor ACLs, enter one of the following commands:

- **show access-list** [*name*]—Displays the access lists, including the line number for each ACE and hit counts. Include an ACL name or you will see all access lists.
- **show running-config access-list** [*name*]—Displays the current running access-list configuration. Include an ACL name or you will see all access lists.

History for ACLs

Feature Name	Releases	Description
Extended, standard, webtype ACLs	7.0(1)	ACLs are used to control network access or to specify traffic for many features to act upon. An extended access control list is used for through-the-box access control and several other features. Standard ACLs are used in route maps and VPN filters. Webtype ACLs are used in clientless SSL VPN filtering. EtherType ACLs control non-IP layer 2 traffic. We introduced the following commands: **access-list extended, access-list standard, access-list webtype**, **access-list ethertype**.
Real IP addresses in extended ACLs	8.3(1)	When using NAT or PAT, mapped addresses and ports are no longer used in an ACL for several features. You must use the real, untranslated addresses and ports for these features. Using the real address and port means that if the NAT configuration changes, you do not need to change the ACLs.
Support for Identity Firewall in extended ACLs	8.4(2)	You can now use identity firewall users and groups for the source and destination. You can use an identity firewall ACL with access rules, AAA rules, and for VPN authentication. We modified the following commands: **access-list extended**.
EtherType ACL support for IS-IS traffic	8.4(5), 9.1(2)	In transparent firewall mode, the ASA can now control IS-IS traffic using an EtherType ACL. We modified the following command: **access-list ethertype** {**permit** | **deny**} **isis**.
Support for Cisco TrustSec in extended ACLs	9.0(1)	You can now use Cisco TrustSec security groups for the source and destination. You can use an identity firewall ACL with access rules. We modified the following commands: **access-list extended**.

Feature Name	Releases	Description
Unified extended and webtype ACLs for IPv4 and IPv6	9.0(1)	Extended and webtype ACLs now support IPv4 and IPv6 addresses. You can even specify a mix of IPv4 and IPv6 addresses for the source and destination. The **any** keyword was changed to represent IPv4 and IPv6 traffic. The **any4** and **any6** keywords were added to represent IPv4-only and IPv6-only traffic, respectively. The IPv6-specific ACLs are deprecated. Existing IPv6 ACLs are migrated to extended ACLs. See the release notes for more information about migration. We modified the following commands: **access-list extended**, **access-list webtype**. We removed the following commands: **ipv6 access-list**, **ipv6 access-list webtype**, **ipv6-vpn-filter**.
Extended ACL and object enhancement to filter ICMP traffic by ICMP code	9.0(1)	ICMP traffic can now be permitted/denied based on ICMP code. We introduced or modified the following commands: **access-list extended** , **service-object**, **service**.
Configuration session for editing ACLs and objects. Forward referencing of objects and ACLs in access rules.	9.3(2)	You can now edit ACLs and objects in an isolated configuration session. You can also forward reference objects and ACLs, that is, configure rules and access groups for objects or ACLs that do not yet exist. We introduced the **clear configuration session**, **clear session**, **configure session**, **forward-reference**, and **show configuration session** commands.
ACL support for Stream Control Transmission Protocol (SCTP)	9.5(2)	You can now create ACL rules using the **sctp** protocol, including port specifications. We modified the following command: **access-list extended** .
Ethertype rule support for the IEEE 802.2 Logical Link Control packet's Destination Service Access Point address.	9.6(2)	You can now write Ethertype access control rules for the IEEE 802.2 Logical Link Control packet's Destination Service Access Point address. Because of this addition, the **bpdu** keyword no longer matches the intended traffic. Rewrite **bpdu** rules for **dsap 0x42**. We modified the following commands: **access-list ethertype**
Support in routed mode for Ethertype rules on bridge group member interfaces and extended access rules on Bridge Group Virtual Interfaces (BVI).	9.7(1)	You can now create Ethertype ACLs and apply them to bridge group member interfaces in routed mode. You can also apply extended access rules to the Bridge Virtual Interface (BVI) in addition to the member interfaces. We modified the following commands: **access-group**, **access-list ethertype** .

Feature Name	Releases	Description
EtherType access control list changes.	9.9(1)	EtherType access control lists now support Ethernet II IPX (EII IPX). In addition, new keywords are added to the DSAP keyword to support common DSAP values: BPDU (0x42), IPX (0xE0), Raw IPX (0xFF), and ISIS (0xFE). Consequently, existing EtherType access control entries that use the BPDU or ISIS keywords will be converted automatically to use the DSAP specification, and rules for IPX will be converted to 3 rules (DSAP IPX, DSAP Raw IPX, and EII IPX). In addition, packet capture that uses IPX as an EtherType value has been deprecated, because IPX corresponds to 3 separate EtherTypes. We modified the following commands: **access-list ethertype** added the new keywords **eii-ipx** and **dsap** {**bpdu** | **ipx** | **isis** | **raw-ipx**}; **capture ethernet-type** no longer supports the **ipx** keyword.