# Object-group ACLs

## Feature History for Object-group ACLs

This table provides release and platform support information for the features explained in this module.

These features are available in all the releases subsequent to the one they were introduced in, unless noted otherwise.

| Release | Feature Name and Description | Supported Platform |
|---|---|---|
| Cisco IOS XE 17.18.1 | Object-group ACLs:<br><br>Object-groups access control lists (OGACLs) enable network administrators to classify users, devices, or protocols into logical groups and apply these groups to ACLs. | Cisco C9350 Series Smart Switches<br><br>Cisco C9610 Series Smart Switches |

## Object-group ACLs

Object-groups access control lists (OGACLs) enable network administrators to classify users, devices, or protocols into logical groups and apply these groups to ACLs. By leveraging object groups, you can create and enforce access control policies based on groups rather than individual elements.

## Advantages of OGACLs

These are the advantages of OGACLs:

- Unlike traditional ACLs that rely on individual IP addresses, protocols, and ports, object groups allow for greater flexibility and scalability. Each access control entry (ACE) can now permit or deny access for an entire group of users to a group of servers or services, streamlining the rule set and management process.
- In extensive network environments, ACLs can become lengthy and challenging to manage, especially when changes are frequent. OGACLs are more concise, easier to read, and simpler to configure and update. This approach significantly reduces the complexity associated with both static and dynamic ACL deployments on networking devices.
- The use of object groups also enhances the Cisco IOS Firewall by simplifying policy creation. For example, you can easily configure rules so that "Group A has access to Group A services," improving efficiency and eliminating configuration errors.
- In environments with high volumes of inbound and outbound packets, OGACLs offer enhanced performance compared to conventional ACLs.
- In large-scale configurations, OGACLs reduce storage requirements in NVRAM by eliminating the need to define individual ACEs for each address and protocol pair, streamlining both configuration and resource usage.
- You can configure both conventional ACEs and ACEs that reference object groups within the same ACL. This flexibility allows seamless integration of object groups into existing ACL configurations.
- All features that use or reference conventional ACLs are compatible with OGACLs, and the feature interactions for conventional ACLs are the same with OGACLs. This feature extends the conventional ACLs to support OGACLs including the source and destination addresses and ports and any new keywords.

# Object Groups

An object group can contain a single object, such as an individual IP address, network, or subnet, or multiple objects, including combinations of several IP addresses, networks, or subnets.

## Advantages of Object Groups

These are the advantages of object groups:

- A typical ACE allows a group of users to access only a specific group of servers. With OGACLs, you can create a single ACE referencing an object group, instead of writing multiple ACEs for each unique IP address. Similarly, protocol port object groups can be used to grant access to specific sets of applications for designated user groups. ACEs can reference object groups for the source, the destination, both, or neither.
- Object groups enable separation of ownership for ACE components. For example, individual departments can manage their own group memberships, while a central administrator controls the ACE to define inter-departmental access.
- Object groups can also be utilized in features that use Cisco Policy Language (CPL) class maps, further extending their flexibility and usefulness.

# Types of Supported Object Groups

This feature supports two main types of object groups for grouping ACL parameters:

- Network Object Groups: Used to group IP addresses.
- Service Object Groups: Used to group protocols, protocol services (ports), and Internet Control Message Protocol (ICMP) types.

## 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
- Subnets
- Host IP addresses
- Network address of group members
- Nested object groups
- Other network 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

# Integration with Network Features

Object group-based ACLs can be used alongside various network features, including quality of service (QoS) match criteria, Cisco IOS Firewall, Dynamic Host Configuration Protocol (DHCP), and other functionalities that utilize extended ACLs. Additionally, these ACLs are compatible with multicast traffic management.

## Guidelines for OGACL

These guidelines are applicable to all switches:

- You can add, delete, or modify objects within an object group membership list without needing to delete or redefine the object group itself.
- You can update the membership list without having to redefine the ACL ACE that references the object group.
- You can configure OGACLs multiple times using only a source group, only a destination group, or both source and destination groups as needed.
- You cannot delete an object group if it is currently being used within an ACL or a CPL policy.
- You can use object groups only in extended named and numbered ACLs.
- Object group-based ACLs support only IPv4 or IPv6 addresses.
- Object group-based ACLs support only Layer 3 interfaces (such as routed interfaces and VLAN interfaces) and sub-interfaces.
- Object group-based ACLs are not supported with IPsec.
- The number of object group-based ACEs supported in an ACL varies depending on platform, subject to TCAM availability.

These guidelines are applicable only to Cisco C9610 Series Smart Switches:

- Object groups are only supported in extended ACLs.
- Object group-based ACLs support both IPv4 and IPv6 addresses.
- Object group-based ACLs are not supported on Layer 2 interfaces.
- Object group-based ACLs are not supported with IPsec.
- ACL statements using object groups will be ignored on packets that are sent to RP for processing. This guidelines is applicable for all switches except Cisco C9610 Series Smart Switches.
- Object group-based ACLs are supported only on ingress port. There is no support on egress direction.
- IPv6 object group-based ACLs with Log option are not supported. However, IPv4 object group-based ACLs with Log option is supported.
- IPv4 object group-based ACLs for multicast packet control are not supported.
- IPv6 object group-based ACLs for control packet are not supported.
- You cannot configure conventional ACEs and ACEs that refer to object groups in the same ACL.
- Per ACE statistics is supported only for Deny ACEs. Per ACE statistics for Permit ACE is not supported. If the same ACL is applied to multiple ports, then the deny counters are cumulative of all the ports on which the ACL is attached.

## How to configure OGACLs

To configure OGACLs, follow these steps:

### Procedure

**Step 1** 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)
- service object groups (which use operators such as lt, eq, gt, neq, and range with port numbers)

**Step 2** Configure access control entries (ACEs) that apply a policy (such as permit or deny) to those object groups.

## Create a Network Object Group

A network object group can include a single object, such as an individual IP address, hostname, another network object group, or subnet, or multiple objects. You can use a network object-group-based ACL to create access control policies for all the objects within the group.

To create a network object group, perform this task:

### Procedure

**Step 1** **enable**

**Example:**

```
Device> enable
```

Enables privileged EXEC mode.

Enter your password, if prompted.

**Step 2** **configure terminal**

**Example:**

```
Device# configure terminal
```

Enters global configuration mode.

**Step 3** **object-group network** *object-group-name*

**Example:**

```
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*

**Example:**

```
Device(config-network-group)# description test engineers
```

(Optional) Specifies a description of the object group.

You can use up to 200 characters.

**Step 5** **host** {*host-address* | *host-name*}

**Example:**

```
Device(config-network-group)# host 209.165.200.237
```

(Optional)Specifies the IP address or name of a host.

If you specify a host address, you must use an IPv4 address.

**Step 6** *network-address* {/*nn* | *network-mask*}

**Example:**

```
Device(config-network-group)# 209.165.200.225 255.255.255.224
```

(Optional) Specifies a subnet object.

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*

**Example:**

```
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.

- 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).
- 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).
- You can use an unlimited number of levels of nested object groups (however, a maximum of two levels is recommended).

**Step 8** Repeat the steps until you have specified objects on whichyou want to base your object group.

**Step 9** **end**

**Example:**

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

Exits network object-group configuration mode and returns to privileged EXEC mode.

## Create a Service Object Group

A service object group allows you to define specific TCP and/or UDP ports or port ranges. When linked to an access control list (ACL), a service object-group-based ACL can control access to the designated ports.

To create a service object group, perform this task.

### Procedure

**Step 1** **enable**

**Example:**

```
Device> enable
```

Enables privileged EXEC mode.

Enter your password, if prompted.

**Step 2** **configure terminal**

**Example:**

```
Device# configure terminal
```

Enters global configuration mode.

**Step 3** **object-group service** *object-group-name*

**Example:**

```
Device(config)# object-group service my-service-object-group
```

Defines an object group name and enters serviceobject-group configuration mode.

**Step 4** **description** *description-text*

**Example:**

```
Device(config-service-group)# description test engineers
```

(Optional) Specifies a description of the object group.

You can use up to 200 characters.

**Step 5** *protocol*

**Example:**

```
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*]

**Example:**

```
Device(config-service-group)# tcp-udp range 2000 2005
```

(Optional) Specifies TCP, UDP, or both

**Step 7** **icmp** *icmp-type*

**Example:**

```
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*

**Example:**

```
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.

- The type of child object group must match that of the parent (for example, if youare creating a network object group, you must specify another network object group as the child).
- 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 objectthat 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).
- 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 whichyou want to base your object group.

**Step 10** end

**Example:**

```
Device(config-service-group)# end
```

Exits service object-group configuration mode and returns to privileged EXEC mode.

## Create an Object-Group-Based ACL

When creating an object-group-based access control list (ACL), configure the ACL to reference one or more object groups. Similar to traditional ACLs, you can apply the same access policy to multiple interfaces.

Within a single object-group-based ACL, you can define multiple ACEs that reference object groups, and you can reuse the same object group in different ACEs as needed.

To create an object-group-based ACL, perform this task.

### Procedure

**Step 1** **enable**

**Example:**

```
Device> enable
```

Enables privileged EXEC mode.

Enter your password, if prompted.

**Step 2** **configure terminal**

**Example:**

```
Device# configure terminal
```

Enters global configuration mode.

**Step 3** **ip access-list extended** *access-list-name*

**Example:**

```
Device(config)# ip access-list extended nomarketing
```

Defines an extended IP access list using a name and enters extended access-list configuration mode.

**Step 4** **remark** *remark*

**Example:**

```
Device(config-ext-nacl)# remark protect server by denying access from the
Marketing network
```

(Optional) Adds a comment about the configured access list entry.

- A remark can precede or follow an access list entry.
- 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**]

**Example:**

```
Device(config-ext-nacl)# deny ip 209.165.200.244 255.255.255.224 host
209.165.200.245 log
```

Example based on object-group:

```
Device(config)# object-group network my_network_object_group
Device(config-network-group)# 209.165.200.224 255.255.255.224
Device(config-network-group)# exit
Device(config)# object-group network my_other_network_object_group
Device(config-network-group)# host 209.165.200.245
Device(config-network-group)# exit
Device(config)# ip access-list extended nomarketing
Device(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.

- Optionally use the **object-group** *service-object-group-name* keyword and argument as a substitute for the *protocol* argument.
- Optionally use the **object-group** *source-network-object-group-name* keyword and argument as a substitute for the **source** *source-wildcard* arguments.
- Optionally use the **object-group** *destination-network-object-group-name* keyword and argument as a substitute for the **destination** *destination-wildcard* arguments.
- 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.

- 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.
- 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.

  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 messagesl ogged to the console is controlledby the **logging console** command.

**Step 6** **remark** *remark*

**Example:**

```
Device(config-ext-nacl)# remark allow TCP from any source to any destination
```

(Optional) Adds a comment about the configured access list entry.

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**]

**Example:**

```
Device(config-ext-nacl)# permit tcp any any
```

Permits any packet that matches all conditions specified in the statement.

Every access list needs at least one permit statement.

- Optionally use the **object-group** *service-object-group-name* keyword and argument as a substitute for the protocol.
- Optionally use the **object-group** *source-network-object-group-name* keyword and argument as a substitute for the *source source-wildcard*.
- Optionally use the **object-group** *destination-network-object-group-name* keyword and argument as a substitute for the *destination destination-wildcard*.

  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.
- Optionally use the **any** keyword as a substitutefor the *source source-wildcard* or *destination destination-wildcard* to specify the address and wildcard of 0.0.0.0 255.255.255.255.

  In this example, TCP packets are allowed from any source to any destination.
- 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**

**Example:**

```
Device(config-ext-nacl)# end
```

Exits extended access-list configuration mode and returns to privileged EXEC mode.

## Apply an Object-Group-Based ACL to an Interface

An OGACL can be used to control traffic on any interface where it is applied. Use the **ip access-group** command to apply an object group-based ACL to an interface.

To apply an OGACL to an interface, perform this task.

### Procedure

**Step 1** **enable**

**Example:**

```
Device> enable
```

Enables privileged EXEC mode.

Enter your password, if prompted.

**Step 2** **configure terminal**

**Example:**

```
Device# configure terminal
```

Enters global configuration mode.

**Step 3** **interface** *type number*

**Example:**

```
Device(config)# interface vlan 100
```

Specifies the interface and enters interface configuration mode.

**Step 4** **ip access-group** {*access-list-name* | *access-list-number*} {**in** | **out**}

**Example:**

```
Device(config-if)# ip access-group my-ogacl-policy in
```

Applies the ACL to the interface and specifies whether to filter inbound or outbound packets.

**Step 5** **end**

**Example:**

```
Device(config-if)# end
```

Exits interface configuration mode and returns to privileged EXEC mode.

## Verify Object Groups for ACLs

| Command | Description |
|---|---|
| **show object-group [*object-group-name*]** | Displays the configuration in the named or numbered object group (or in all object groups if no name is entered). |
| **show ip access-list [*access-list-name*]** | 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 OGACLs

These sections provide configuration examples for OGACL.

### Example: Create 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: Create 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: Create 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: Verify 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
```