# Object Groups for ACLs

The Object Groups for ACLs feature lets you classify users, devices, or protocols into groups and apply these groups to access control lists (ACLs) to create access control policies for these groups. This feature lets you use object groups instead of individual IP addresses, protocols, and ports, which are used in conventional ACLs. This feature allows multiple access control entries (ACEs). You can use each ACE to allow an entire group of users to access a group of servers or services or to deny them access; thereby reducing the size of an ACL and improving manageability.

This module describes object-group ACLs with zone-based policy firewalls and how to configure them for zone-based firewalls.



## Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

## Restrictions for Object Groups for ACLs

The following restrictions apply to the Object Groups for ACLs feature on zone-based firewalls:

- IPv6 is not supported.
- Dynamic and per-user access control lists (ACLs) are not supported.
- You cannot remove an object group or make an object group empty if it is used in an ACL.

- ACL statements using object groups will be ignored on packets that are sent to RP for processing.
- Object groups are supported only for IP extended ACLs.

# Information About Object Groups for ACLs

## Overview of Object Groups for ACLs

In large networks, the number of lines in an access control list (ACL) can be large (hundreds of lines) and difficult to configure and manage, especially if the ACLs frequently change. Object group-based ACLs are smaller, more readable, and easier to configure and manage. Object-group-based ACLs simplify static ACL deployments for large user access environments on Cisco IOS routers. The zone-based firewall benefits from object groups, because object groups simplify policy creation (for example, group A has access to group A services).

You can configure conventional access control entries (ACEs) and ACEs that refer to object groups in the same ACL. You can use object-group-based ACLs with quality of service (QoS) match criteria, zone-based policy firewall, Dynamic Host Configuration Protocol (DHCP), and any other features that use extended ACLs.

In addition, you can use object-group-based ACLs with multicast traffic. When there are many inbound and outbound packets, using object group-based ACLs increases performance compared to conventional ACLs. Also, in large configurations, this feature reduces the storage required in NVRAM, because you need not define an individual ACE for every address and protocol pairing.

## Integration of Zone-Based Firewalls with Object Groups

Zone-based firewalls use object-group access control lists (ACLs) to apply policies to specific traffic. You define an object-group ACL, associate it with a zone-based firewall policy, and apply the policy to a zone pair to inspect the traffic.

In Cisco IOS XE Release 3.12S, only expanded object-group ACLs are supported with firewalls.

The following features work with object groups that are configured on a firewall:

- Static and dynamic network address translation (NAT)
- Service NAT (NAT that supports non-standard FTP port numbers configured by the **ip nat service** command)
- FTP application layer gateway (ALG)
- Session Initiation Protocol (SIP) ALG

In a class map, you can configure a maximum of 64 matching statements using the **match access-group** command.

## Objects Allowed in Network Object Groups

A network object group is a group of any of the following objects:

- Any IP address—includes a range from 0.0.0.0 to 255.255.255.255 (This is specified using the **any** command.)

- Host IP addresses
- Hostnames
- Other network object groups
- Subnets

- Host IP addresses
- Network address of group members
- Nested object groups

# Objects Allowed in Service Object Groups

A service object group is a group of any of the following objects:

- Source and destination protocol ports (such as Telnet or Simple Network Management Protocol [SNMP])
- Internet Control Message Protocol (ICMP) types (such as echo, echo-reply, or host-unreachable)
- Top-level protocols (such as Encapsulating Security Payload [ESP], TCP, or UDP)
- Other service object groups

# ACLs Based on Object Groups

All features that use or reference conventional access control lists (ACLs) are compatible with object-group-based ACLs, and the feature interactions for conventional ACLs are the same with object-group-based ACLs. This feature extends the conventional ACLs to support object-group-based ACLs and also adds new keywords and the source and destination addresses and ports.

You can add, delete, or change objects in an object group membership list dynamically (without deleting and redefining the object group). Also, you can add, delete, or change objects in an object group membership list without redefining the ACL access control entry (ACE) that uses the object group. You can add objects to groups, delete them from groups, and then ensure that changes are correctly functioning within the object-group-based ACL without reapplying the ACL to the interface.

You can configure an object-group-based ACL multiple times with a source group only, a destination group only, or both source and destination groups.

You cannot delete an object group that is used within an ACL or a class-based policy language (CPL) policy.

# Guidelines for Object Group ACLs

- Object groups must have unique names. For example, to create a network object group named “Engineering” and a service object group named “Engineering,” you must add an identifier (or tag) to at least one object group name to make it unique. For example, you can use the names “Engineering-admins” and “Engineering-hosts” to make the object group names unique and to make it easier for identification.
- Additional objects can be added to an existing object group. After adding an object group, you can add more objects as required for the same group name. You do not need to re-enter existing objects; the previous configuration remains in place until the object group is removed.

- Different objects can be grouped together. For example, objects such as hosts, protocols, or services can be grouped together and configured under the same group name. Network objects can be defined only under a network group, and service objects can be defined only under a service group.
- When you define a group with the **object-group** command and use any security appliance command, the command applies to every item in that group. This feature can significantly reduce your configuration size.
- If an ACL that is associated with a class-map for ZBF inspections includes object-groups, when you add entries to or remove entries from the ACL, the changes take effect only after you exit the access-list configuration prompt.

# How to Configure Object Groups for ACLs

To configure object groups for ACLs, you first create one or more object groups. These can be any combination of network object groups (groups that contain objects such as, host addresses and network addresses) or service object groups (which use operators such as **lt**, **eq**, **gt**, **neq**, and **range** with port numbers). Then, you create access control entries (ACEs) that apply a policy (such as **permit** or **deny**) to those object groups.

## Creating a Network Object Group

A network object group that contains a single object (such as a single IP address, a hostname, another network object group, or a subnet) or multiple objects with a network object-group-based ACL to create access control policies for the objects.

Perform this task to create a network object group.

### SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **object-group network** *object-group-name*
4. **description** *description-text*
5. **host** {*host-address* | *host-name*}
6. *network-address* {*/nn* | *network-mask*}
7. **group-object** *nested-object-group-name*
8. Repeat the steps until you have specified objects on which you want to base your object group.
9. **end**

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Device> enable` | Enables privileged EXEC mode.<br>• Enter your password if prompted. |

| | Command or Action | Purpose |
|---|---|---|
| **Step 2** | **configure terminal**<br>**Example:**<br>`Device# configure terminal` | Enters global configuration mode. |
| **Step 3** | **object-group network** *object-group-name*<br>**Example:**<br>`Device(config)# object-group network my-network-object-group` | Defines the object group name and enters network object-group configuration mode. |
| **Step 4** | **description** *description-text*<br>**Example:**<br>`Device(config-network-group)# description test engineers` | (Optional) Specifies a description of the object group.<br>• You can use up to 200 characters. |
| **Step 5** | **host** {*host-address* \| *host-name*}<br>**Example:**<br>`Device(config-network-group)# host 209.165.200.237` | (Optional) Specifies the IP address or name of a host.<br>• If you specify a host address, you must use an IPv4 address. |
| **Step 6** | *network-address* {*/nn* \| *network-mask*}<br>**Example:**<br>`Device(config-network-group)# 209.165.200.225 255.255.255.224` | (Optional) Specifies a subnet object.<br>• You must specify an IPv4 address for the network address. The default network mask is 255.255.255.255. |
| **Step 7** | **group-object** *nested-object-group-name*<br>**Example:**<br>`Device(config-network-group)# group-object my-nested-object-group` | (Optional) Specifies a nested (child) object group to be included in the current (parent) object group.<br>• The type of child object group must match that of the parent (for example, if you are creating a network object group, you must specify another network object group as the child).<br>• You can use duplicated objects in an object group only via nesting of group objects. For example, if object 1 is in both group A and group B, you can define a group C that includes both A and B. However, you cannot include a group object that causes the group hierarchy to become circular (for example, you cannot include group A in group B and then also include group B in group A).<br>• You can use an unlimited number of levels of nested object groups (however, a maximum of two levels is recommended). |

| | Command or Action | Purpose |
|---|---|---|
| **Step 8** | Repeat the steps until you have specified objects on which you want to base your object group. | — |
| **Step 9** | **end**<br>**Example:**<br>`Device(config-network-group)# end` | Exits network object-group configuration mode and returns to privileged EXEC mode. |

# Creating a Service Object Group

Use a service object group to specify TCP and/or UDP ports or port ranges. When the service object group is associated with an access control list (ACL), this service object-group-based ACL can control access to ports.

### SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **object-group service** *object-group-name*
4. **description** *description-text*
5. *protocol*
6. {**tcp** | **udp** | **tcp-udp**} [**source** {{[**eq**] | **lt** | **gt**} *port1* | **range** *port1 port2*}] [{[**eq**] | **lt** | **gt**} *port1* | **range** *port1 port2*]
7. **icmp** *icmp-type*
8. **group-object** *nested-object-group-name*
9. Repeat the steps to specify the objects on which you want to base your object group.
10. **end**

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Device> enable` | Enables privileged EXEC mode.<br>• Enter your password if prompted. |
| **Step 2** | **configure terminal**<br>**Example:**<br>`Device# configure terminal` | Enters global configuration mode. |
| **Step 3** | **object-group service** *object-group-name*<br>**Example:**<br>`Device(config)# object-group service my-service-object-group` | Defines an object group name and enters service object-group configuration mode. |

| | Command or Action | Purpose |
|---|---|---|
| **Step 4** | **description** *description-text*<br>**Example:**<br>`Device(config-service-group)# description test engineers` | (Optional) Specifies a description of the object group.<br>• You can use up to 200 characters. |
| **Step 5** | *protocol*<br>**Example:**<br>`Device(config-service-group)# ahp` | (Optional) Specifies an IP protocol number or name. |
| **Step 6** | {**tcp** \| **udp** \| **tcp-udp**} [**source** {{[**eq**] \| **lt** \| **gt**} *port1* \| **range** *port1 port2*}] [{[**eq**] \| **lt** \| **gt**} *port1* \| **range** *port1 port2*]<br>**Example:**<br>`Device(config-service-group)# tcp-udp range 2000 2005` | (Optional) Specifies TCP, UDP, or both. |
| **Step 7** | **icmp** *icmp-type*<br>**Example:**<br>`Device(config-service-group)# icmp conversion-error` | (Optional) Specifies the decimal number or name of an Internet Control Message Protocol (ICMP) type. |
| **Step 8** | **group-object** *nested-object-group-name*<br>**Example:**<br>`Device(config-service-group)# group-object my-nested-object-group` | (Optional) Specifies a nested (child) object group to be included in the current (parent) object group.<br>• The type of child object group must match that of the parent (for example, if you are creating a network object group, you must specify another network object group as the child).<br>• You can use duplicated objects in an object group only via nesting of group objects. For example, if object 1 is in both group A and group B, you can define a group C that includes both A and B. However, you cannot include a group object that causes the group hierarchy to become circular (for example, you cannot include group A in group B and then also include group B in group A).<br>• You can use an unlimited number of levels of nested object groups (however, a maximum of two levels is recommended). |
| **Step 9** | Repeat the steps to specify the objects on which you want to base your object group. | — |
| **Step 10** | **end**<br>**Example:** | Exits service object-group configuration mode and returns to privileged EXEC mode. |

| | Command or Action | Purpose |
|---|---|---|
| | `Device(config-service-group)# end` | |

# Creating an Object-Group-Based ACL

When creating an object-group-based access control list (ACL), configure an ACL that references one or more object groups. As with conventional ACLs, you can associate the same access policy with one or more interfaces.

You can define multiple access control entries (ACEs) that reference object groups within the same object-group-based ACL. You can also reuse a specific object group in multiple ACEs.

Perform this task to create an object-group-based ACL.

### SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip access-list extended** *access-list-name*
4. **remark** *remark*
5. **deny** *protocol source* [*source-wildcard*] *destination* [*destination-wildcard*] [**option** *option-name*] [**precedence** *precedence*] [**tos** *tos*] [**established**] [**log** | **log-input**] [**time-range** *time-range-name*] [**fragments**]
6. **remark** *remark*
7. **permit** *protocol source* [*source-wildcard*] *destination* [*destination-wildcard*] [**option** *option-name*] [**precedence** *precedence*] [**tos** *tos*] [**established**] [**log** | **log-input**] [**time-range** *time-range-name*] [**fragments**]
8. Repeat the steps to specify the fields and values on which you want to base your access list.
9. **end**

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Device> enable` | Enables privileged EXEC mode.<br>• Enter your password if prompted. |
| **Step 2** | **configure terminal**<br>**Example:**<br>`Device# configure terminal` | Enters global configuration mode. |
| **Step 3** | **ip access-list extended** *access-list-name*<br>**Example:**<br>`Device(config)# ip access-list extended nomarketing` | Defines an extended IP access list using a name and enters extended access-list configuration mode. |

| | Command or Action | Purpose |
|---|---|---|
| **Step 4** | **remark** *remark*<br>**Example:**<br>`Device(config-ext-nacl)# remark protect server by denying access from the Marketing network` | (Optional) Adds a comment about the configured access list entry.<br>• A remark can precede or follow an access list entry.<br>• In this example, the remark reminds the network administrator that the subsequent entry denies the Marketing network access to the interface. |
| **Step 5** | **deny** *protocol source* [*source-wildcard*] *destination* [*destination-wildcard*] [**option** *option-name*] [**precedence** *precedence*] [**tos** *tos*] [**established**] [**log** \| **log-input**] [**time-range** *time-range-name*] [**fragments**]<br>**Example:**<br>`Device(config-ext-nacl)# deny ip 209.165.200.244 255.255.255.224 host 209.165.200.245 log`<br>`Example based on object-group:`<br>`Router(config)#object-group network my_network_object_group`<br>`Router(config-network-group)#209.165.200.224 255.255.255.224`<br>`Router(config-network-group)#exit`<br>`Router(config)#object-group network my_other_network_object_group`<br>`Router(config-network-group)#host 209.165.200.245`<br>`Router(config-network-group)#exit`<br>`Router(config)#ip access-list extended nomarketing`<br>`Router(config-ext-nacl)#deny ip object-group my_network_object_group object-group my_other_network_object_group log` | (Optional) Denies any packet that matches all conditions specified in the statement.<br>• Optionally use the **object-group** *service-object-group-name* keyword and argument as a substitute for the *protocol.* argument<br>• Optionally use the **object-group** *source-network-object-group-name* keyword and argument as a substitute for the *source source-wildcard.* arguments<br>• Optionally use the **object-group** *destination-network-object-group-name* keyword and argument as a substitute for the *destination destination-wildcard.* arguments<br>• If the *source-wildcard* or *destination-wildcard*is omitted, a wildcard mask of 0.0.0.0 is assumed, which matches all bits of the source or destination address, respectively.<br>• Optionally use the **any** keyword as a substitute for the *source source-wildcard* or *destination destination-wildcard* to specify the address and wildcard of 0.0.0.0 255.255.255.255.<br>• Optionally use the **host** *source* keyword and argument to indicate a source and source wildcard of *source* 0.0.0.0 or the **host** *destination* keyword and argument to indicate a destination and destination wildcard of *destination* 0.0.0.0.<br>• In this example, packets from all sources are denied access to the destination network 209.165.200.244. Logging messages about packets permitted or denied by the access list are sent to the facility configured by the **logging facility** command (for example, console, terminal, or syslog). That is, any packet that matches the access list will cause an informational logging message about the packet to be sent to the configured facility. The level of messages logged to the console is controlled by the **logging console** command. |

| | Command or Action | Purpose |
|---|---|---|
| | | • |
| **Step 6** | **remark** *remark*<br>**Example:**<br>`Device(config-ext-nacl)# remark allow TCP from any source to any destination` | (Optional) Adds a comment about the configured access list entry.<br>• A remark can precede or follow an access list entry. |
| **Step 7** | **permit** *protocol source* [*source-wildcard*] *destination* [*destination-wildcard*] [**option** *option-name*] [**precedence** *precedence*] [**tos** *tos*] [**established**] [**log** \| **log-input**] [**time-range** *time-range-name*] [**fragments**]<br>**Example:**<br>`Device(config-ext-nacl)# permit tcp any any` | Permits any packet that matches all conditions specified in the statement.<br>• Every access list needs at least one permit statement.<br>• Optionally use the **object-group** *service-object-group-name* keyword and argument as a substitute for the *protocol.*<br>• Optionally use the **object-group** *source-network-object-group-name* keyword and argument as a substitute for the *source source-wildcard.*<br>• Optionally use the **object-group** *destination-network-object-group-name* keyword and argument as a substitute for the *destination destination-wildcard.*<br>• If *source-wildcard* or *destination-wildcard* is omitted, a wildcard mask of 0.0.0.0 is assumed, which matches on all bits of the source or destination address, respectively.<br>• Optionally use the **any**keyword as a substitute for the *source source-wildcard* or *destination destination-wildcard* to specify the address and wildcard of 0.0.0.0 255.255.255.255.<br>• In this example, TCP packets are allowed from any source to any destination.<br>• Use the **log-input** keyword to include input interface, source MAC address, or virtual circuit in the logging output. |
| **Step 8** | Repeat the steps to specify the fields and values on which you want to base your access list. | Remember that all sources not specifically permitted are denied by an implicit **deny** statement at the end of the access list. |
| **Step 9** | **end**<br>**Example:**<br>`Device(config-ext-nacl)# end` | Exits extended access-list configuration mode and returns to privileged EXEC mode. |

# Configuring Class Maps and Policy Maps for Object Groups

## SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **class-map type inspect match-all** *class-map-name*
4. **match access-group name** *access-list-name*
5. **exit**
6. **policy-map type inspect** *policy-map-name*
7. **class type inspect** *class-map-name*
8. **pass**
9. **exit**
10. **class class-default**
11. **drop**
12. **end**

## DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Device> enable` | Enables privileged EXEC mode.<br>• Enter your password if prompted. |
| **Step 2** | **configure terminal**<br>**Example:**<br>`Device# configure terminal` | Enters global configuration mode. |
| **Step 3** | **class-map type inspect match-all** *class-map-name*<br>**Example:**<br>`Device(config)# class-map type inspect match-all ogacl-cmap` | Creates a Layer 3 and Layer 4 inspect type class map and enters the class-map configuration mode. |
| **Step 4** | **match access-group name** *access-list-name*<br>**Example:**<br>`Device(config-cmap)# match access-group name my-ogacl-policy` | Configures a match criterion for a class map on the basis of the specified ACL. |
| **Step 5** | **exit**<br>**Example:**<br>`Device(config-cmap)# exit` | Exits class-map configuration mode and returns to global configuration mode. |
| **Step 6** | **policy-map type inspect** *policy-map-name*<br>**Example:**<br>`Device(config)# policy-map type inspect ogacl-pmap` | Creates a inspect-type policy map and enters policy-map configuration mode. |

| | Command or Action | Purpose |
|---|---|---|
| **Step 7** | **class type inspect** *class-map-name*<br>**Example:**<br>`Device(config-pmap)# class type inspect ogacl-cmap` | Specifies the traffic class on which an action is to be performed and enters policy-map class configuration mode. |
| **Step 8** | **pass**<br>**Example:**<br>`Device(config-pmap-c)# pass` | Allows packets to be sent to a device without being inspected. |
| **Step 9** | **exit**<br>**Example:**<br>`Device(config-pmap-c)# exit` | Exits policy-map class configuration mode and returns to policy-map configuration mode. |
| **Step 10** | **class class-default**<br>**Example:**<br>`Device(config-pmap)# class class-default` | Specifies the default class to configure or modify a policy and enters policy-map class configuration mode. |
| **Step 11** | **drop**<br>**Example:**<br>`Device(config-pmap-c)# drop` | Drops packets that are sent to a device. |
| **Step 12** | **end**<br>**Example:**<br>`Device(config-pmap-c)# end` | Exits policy-map class configuration mode and returns to privileged EXEC mode. |

# Configuring Zones for Object Groups

### SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **zone security** *zone-name*
4. **exit**
5. **zone security** *zone-name*
6. **exit**
7. **interface** *type number*
8. **zone-member security** *zone-name*
9. **end**

### DETAILED STEPS

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:** | Enables privileged EXEC mode.<br>• Enter your password if prompted. |

| | Command or Action | Purpose |
|---|---|---|
| | `Device> enable` | |
| **Step 2** | **configure terminal**<br>**Example:**<br>`Device# configure terminal` | Enters global configuration mode. |
| **Step 3** | **zone security** *zone-name*<br>**Example:**<br>`Device(config)# zone security outside` | Creates a security zone and enters security zone configuration mode.<br>• You need two security zones to create a zone pair: a source zone and a destination zone |
| **Step 4** | **exit**<br>**Example:**<br>`Device(config-sec-zone)# exit` | Exits security zone configuration mode and returns to global configuration mode. |
| **Step 5** | **zone security** *zone-name*<br>**Example:**<br>`Device(config)# zone security inside` | Creates a security zone and enters security zone configuration mode.<br>• You need two security zones to create a zone pair: a source zone and a destination zone |
| **Step 6** | **exit**<br>**Example:**<br>`Device(config-sec-zone)# exit` | Exits security zone configuration mode and returns to global configuration mode. |
| **Step 7** | **interface** *type number*<br>**Example:**<br>`Device(config)# interface gigabitethernet 0/1/1` | Configures an interface and enters interface configuration mode. |
| **Step 8** | **zone-member security** *zone-name*<br>**Example:**<br>`Device(config-if)# zone-member security inside` | Attaches an interface to a security zone. |
| **Step 9** | **end**<br>**Example:**<br>`Device(config-if)# end` | Exits interface configuration mode and returns to global configuration mode. |

# Applying Policy Maps to Zone Pairs for Object Groups

### SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **zone-pair security** *zone-pair-name* **source** {*zone-name* | **default** | **self**} **destination** {*zone-name* | **default** | **self**}
4. **service-policy type inspect** *policy-map-name*

5. **end**

**DETAILED STEPS**

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Device> enable` | Enables privileged EXEC mode.<br>• Enter your password if prompted. |
| **Step 2** | **configure terminal**<br>**Example:**<br>`Device# configure terminal` | Enters global configuration mode. |
| **Step 3** | **zone-pair security** *zone-pair-name* **source** {*zone-name* \| **default** \| **self**} **destination** {*zone-name* \| **default** \| **self**}<br>**Example:**<br>`Device(config)# zone-pair security out-to-in source outside destination inside` | Creates a zone pair and enters security zone-pair configuration mode. |
| **Step 4** | **service-policy type inspect** *policy-map-name*<br>**Example:**<br>`Device(conf-sec-zone-pair)# service-policy type inspect ogacl-pmap` | Attaches a firewall policy map to a security zone pair. |
| **Step 5** | **end**<br>**Example:**<br>`Device(config-sec-zone-pair)# end` | Exits security zone-pair configuration mode and returns to global configuration mode. |

# Verifying Object Groups for ACLs

**SUMMARY STEPS**

1. **enable**
2. **show object-group** [*object-group-name*]
3. **show ip access-list** [*access-list-name*]

**DETAILED STEPS**

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **enable**<br>**Example:**<br>`Device> enable` | Enables privileged EXEC mode.<br>• Enter your password if prompted. |
| **Step 2** | **show object-group** [*object-group-name*]<br>**Example:** | Displays the configuration in the named or numbered object group (or in all object groups if no name is entered). |

| | Command or Action | Purpose |
|---|---|---|
| | `Device# show object-group my-object-group` | |
| **Step 3** | **show ip access-list** [*access-list-name*]<br>**Example:**<br>`Device# show ip access-list my-ogacl-policy` | Displays the contents of the named or numbered access list or object group-based ACL (or for all access lists and object group-based ACLs if no name is entered). |

# Configuration Examples for Object Groups for ACLs

## Example: Creating a Network Object Group

The following example shows how to create a network object group named my-network-object-group, which contains two hosts and a subnet as objects:

```
Device> enable
Device# configure terminal
Device(config)# object-group network my-network-object-group
Device(config-network-group)# description test engineers
Device(config-network-group)# host 209.165.200.237
Device(config-network-group)# host 209.165.200.238

Device(config-network-group)# 209.165.200.241 255.255.255.224
Device(config-network-group)# end
```

The following example shows how to create a network object group named my-company-network, which contains two hosts, a subnet, and an existing object group (child) named my-nested-object-group as objects:

```
Device> enable
Device# configure terminal
Device(config)# object-group network my-company-network
Device(config-network-group)# host host1
Device(config-network-group)# host 209.165.200.242
Device(config-network-group)# 209.165.200.225 255.255.255.224
Device(config-network-group)# group-object my-nested-object-group
Device(config-network-group)# end
```

## Example: Creating a Service Object Group

The following example shows how to create a service object group named my-service-object-group, which contains several ICMP, TCP, UDP, and TCP-UDP protocols and an existing object group named my-nested-object-group as objects:

```
Device> enable
Device# configure terminal
Device(config)# object-group service my-service-object-group
Device(config-service-group)# icmp echo
Device(config-service-group)# tcp smtp
```

```
Device(config-service-group)# tcp telnet
Device(config-service-group)# tcp source range 1 65535 telnet
Device(config-service-group)# tcp source 2000 ftp
Device(config-service-group)# udp domain
Device(config-service-group)# tcp-udp range 2000 2005
Device(config-service-group)# group-object my-nested-object-group
Device(config-service-group)# end
```

# Example: Creating an Object Group-Based ACL

The following example shows how to create an object-group-based ACL that permits packets from the users in my-network-object-group if the protocol ports match the ports specified in my-service-object-group:

```
Device> enable
Device# configure terminal
Device(config)# ip access-list extended my-ogacl-policy
Device(config-ext-nacl)# permit object-group my-service-object-group object-group
my-network-object-group any
Device(config-ext-nacl)# deny tcp any any
Device(config-ext-nacl)# end
```

# Example: Configuring Class Maps and Policy Maps for Object Groups

```
Device# configure terminal
Device(config)# class-map type inspect match-all ogacl-cmap
Device(config-cmap)# match access-group name my-ogacl-policy
Device(config-cmap)# exit
Device(config)# policy-map type inspect ogacl-pmap
Device(config-pmap)# class type inspect ogacl-cmap
Device(config-pmap-c)# pass
Device(config-pmap-c)# exit
Device(config-pmap)# class class-default
Device(config-pmap-c)# drop
Device(config-pmap-c)# end
```

# Example: Configuring Zones for Object Groups

```
Device# configure terminal
Device(config)# zone security outside
Device(config-sec-zone)# exit
Device(config)# zone security inside
Device(config-sec-zone)# exit
Device(config)# zone-pair security out-to-in source outside destination inside
Device(conf-sec-zone-pair)# exit
Device(config)# interface gigabitethernet 0/1/1
Device(config-if)# zone-member security inside
Device(config-if)# exit
Device(config)# interface gigabitethernet 0/1/0
Device(config-if)# zone-member security outside
Device(config-if)# end
```

# Example: Applying Policy Maps to Zone Pairs for Object Groups

```
Device# configure terminal
Device(config)# zone-pair security out-to-in source outside destination inside
Device(config-sec-zone-pair)# service-policy type inspect ogacl-pmap
Device(config-sec-zone-pair)# end
```

# Example: Verifying Object Groups for ACLs

The following example shows how to display all object groups:

```
Device# show object-group

Network object group auth-proxy-acl-deny-dest
 host 209.165.200.235
Service object group auth-proxy-acl-deny-services
 tcp eq www
 tcp eq 443
Network object group auth-proxy-acl-permit-dest
 209.165.200.226 255.255.255.224
 209.165.200.227 255.255.255.224
 209.165.200.228 255.255.255.224
 209.165.200.229 255.255.255.224
 209.165.200.246 255.255.255.224
 209.165.200.230 255.255.255.224
 209.165.200.231 255.255.255.224
 209.165.200.232 255.255.255.224
 209.165.200.233 255.255.255.224
 209.165.200.234 255.255.255.224
Service object group auth-proxy-acl-permit-services
 tcp eq www
 tcp eq 443
```

The following example shows how to display information about specific object-group-based ACLs:

```
Device# show ip access-list my-ogacl-policy

Extended IP access list my-ogacl-policy
10 permit object-group eng_service any any
```

# Additional References for Object Groups for ACLs

### Related Documents

| Related Topic | Document Title |
|---|---|
| Cisco IOS commands | Cisco IOS Master Command List, All Releases |

| Related Topic | Document Title |
|---|---|
| Security commands | • Cisco IOS Security Command Reference: Commands A to C<br>• Cisco IOS Security Command Reference: Commands D to L<br>• Cisco IOS Security Command Reference: Commands M to R<br>• Cisco IOS Security Command Reference: Commands S to Z |
| ACL configuration guide | *Security Configuration Guide: Access Control Lists* |

### Technical Assistance

| Description | Link |
|---|---|
| The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password. | http://www.cisco.com/cisco/web/support/index.html |

# Feature Information for Object Groups for ACLs

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

*Table 1: Feature Information for Object Groups for ACLs*

| Feature Name | Releases | Feature Information |
|---|---|---|
| Object Groups for ACLs | Cisco IOS XE Release 3.12S | The Object Groups for ACLs feature lets you classify users, devices, or protocols into groups and apply them to access control lists (ACLs) to create access control policies for those groups. This feature allows multiple access control entries (ACEs), but now you can use each ACE to allow an entire group of users to access a group of servers or services or to deny them from doing so. You can use object-group ACLs with zone-based firewalls.<br><br>The following commands were introduced or modified: **deny**, **ip access-group**, **ip access-list**, **object-group network**, **object-group service**, **permit**, **show ip access-list**, and **show object-group**. |