Technical Guide

Access Control Lists (ACLs)

Feature Overview and Configuration Guide

Introduction

This guide describes Access Control Lists (ACLs), and general ACL configuration information. For detailed command information and examples for ACL commands, see the following chapters in your product's Command Reference:

- IPv4 Hardware Access Control List (ACL) Commands
- IPv6 Hardware Access Control List (ACL) Commands
- IPv4 Software Access Control List (ACL) Commands
- IPv6 Software Access Control List (ACL) Commands

Hardware ACLs are applied directly to interfaces, or are used for Quality of Service (QoS) classifications. Software ACLs are applied to Routing and Multicasting.

Products and software version that apply to this guide

This guide applies to AlliedWare Plus™ products that support ACLs, running version **5.4.4** or later. However, support and implementation of ACLs varies between products, and feature support may change in later software versions. To see whether a product or version supports a particular feature or command, see the product's Command Reference for the relevant software version.

- The hardware ACL action **send-to-vlan-port** is supported from version 5.4.6-2.1 onwards.
- Per-VLAN ACLs are supported from version 5.4.6-2.1 onwards (on most switch series - see "Filtering traffic on VLANs (per-VLAN ACLs)" on page 17 for support details).
- Version 5.4.8-0.2 changes the number of rule entries used by per-VLAN ACLs on some switches - see "Filtering traffic on VLANs (per-VLAN ACLs)" on page 17.
- ACL groups are supported from version 5.5.0-1.1 onwards.
- The **show access-list counters** command is supported from version 5.5.1-2.1 onwards.
- You can use ACLs to drop unwanted packets without sending them to the CPU from version 5.5.3-0.1.
- Supports logging of packets denied by hardware ACLs from version 5.5.3-0.1.

Content

Overview

An Access Control List is one filter, or a sequence of filters, that are applied to an interface to either block or pass (or when using QoS, apply priority to) packets that match the filter definitions. ACLs are used to restrict network access by hosts and devices and to control network traffic.

An ACL contains an ordered list of filters. Each filter specifies either permit or deny and a set of conditions the packet must satisfy in order to match the filter. The meaning of permit or deny entries depends on the context in which the ACL is used - either on an inbound or an outbound interface.

When a packet is received on an interface, the switch compares fields in the packet against filters in the ACL to check whether the packet has permission to be forwarded, based on the filter properties. The comparison process stops as soon as the first match is found, and then the action of the ACL is applied. If no entries match, then, for the case of AlliedWare Plus hardware ACLs, the ACL ends in an implicit 'permit all else' clause. So, the unmatched packets are permitted.

Because filters in an ACL are applied sequentially and their action stops at the first match, it is very important that you apply the filters in the correct order. For example you might want to pass all traffic from VLAN 4 except for that arriving from two selected addresses A and B. Setting up a filter that first passes all traffic from VLAN 4 then denies traffic from addresses A and B will not filter out traffic from A and B if they are members VLAN 4. To ensure that the traffic from A and B is always blocked you should first apply the filter to block traffic from A and B, then apply the filter to allow all traffic from VLAN 4. You can assign sequence numbers to filters. See "ACL filter sequence numbers" on page 45 for more information.

ACL rules

- The source or destination address or the protocol of each packet being filtered are tested against the filters in the ACL, one condition at a time (for a permit or a deny filter).
- If a packet does not match a filter then the packet is checked against the next filter in the ACL.
- If a packet and a filter match, the subsequent filters in the ACL are not checked and the packet is permitted or denied as specified in the matched filter.
- The first filter that the packet matches determines whether the packet is permitted or denied. After the first match, no subsequent filters are considered.
- If the ACL denies the address or protocol then the software discards the packet.
- For hardware ACLs, if no filters match then the packet is forwarded.
- Checking stops after the first match, so the order of the filters in the ACL is critical. The same permit or deny filter specified in a different order could result in a packet being passed in one situation and denied in another situation.
- Multiple ACLs per interface, per protocol (i.e. IPv4 and IPv6), per direction are allowed.
- For inbound ACLs, a permit filter continues to process the packet after receiving it on an inbound interface, and a deny filter discards the packet.

Hardware and software ACL types

Access Control Lists used in AlliedWare Plus are separated into two different types, software ACLs and hardware ACLs. You can define both types as either named or numbered.

Note: The filtering principles applied to software ACLs (those in the range 1 to 2699) are different to those applied to hardware ACLs (those in the range 3000 to 4699).
- software ACLs will **deny** access unless **explicitly permitted** by an ACL action.
- hardware ACLs will **permit** access unless **explicitly denied** by an ACL action.

Numbered ACLs (for hardware and software ACLs)

Numbered ACLs are assigned an ACL number within the range 1 to 4699. ACL numbers are grouped into ranges, where each range denotes a specific functionality. The following table shows the number ranges and functionality that your switch supports.

Table 1: ACL numeric ranges and functionality

ACL NUMBER RANGE	FUNCTION
1 to 99	IP standard ACL[1]
100 to 199	IP extended ACL [1]
1300 to 1999	IP standard expanded ACL [1]
2000 to 2699	IP extended expanded ACL [1]
3000 to 3699	Hardware IP ACL
4000 to 4699	Hardware MAC ACL

[1] Software ACLs that use either the ranges 1-99, 100-199, 1300-1999, 2000-2699, or are named ACLs (that use the standard or extended keyword followed by a text string), are used in features such as SNMP, IGMP and OSPF.

Hardware ACLs

These ACL types are applied directly to an interface, or are used for QoS classifications.

Hardware ACLs use the following ranges:

- 3000-3699 for hardware IP ACLs
- 4000-4699 for hardware MAC ACLs
- named hardware IPv4 ACLs
- named hardware IPv6 ACLs

Software ACLs

You can name software ACL types using the standard or extended keyword, followed by a text string.

Software ACLs use the following ranges:

- 1-99 (IP standard ACL range)
- 100-199 (IP extended ACL range)
- 1300-1999 (IP standard expanded ACL range)
- 2000-2699 (IP extended expanded ACL range)
- named standard IPv4 ACLs
- named extended IPv4 ACLs
- named standard IPv6 ACLs
- named extended IPv6 ACLs

In AlliedWare Plus, software ACLs are not used directly for filtering packets that are being forwarded.

Rather, the software ACLs are used more for purposes like defining:

- ranges of address to which protocol parameters are applied
- ranges of address which are excluded from being operated on by protocols
- routes to be included/excluded in the operation of routing protocols

The types of features that make use of software ACLs for these purposes are SNMP, PIM, IGMP, OSPF, and BGP. Examples of where software ACLs are used include:

- Specifying a set of RIP routes to which a particular Administrative Distance should be applied:

```
awplus(config)# distance <1-255> ip/mask <access-list>
```

- Filtering which routes from the OSPF route table should be imported into the main IP route table:

```
awplus(config)# distribute-list <access-list> in
```

- Defining the addresses of management stations that can access a given SNMP community:

```
awplus(config)# access-list 66 permit 192.168.11.5
awplus(config)# snmp-server community example1rw rw 66
```

- Specifying the range of multicast groups for which a router is offering PIM RP candidacy:

```
awplus(config)# ip pim rp-candidate <interface>[priority <priority>|
interval <interval>| grouplist <grouplist>]
```

Defining hardware MAC ACLs

Hardware MAC ACLs are used to filter traffic based on specific source or destination MAC addresses contained within the data frames. They can be applied to ports in the form of access groups.

A MAC access list requires the following components:

- an ACL number in the range 4000-4699.
- an action, such as permit, or deny. See "Actions for hardware ACLs" on page 13
- a source MAC address. You can use the format, HHHH.HHHH.HHHH to filter on a specific MAC address (where H is a hexadecimal number), or you can filter on any source MAC address by entering the word “any”.
- a source MAC mask. This mask determines which portion of the source MAC address header will be compared with that found in the incoming packets. The mask is configured in the format HHHH.HHHH.HHHH where each H is a hexadecimal number. In practice each hex number will normally be either 0 (to represent a match) or F (to represent a don’t care condition). A mask is not required if the source address is specified as “any”.
- a destination MAC address. You can use the format, HHHH.HHHH.HHHH, to filter on a specific MAC address (where H is a hexadecimal number), or you can filter on any destination MAC address by entering the word “any”.
- a destination MAC mask. This mask determines which portion of the destination MAC address header will be compared with that found in the incoming packets. The mask is configured in the format <HHHH.HHHH.HHHH> where each H is a hexadecimal number. In practice each hex number will normally be either 0 (to represent a match) or F (to represent a don’t care condition). A mask is not required if the source address is specified as “any”.

Example To permit packets coming from a specific MAC address of 0030.841A.1234 and with any destination address:

```
awplus# configure terminal
awplus(config)# access-list 4000 permit 0030.841A.1234 0000.0000.0000 any
```

Defining hardware IP ACLs

Hardware IP ACLs are used to filter traffic based on specific source or destination IP addresses and/or Layer 4 parameters contained within the data frames. They can be applied to ports in the form of access groups.

An IP access list requires the following components:

- an ACL number in the range 3000-3699
- an action, see "Actions for hardware ACLs" on page 13
- a packet type:
  - **IP**: This matches any type of IP packet. A source and destination address must be specified, although they can be “any”. The source address matches packets coming from specified networking devices or hosts. The destination address matches packets going to specified networking devices or hosts.
  - **ICMP**: This matches ICMP packets. A source and destination address must be specified, although they can be “any”. An ICMP type can optionally be specified after the destination address.
  - **TCP**: This matches TCP packets. A source and destination address must be specified, although they can be “any”. After the source address, a source port can optionally be specified and after the destination address a destination port can optionally be specified. The port matching can be done using **eq** (equal to), **gt** (greater than), **lt** (less than), **ne** (not equal to), or **range** (for a range of ports, which requires a start port and an end port).
  - **UDP**: This matches UDP packets and has the same options as TCP.
  - **proto**: This allows any IP protocol type to be specified (e.g. 89 for OSPF). A source and destination address must be also specified, although they can be “any”.

For example:

- To match (and permit) any type of IP packet containing a destination address of 192.168.1.1:

```
awplus(config)# access-list 3000 permit ip any 192.168.1.1/32
```

- To match (and permit) an ICMP packet with a source address of 192.168.x.x and an ICMP code of 4:

```
awplus(config)# access-list 3001 permit icmp 192.168.0.0/16 any icmp-
type 4
```

- To match a TCP packet with a source address of 192.168.x.x, source port of 80 and a destination port from 100 to 150:

```
awplus(config)# access-list 3002 permit tcp 192.168.0.0/16 eq 80 any
range 100 150
```

- To match a UDP packet with a source address of 192.168.x.x, a destination address of 192.168.1.x, and a destination port greater than 1024:

```
awplus(config)# access-list 3003 permit udp 192.168.0.0/16
192.168.1.0/24 gt 1024
```

- To match (and permit) a UDP packet with a source address of 192.168.30.2/32 and a destination port of 5062:

```
awplus(config)# access-list 3002 permit udp any 192.168.30.2/32 eq 5062
```

- To match to any OSPF packet:

```
awplus(config)# access-list 3004 permit proto 89 any any
```

Note: An IP address mask can be specified using either of the following notations:

- A.B.C.D/M: This is the most common; e.g. 192.168.1.0/24
- A.B.C.D A.B.C.D: 192.168.1.1 0.0.0.0 is the same as 192.168.1.1/32 and 192.168.1.1 255.255.255.255 is the same as “any”

ACLs use reverse masking, also referred to as wildcard masking, to indicate to the switch whether to check or ignore corresponding IP address bits when comparing the address bits in an ACL filter to a packet being submitted to the ACL.

Reverse masking for IP address bits specify how the switch treats the corresponding address bits. A reverse mask is also called an inverted mask because a 1 and 0 mean the opposite of what they mean in a subnet or a network mask.

A reverse mask bit 0 means check the corresponding bit value.

A reverse mask bit 1 means ignore the corresponding bit value.

- host A.B.C.D: This is the same as A.B.C.D/32

Defining named hardware ACLs

A named sequential hardware ACL consists of a series of filter entries. The only limit on the number of filter entries that you can add to the ACL is on the number of entries that can fit into the hardware table - the software does not put any other lower limit on the number of entries.

Entries in the ACL can be from four different types:

1. IP protocol filter entry

This can match on any combination of the fields:

- IP protocol number, for example 1 for ICMP, 2 for IGMP, 50 for ESP, 89 for OSPF, etc. Or, you can simply specify “IP”, to match any IP protocol
- Source IP address—an individual IP address or a subnet
- Dest IP address—an individual IP address or a subnet
- Source MAC address—an individual MAC address or a range
- Dest MAC address—an individual MAC address or a range
- VLAN ID

2. MAC filter entry

This can match on any combination of the fields:

- Source MAC address—an individual MAC address or a range
- Dest MAC address—an individual MAC address or a range
- VLAN ID
- Inner VLAN ID

3. ICMP protocol filter entry

This can match on any combination of the fields:

- Source IP address—an individual IP address or a subnet
- Dest IP address—an individual IP address or a subnet
- ICMP type
- VLAN ID

4. **TCP/UDP protocol filter entry**

- Source IP address—an individual IP address or a subnet
- Dest IP address—an individual IP address or a subnet
- Source TCP/UDP port—either single port number or a range
- Dest TCP/UDP port—either single port number or a range
- VLAN ID

You can find the exact syntax of the commands to create these entries in your switch's Command Reference.

There is no restriction on the combinations of filter entry types that can exist together in the same ACL.

To define a Named IPv4 Hardware ACL

Step 1: **Create the ACL**

```
awplus(config)# access-list hardware hw_name
```

This will put you into Hardware ACL Configuration mode, with the prompt:

```
awplus(config-ip-hw-acl)#
```

Step 2: **Create the individual filter entries within the ACL**

```
awplus(config-ip-hw-acl)# permit ip 192.168.1.0 0.0.0.255 192.168.2.0
0.0.0.255
awplus(config-ip-hw-acl)# deny ip 192.168.1.0 0.0.0.255 any
```

and so on...

Defining hardware IPv6 ACLs

There are no numbered IPv6 hardware ACLs. The only form of IPv6 hardware ACLs available in AlliedWare Plus are **Named** IPv6 hardware ACLs.

To create an IPv6 hardware ACL

Step 1: **Create the ACL with the command**

```
awplus(config)# ipv6 access-list ipv6_hw_name
```

This puts you into IPv6 hardware configuration mode, with the prompt:

```
awplus(config-ipv6-hw-acl)#
```

Step 2: **Define the filters that comprise the content of the ACL**

```
awplus(config-ipv6-hw-acl)# permit ip 2001:db8::/64 2001:db9::/64
awplus(config-ipv6-hw-acl)# deny ip 2001:db8::/64 any
```

Actions for hardware ACLs

Depending on your switch family, the following actions are available for hardware ACLs:

Table 2: Hardware ACL parameter actions

PARAMETER	ACTION
deny	Discard the packet
permit	Allow the packet
copy-to-cpu	Send a copy of the packet to the CPU and forward it as well.
send-to-cpu	Send the packet to the CPU and do not forward it. Note that specifying this action could result in EPSR healthcheck messages and other control packets being dropped.
send-to-mirror	Send the packet to the mirror port so packets are not switched.
copy-to-mirror	Send a copy of the packet to the mirror port and forward it as well.
send-to-vlan-port vlan <***vid***> port <***port***>	Send the packet out the specified port, tagged with the specified VLAN. This option was introduced in 5.4.6-2.1.
deny-and-not-cpu	Drop the packet and make sure that it isn't sent to the switch's CPU. This option was introduced in 5.5.3-0.1 on some switches. Use it if you want to drop packets that AlliedWare Plus would normally send to the switch's CPU. See "Using ACLs to drop unwanted packets without sending them to the CPU" on page 23 for an example. Note that on some AlliedWare Plus switches, this action is unavailable because the action of **deny** already prevents packets from going to the CPU.

Attaching hardware ACLs to interfaces

A hardware ACL is attached directly to a switchport using the **access-group** command. For example, to permit traffic from **192.168.1.x**, but discard from 192.168.x.x:

```
awplus# configure terminal
awplus(config)# access-list 3000 permit ip 192.168.1.0/24 any
awplus(config)# access-list 3001 deny ip 192.168.0.0/24 any
awplus(config)# interface port1.0.1
awplus(config-if)# access-group 3000
awplus(config-if)# access-group 3001
```

Similarly, a named hardware ACL can applied to an interface with the commands:

```
awplus(config)# interface port1.0.1
awplus(config-if)# access-group <ACL-name>
```

Global ACLs - attaching hardware ACLs to all switch ports at once

On some switch models, you can attach a hardware ACL directly to all switchports using the **access-group** command. For example, to permit traffic from **192.168.1.x**, but discard from 192.168.x.x:

```
awplus#configure terminal
awplus(config)# access-list 3000 permit ip 192.168.1.0/24 any
awplus(config)# access-list 3001 deny ip 192.168.0.0/24 any
awplus(config)# access-group 3000
awplus(config)# access-group 3001
```

Seeing how many packets match hardware ACLs

From version 5.5.1-2.1 onwards, access-list counters let you see the number of packets that match one or all of your hardware ACLs. Every time a hardware ACL allows or drops a packet, its counter increments. This lets you check your ACL configuration. To display the number, use the command:

```
awplus#show access-list counters
```

You can optionally limit this to a single ACL. For example, to display the matches for the ACL named 'ACL-1', use the command:

```
awplus#show access-list counters ACL-1
```

The output of **show access-list counters** looks like this:

```
awplus#show access-list counters
Hardware ACL Packet Counters

ACL-1
Packet Hits: 17
ACL-2
Packet Hits: 0
ACL-3
Packet Hits: 1
```

Logging of packets denied by hardware ACLs

From version 5.5.3-0.1 onwards, you can turn on logging of packets that are denied by hardware IP ACLs, on some switches. Hardware IP ACLs are ACLs with an ID number from 3000 to 3699. If you turn on logging, the switch will produce a log message when it drops denied packets.

Caution: If you enable this logging, do not use an action of **copy-to-cpu** or **send-to-cpu** in any of the switch's ACLs. If you do, packets that match those ACLs will also be dropped.

The switch will send the log messages to:

- The terminal monitor, by default, as long as you enter the command **terminal monitor**. and
- If you change the severity of the messages to **warn**, to the buffered log, permanent log, and external log as well. The external log defines what logs will be sent to external syslog servers.

The log message format is:

```
DENY>  IN=<interface-name> OUT=<blank> MAC=<dest-mac-addr:src-mac-addr:eth-type>
SRC=<source-ip-address> DST=<dest-ip-address> LEN=<packet-length> TOS=<type-of-
service> PREC=<precedence> TTL=<time-to-live> ID=<packet-ID> PROTO=<L4-protocol>
SPT=<L4-source-port> DPT=<L4-dest-port> LEN=<L4-packet-length> MARK=<internal-
packet-marking>
```

For example, an example of a permanent log message is:

```
2023 March 03 07:04:37 local5.warning awplus ulogd[931]: DENY>  IN=vlan100 OUT=
MAC=e0:1a:ea:50:51:fc:00:00:11:11:22:22:08:00 SRC=192.168.1.100 DST=192.170.1.1
LEN=42 TOS=00 PREC=0x00 TTL=64 ID=0 PROTO=UDP SPT=63 DPT=63 LEN=22 MARK=0x20000000
```

Configuring logging of denied hardware ACLs

To turn on logging, do the following steps:

Step 1: **Enable logging and optionally change its settings**

Use the commands:

```
awplus#configure terminal
awplus(config)#access-list hardware-deny-log [warn|info] [rate <1-1000>]
[burst <1-1000>] [timeout <1-3600>]
```

Caution: Be careful when changing these settings. If your changes significantly increase the number of log messages produced, the switch's CPU may become busy.

In this command:

- **warn** means that the logs have a severity level of 'warning' and **info** means that the logs have a severity level of 'info'. The default is **info**. If you want the messages to go to the permanent log as well as the terminal, change this to **warn**.
- **rate** limits the number of log messages produced per hour. The switch will stop creating log messages if packets are denied at a higher rate than this specified rate. The default is 12 packets per hour. This means, for example, that if there are 60 denied packets in the first hour, then at most the first 12 of these packets will be logged. After that, logging will stop unless the rate drops below 12 an hour. (The actual number of packets logged depends on the number of flows the dropped packets come from; see the **burst** parameter).
- **burst** is the maximum initial number of packets to log for a given flow. A flow is a set of packets with the same source IP address, destination IP address, source port, and destination port. The default is 5. The burst number reduces by 1 each time a packet in the flow is logged. Then it increases by 1 again every time the rate drops below the limit set by the **rate** parameter, up to the **burst** number.
- **timeout** is the length of time it takes for a flow record to expire, in seconds. The switch considers that a flow has expired if it hasn't received any packets that match that flow for the **timeout** number of seconds. The default is 3600 seconds (1 hour). When a flow expires, its **burst** value is reset and logging may restart (depending on the current rate of denied packets).

Step 2: **Specify logging on the desired ACLs**

Logging is available with the following ACL commands:

```
access-list <3000-3699> deny ip <other-parameters> log
access-list <3000-3699> deny icmp <other-parameters> log
access-list <3000-3699> deny proto <other-parameters> log
access-list <3000-3699> deny tcp <other-parameters> log
access-list <3000-3699> deny udp <other-parameters> log
[<sequence-number>] deny ip <other-parameters> log
[<sequence-number>] deny ipv6 <other-parameters> log
[<sequence-number>] deny icmp <other-parameters> log
[<sequence-number>] deny proto <other-parameters> log
[<sequence-number>] deny tcp <other-parameters> log
[<sequence-number>] deny udp <other-parameters> log
```

The **icmp**, **proto**, **tcp** and **udp** parameters can be used for an IPv4 or IPv6 ACL.

Management ACLs

The Management ACLs feature restricts who is allowed remote access to your device using Telnet or SSH. This Management ACL is a simple security feature that binds an ACL (Access Control List) to the VTY's (Virtual Terminal Lines). This will allow or deny IP addresses included in the ACL to create a connection to your device.

The commands are:

- **vty access-class**
- **vty ipv6 access-class**

Both commands have a **no** variant.

To check the ACLs' setting run the **show running-config** command.

Examples:

vty access-class

Command syntax

```
vty access-class (<1-99>|<1300-1999>)
```

PARAMETER	DESCRIPTION
vty	Virtual terminal
access-class	Filter connections based on an IP standard access-list
<1-99>	IP standard access-list number
<1300-1999>	IP standard access-list number (expanded range)

Example To allow remote logins from IP 10.50.115.41 only, using a numbered IP standard ACL, use the commands:

```
awplus(config)# access-list 25 permit 10.50.115.41 0.0.0.0
awplus(config)# vty access-class 25
```

vty ipv6 access-class

Command syntax

```
vty ipv6 access-class <name>
```

PARAMETER	DESCRIPTION
vty	Virtual terminal
ipv6	Internet Protocol version (IPv6)
access-class	Filter connections based on an IPv6 standard access-list
word	IPv6 standard access-list name

Example To allow remote logins from IP 2001:5a7:1504:e080: :1 only, using a named standard IPv6 AC, use the commands:

```
awplus(config)# ipv6 access-list standard mgt-server permit
  2001:5a7:1504:e080::1/64 exact-match
awplus(config)# vty ipv6 access-class mgt-server
```

Filtering traffic on VLANs (per-VLAN ACLs)

This feature is supported on:

- SBx908, SBx8100, x230, x310, x510, x510DP, x510L, IX5, x610, x930, DC2552XS, IE200, IE300 and IE510 Series switches, running software version 5.4.6-2.1 and later

It is not supported on software versions 5.4.5-x.x, 5.4.6-0.x or 5.4.6-1.x.

Per-VLAN ACLs enables you to filter traffic as it ingresses VLANs, by attaching ACLs to VLANs. To do this, first create your ACLs, then apply the ACLs to a VLAN access-map, and then apply the map to the desired VLANs.

The following example shows the commands to use:

Example

To deny all packets on VLAN 48 and 49, use the following steps:

1. Create VLANs 48 and 49, if they do not already exist.

```
awplus(config)# vlan database
awplus(config-vlan)# vlan 48
awplus(config-vlan)# vlan 49
```

2. Create an ACL to block packets from any source MAC address to any destination MAC address.

```
awplus(config)# access-list 4000 deny any any
```

3. Create a VLAN access-map called (for example) ‘deny_all’ and match this ACL

```
awplus(config)# vlan access-map deny_all
awplus(config-vlan-access-map)# match access-group 4000
awplus(config-vlan-access-map)# exit
```

4. Apply this ACL by applying the access map to VLAN 48 and 49

```
awplus(config)# vlan filter deny_all vlan-list 48-49 input
```

Per-VLAN ACL rules will be applied to all ports on which the VLAN is active. This means they will be applied to all ports that are access ports in the VLAN, all trunk ports that allow packets tagged for the VLAN, and all trunk ports whose native VLAN is this VLAN.

Number of rules

Each VLAN ACL uses one ACL rule entry per VLAN it is filtering. For example, the following situation uses 2 entries:

- rule A is applied to VLAN1 and 3 ports are active in VLAN1, and
- rule B is applied to VLAN2 and 3 ports are active in VLAN2.

However, on FS980M, SBx908, and SBx8100 systems, the number of rules used changes for ports within a VLAN that also have port ACLs or QoS policy-maps attached. On such ports, each VLAN ACL filter uses an extra entry.

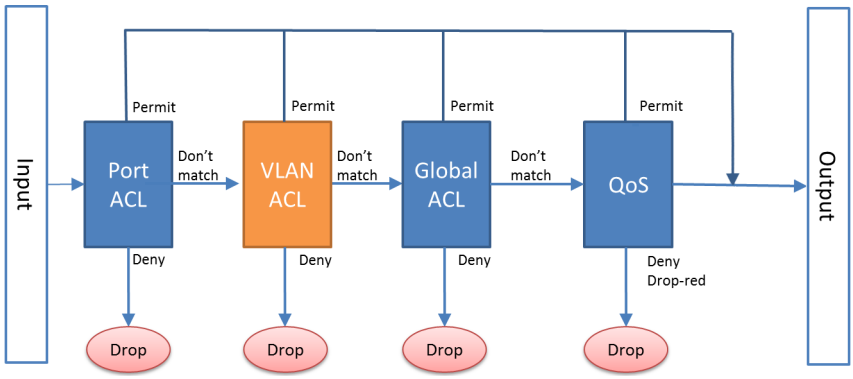
Also, on FS980M, SBx908, SBx8100 systems running version 5.4.7.x.x or earlier, each VLAN ACL uses one ACL rule entry per VLAN it is filtering, **per port it is applied to**. On those versions, the above example would use 6 rule entries.

The maximum number of ACL rules available depends on the switch model and port type. To see the available number of ACLs, use the **show platform classifier statistics utilization brief** command.

Rule precedence

SBx908 and SBx8100 switches

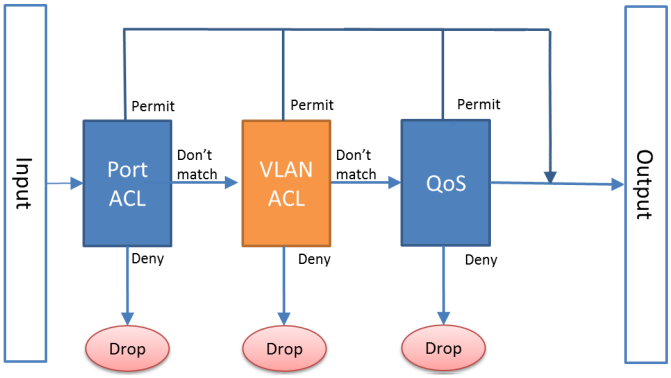
On SBx908 and SBx8100 switches, the switch checks packets against ACLs in the following order on ingress:



Once a packet matches an ACL (either a permit or deny match), the SBx908 or SBx8100 switch stops checking it against further ACLs.

IE200 Series switches

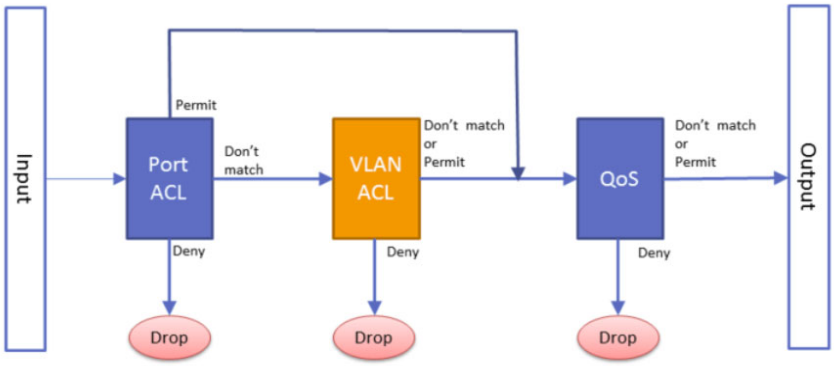
On IE200 Series switches, the switch checks packets against ACLs in the following order on ingress:



Once a packet matches an ACL (either a permit or deny match), the IE200 Series switch stops checking it against further ACLs.

Other switches

On other switches, the switch checks packets against ACLs in the following order on ingress:



On these switches, once an ACL denies a packet, the switch stops checking against further ACLs and drops the packet. If a port ACL permits the packet, the switch does not check VLAN ACLs. If either a port ACL or VLAN ACL permits the packet, the switch checks the packet against QoS ACLs.

Displaying information about per-VLAN ACLs

You can view details of the per-VLAN ACLs by using the following show commands:

- show vlan access-map
- show vlan filter

Output

Example output from **show vlan access-map:**

```
awplus#show vlan access-map

Vlan access map : deny_all
Hardware MAC access list 4000
   10 deny any any

Vlan access map : ip_range
Hardware IP access list 3000
   10 deny ip 192.168.1.1/24 any
```

Example output from **show vlan filter** for the access-map named **deny_all:**

```
awplus#show vlan filter deny_all
Vlan filter : deny_all
    direction : ingress
    vlan list : 48-49
    access map : deny_all
Hardware MAC access list 4000
   10 deny any any
```

ACL groups

ACL groups allow for smaller configuration files to achieve the same ACL configuration. By specifying a list of hosts or ports as a group, that group can be used in an ACL instead of having to specify an ACL for each host/port combination. In some cases, this can be a large saving in configuration.

ACL groups can be used when adding ACLs for multiple hosts that require the same filtering. For example, blocking three ports on four hosts requires twelve lines of ACLs. Using host groups and port groups, only one ACL needs to be entered in the configuration. The hardware entries are not changed; twelve entries are still used in the ACL table. ACL groups are supported in global ACLs on platforms that support global ACLs.

Note: ACL groups help you reduce the configuration that needs to be entered, but do not make more space available in hardware.

Note: ACL groups can only be used in named hardware ACLs.

How to configure ACL groups

You can configure ACL groups by the following steps:

1. Define one or more lists of hosts using the **acl-group** command. These hosts can have masks in the same way hosts specified in existing ACLs do.
2. Define one or more lists of ports, along with their operation (equal, not equal, greater than, less than).
3. Use the **host-group** or **port-group** command to specify the host or port.

Configuration examples

This is an example of how you could configure an ACL where two sources are required to be blocked on 8 different service ports (10, 20, 30, 40, 50, 60, 70, 80). As can be seen below, the hosts and ports are repeated a number of times.

```
awplus(config)#access-list hardware 3005_My_ACL
awplus(config-ip-hw-acl)# deny tcp 1.1.1.1/32 any eq 10
awplus(config-ip-hw-acl)# deny tcp 1.1.1.1/32 any eq 20
awplus(config-ip-hw-acl)# deny tcp 1.1.1.1/32 any eq 30
awplus(config-ip-hw-acl)# deny tcp 1.1.1.1/32 any eq 40
awplus(config-ip-hw-acl)# deny tcp 1.1.1.1/32 any eq 50
awplus(config-ip-hw-acl)# deny tcp 1.1.1.1/32 any eq 60
awplus(config-ip-hw-acl)# deny tcp 1.1.1.1/32 any eq 70
awplus(config-ip-hw-acl)# deny tcp 1.1.1.1/32 any eq 80
awplus(config-ip-hw-acl)# deny tcp 2.2.2.2/32 any eq 10
awplus(config-ip-hw-acl)# deny tcp 2.2.2.2/32 any eq 20
awplus(config-ip-hw-acl)# deny tcp 2.2.2.2/32 any eq 30
awplus(config-ip-hw-acl)# deny tcp 2.2.2.2/32 any eq 40
awplus(config-ip-hw-acl)# deny tcp 2.2.2.2/32 any eq 50
awplus(config-ip-hw-acl)# deny tcp 2.2.2.2/32 any eq 60
awplus(config-ip-hw-acl)# deny tcp 2.2.2.2/32 any eq 70
awplus(config-ip-hw-acl)# deny tcp 2.2.2.2/32 any eq 80
```

ACL groups can be used to prevent repetitive lines in the config, and instead generate this config internally. The same ACL can be created using ACL groups. A host ACL group and a port ACL group must be created first. A host ACL group is a set of hosts that will be applied to an ACL. A port ACL group is a list of service ports that will be applied to an ACL. First, create a host ACL group:

```
awplus(config)# acl-group ip address My_Host_ACL_Group
awplus(config-ip-host-group)# ip 1.1.1.1/32
awplus(config-ip-host-group)# ip 2.2.2.2/32
```

After the host group has been configured, create a port ACL group:

```
awplus(config)#acl-group ip port My_Port_ACL_Group
awplus(config-ip-port-group)# eq 10
awplus(config-ip-port-group)# eq 20
awplus(config-ip-port-group)# eq 30
awplus(config-ip-port-group)# eq 40
awplus(config-ip-port-group)# eq 50
awplus(config-ip-port-group)# eq 60
awplus(config-ip-port-group)# eq 70
awplus(config-ip-port-group)# eq 80
```

Once we have both a host ACL group and a port ACL group, we can create a new ACL using these groups:

```
awplus(config)# access-list hardware 3005_My_ACL
awplus(config-ip-hw-acl)# deny tcp host-group My_Host_ACL_Group any
port-group My_Port_ACL_Group
```

As can be seen from the example above, new hosts or services ports can be denied using the existing ACL and ACL groups. Adding a new host to be denied using the existing config would have required 8 more lines of config. With the new ACL groups, the same can be accomplished using only one line.

Once your ACL groups are configured, they can be seen in the config:

```
! Specify a list of hosts for which we want to block certain TCP
ports
acl-group ip address myhosts
 ip 192.0.2.0/24
 ip 198.51.100.1/32

! Specify the ports we wish to block
acl-group ip port tcp-ports
  eq 10
  eq 20

! Create the list of ACLs.
! This is equivalent to listing each host/port entry
individually
access-list hardware myblacklist
 deny tcp host-group myhosts port-group tcp-ports any

! Use the access list in the same way it has always been done
interface port1.0.1-1.0.24
  access-group myblacklist
```

This example shows a group of IPv4 hosts and a group of TCP ports. The resulting list is applied to all interfaces to block these host/port combinations. If extra hosts or ports are added to the groups, the ACLs in hardware are automatically updated.

Show commands

As ACL groups are simply a more compact way of specifying ACLs, all the standard ACL display commands will work for ACL groups as well. The ACL groups show in the running config, but can also be displayed directly with the following commands:

```
awplus#show acl-group ip address
IPv4 Host Groups:

Host Group: myhosts
 192.0.2.0/24
 198.51.100.1/32
```

```
awplus#show acl-group ip port
Port Groups:

Port Group: tcp-ports
 eq 10
 eq 20
```

Using ACLs to drop unwanted packets without sending them to the CPU

From version 5.5.3-0.1 onwards, some switches have an action on hardware ACLs to drop packets and make sure that they aren't sent to the switch's CPU. This may be useful if you want to drop packets that AlliedWare Plus would normally send to the switch's CPU.

For example, this action may be useful if you have AMF Security and it is registering MAC addresses because it receives loop detection frames (LDF) from other switches in your network. You can use this action to prevent AMF Security from seeing the LDF frames. To configure this, do the following steps on edge devices that have ports with **auth-mac enable** configured on them.

Step 1: Create an ACL that uses the action deny-and-not-cpu

```
awplus(config)# access-list 4000 deny-and-not-cpu any any
```

Step 2: Enable QoS

```
awplus(config)# mls qos enable
```

Step 3: Create a class-map that matches on that ACL and on the Ethernet format for LDF, which is 8899

```
awplus(config)# class-map LDF
awplus(config-cmap)# match access-group 4000
awplus(config-cmap)# match eth-format ethii-any protocol 8899
awplus(config-cmap)# exit
```

Step 4: Create a policy-map and attach the class-map to it

```
awplus(config)# policy-map LDFP
awplus(config-pmap)# class LDF
awplus(config-pmap)# exit
```

Step 5: Attach the policy-map to the ports

```
awplus(config)# interface port1.0.1
awplus(config-if)# service-policy input LDFP
```

Hardware ACLs and QoS classifications

Interface ACLs and QoS policies can both be attached to the same port. Where this is done, packets received on the port will be matched against the ACLs first.

The interface ACLs and QoS classifications are implemented by taking the first matching filter and applying the action defined for that filter. All subsequent matches in the table are then ignored. Thus, because ACLs are also matched first, if the matching ACL has a permit action, the packet is forwarded due to that rule's action and any subsequent QoS rules are bypassed.

You can also apply permit rules using QoS. For example, you might want to permit a source IP address of 192.168.1.x, but block everything else on 192.168.x.x. In this case you could create both the permit and deny rules using QoS. For more information on these applications see "Actions for hardware ACLs" on page 13.

QoS ACLs

When using ACLs though QoS, the same classification and action abilities are available, but QoS has some additional fields that it can match on, see "Expanding ACL match criteria with QoS" on page 26, and also provides the ability to perform metering, marking, and remarking on packets that match the filter definitions. The action used by a QoS class-map is determined by the ACL that is attached to it. If no ACL is attached, it uses the permit action. If an ACL is not required by the class-map (for example, only matching on the VLAN) and a deny action is required, a MAC ACL should be added with **any** for source address and **any** for destination address.

The following example creates a class-map with will deny all traffic on VLAN 2:

```
awplus(config)# access-list 4000 deny any any
awplus(config)# class-map cmap1
awplus(config-cmap)# match access-group 4000
awplus(config-cmap)# match vlan 2
```

The default class-map matches to all traffic and so cannot have any match or ACL commands applied to it. The action for this class-map is set via the **default-action** command and is **permit** by default. It can be changed to **deny** by using the following commands:

```
awplus(config)# policy-map pmap1
awplus(config-pmap)# default-action deny
```

Attaching hardware ACLs using QoS

The same functionality can be achieved using QoS, by attaching the ACL to a class-map, attaching the class-map to a policy-map and attaching the policy-map to a port:

Step 1: **Enable QoS on the switch**

```
awplus(config)# mls qos enable
```

Step 2: **Create access lists**

Create ACL **3000** to permit all packets from the 192.168.1 subnet:

```
awplus(config)# access-list 3000 permit ip 192.168.1.0/24 any
```

Create ACL **3001** to deny all packets from the 192.168.0 subnet:

```
awplus(config)# access-list 3001 deny ip 192.168.0.0/24 any
```

Step 3: **Attach access-groups to class-maps**

Attach ACL **3000** to the class-map **cmap1**:

```
awplus(config)# class-map cmap1
awplus(config-cmap)# match access-group 3000
awplus(config-cmap)# exit
```

Attach ACL **3001** to another class-map (**cmap2**)

```
awplus(config)# class-map cmap2
awplus(config-cmap)# match access-group 3001
awplus(config-cmap)# exit
```

Step 4: **Attach class-maps to policy-maps**

Attach the class-map **cmap1** to policy-map **pmap1**:

```
awplus(config)# policy-map pmap1
awplus(config-pmap)# class cmap1
awplus(config-pmap-c)# exit
```

Add the class-map **cmap2** to the policy-map **pmap1**:

```
awplus(config-pmap)# class cmap2
awplus(config-pmap-c)# exit
```

Return to Global Configuration mode:

```
awplus(config-pmap)# exit
```

Step 5: **Add policy-maps to ports**

Add policy-map **pmap1** to **port1.0.1**

```
awplus(config)# interface port1.0.1
awplus(config-if)# service-policy input pmap1
```

Only one ACL can be attached to a class-map, but multiple class-maps can be attached to a policy-map. Interface ACLs can be attached to the same port as a QoS policy, with the interface ACLs being matched first as described at the beginning of the section about "Hardware ACLs and QoS classifications" on page 24.

Expanding ACL match criteria with QoS

Another reason for using QoS rather than interface ACLs is that QoS provides a lot more fields on which to match. These are accessed through the match commands in **config-cmap mode**.

Config-cmap mode describes the fields that can be matched on. Only one of each type can be matched, with the exception of tcp-flags (see below for classification). If multiple matches are specified, they are ANDed together. The following example shows how you can match a packet on VLAN 2, that has a source IP address of 192.168.x.x and a DSCP of 12:

Step 1: **Create ACL *3000* to permit all packets from the 192.168 subnet:**

```
awplus# configure terminal
awplus(config)# access-list 3000 permit ip 192.168.0.0/16 any
```

Step 2: **Apply ACL 3000 to the class-map cmap1, add the matching criteria of VLAN 2 and DSCP 12:**

```
awplus(config)# class-map cmap1
awplus(config-cmap)# match access-group 3000
awplus(config-cmap)# match vlan 2
awplus(config-cmap)# match dscp 12
awplus(config-cmap)# exit
```

Using QoS match commands with TCP flags

Usually, if multiple matches of the same type are specified, the matching process will apply to the last match that you specified. For TCP flags however, the arguments are ANDed together. For example, the following series of commands will match on a packet that has **ack**, **syn**, and **fin** set:

```
awplus# configure terminal
awplus(config)# class-map cmap1
awplus(config-cmap)# match tcp-flags ack
awplus(config-cmap)# match tcp-flags syn
awplus(config-cmap)# match tcp-flags fin
awplus(config-cmap)# exit
```

The following commands will achieve the same result:

```
awplus# configure terminal
awplus(config)# class-map cmap1
awplus(config-cmap)# match tcp-flags ack syn fin
awplus(config-cmap)# exit
```

Note that the matching is looking to see whether “any” of the specified flags are **set**. There is no checking for whether any of these flags are unset. Therefore the following commands will match on a packet in any of the following combinations of syn and ack status flags as shown in the following table:

```
awplus# configure terminal
awplus(config)# class-map cmap1
awplus(config-cmap)# match tcp-flags syn
awplus(config-cmap)# exit
```

SYN	ACK	MATCH ON PACKET
Set	Set	Yes
Set	Unset	Yes
Unset	Set	No
Unset	Unset	No

If you want to drop packets with syn only, but not with ack and syn, the following two class-maps can be used (note that ACL 4000 is used to apply a drop action as described in "Actions for hardware ACLs" on page 13):

Step 1: **Create access lists**

Create ACL **4000** to deny all packets with **any** source or destination address:

```
awplus#configure terminal
awplus(config)#access-list 4000 deny any any
```

Step 2: **Create class-maps**

Create the class-map **cmap1** and configure it to match on the TCP flags, **ack** and **syn**:

```
awplus(config)#class-map cmap1
awplus(config-cmap)# match tcp-flags ack syn
awplus(config-cmap)# exit
```

Create the class-map **cmap2** and configure it to match on the TCP flag, **syn**:

```
awplus(config)# class-map cmap2
awplus(config-cmap)# match tcp-flags syn
```

Step 3: **Apply access-groups to class-maps**

Apply ACL **4000** to this class-map (i.e. to **cmap2**):

```
awplus(config-cmap)# match access-group 4000
awplus(config-cmap)# exit
```

Step 4: **Create policy-maps**

Create the policy-map pmap1 and associate it with **cmap1**:

```
awplus(config)# policy-map pmap1
awplus(config-pmap)# class cmap1
awplus(config-pmap-c)# exit
```

Step 5: **Associate class-maps with policy-maps**

Associate **cmap2** with this policy-map (**pmap1**)

```
awplus(config-pmap)# class cmap2
awplus(config-pmap-c)# exit
```

Profile limitations on SBx908 and x900 Series switches

On SBx908 and x900 Series switches, a profile is a mask that comprises 16 bytes. Each filter item that is added to the ACL set will consume a portion of the 16 bytes. Note that Hardware ACLs and QoS filters both share this single mask. However each time a mask component is defined within the mask, it can by used in many ACLs - so it is the **number** of different components that is important.

The following table will help you manage your ACL mask:

Table 3: ACL mask components

PROTOCOL COMPONENT	BYTES USED IN THE MASK
Protocol Type	2
Ethernet format	2
IP protocol type (e.g. TCP, UDP)	1
Source IPv4 address[1]	0
Destination IPv4 address[1]	0
Source IPv6 address[1]	0
Destination IPv6 address	10
IPv6 Next Header	1
TCP port number	2
UDP port number	2
MAC source address	6
MAC destination address	6
Inner Vlan ID (includes two bytes TPID)	4
Inner CoS (includes two bytes TPID)	3
Tag Protocol Identifier (TPID)	2
Inner Tag Protocol Identifier (TPID)	2
IP precedence value	1

[1] Most classification fields use some of the 16 ‘profile bytes’ that are available to classify incoming packets. For example, an ACL with both source and destination MAC address fields set will consume 12 of the 16 bytes. However IPv6 source address, and IPv4 source and destination address, will not consume any bytes, allowing greater freedom in ACL configuration options.

Hardware filter example

If you make a hardware filter that matches on destination IP address and source TCP port, this adds 3 bytes to the mask:

- 1 byte for the IP protocol field (to indicate TCP)
- 2 bytes for the source TCP port number.

If you now create the following (additional) hardware filters:

- A hardware filter that matches on source MAC address: this adds 6 more bytes to the mask.
- A QoS class map that matches on destination IP address and DSCP (1 byte): this adds 1 more byte to the mask, for the DSCP.
- A hardware filter that matches on source IP address and source TCP port: this does not change the mask, because the switch already matches on source TCP port, and source IP address does not use any bytes.
- A hardware filter that matches on source UDP port: this does not add any length to the mask, because it shares the same 2 bytes as the source TCP port. However, if you next make a hardware filter that matches on destination TCP or UDP port, that uses another 2 bytes.

Are there now enough bytes for your set of filters?

The mask has a maximum size of **16 bytes**. When it reaches the 16-byte limit, no more classifiers can be used that would cause the mask to increase in size. The switch can still accept classifiers that use fields that have already been included in the mask.

There is no particular number of hardware filters or QoS flow groups that will cause the mask to reach its 16-byte limit - it could happen after a few filters, or you might be able to create hundreds of filters without the mask reaching its limit.

To determine whether you will have enough filter length, look at the fields you want to filter, determine the number of bytes for each field, and sum up the total number of bytes. If that number is less than 16, there is enough filter length. Don’t forget to count TCP and UDP source port as a single field, and likewise to count TCP and UDP destination port as a single field.

Maximum number of hardware ACLs

The maximum number of individual hardware ACLs depends on the hardware type of the port you apply the ACLs to, and the setting of the **platform hwfilter-size** command.
On SBx908 and x900 Series switches (but not for the SBx908 GEN2 switch), two hardware types exist: **Hardware Version 1** and **Hardware Version 2.** The following table shows the hardware modules and the version to which each belongs:

Table 4: Hardware types and modules

HARDWARE VERSION 1	HARDWARE VERSION 2
Baseboard	XEM-12Sv2
XEM-12T	XEM-12Tv2
XEM-12S	XEM-2XP
XEM-1XP	XEM-2XS
	XEM-2XT
	XEM-24T

The following table shows the maximum number of filters for each hardware type and **platform hwfilter-size** command setting.

Table 5: Hardware types and filters

HARDWARE TYPE	SETTING OF PLATFORM HWFILTER-SIZE	MAXIMUM
Hardware Type 1	ipv4-full-ipv6 (Default. Can filter on MAC addresses and IPv4 and IPv6 settings)	1024
Hardware Type 1	basic (Can only filter on MAC addresses and IPv4 settings)	2048
Hardware Type 2	ipv4-full-ipv6	4096
Hardware Type 2	basic	8192

Viewing the number of hardware ACLs and bytes used

The number of hardware ACLs and bytes used are displayed in the output from the **show platform classifier statistics utilization brief** command, as shown in Figure 1.

Figure 1: Example output from **show platform classifier statistics utilization brief**

```
[Instance 0]
  [port1.0.1-port1.0.12]
  Number of PCE Entries:

  Note: Total available rules depends on routingratio setting
                 Used / Total
  --------------------------------
  MC V6             128 ('ipv6 multicast-routing' uses this)
  System              0
  DHCP Snooping       0
  Web Auth            0
  Loop Detection      0
  EPSR                0
  SNAP                0
          0
  ACL                 0
  QoS                 0
  RA Guard            0
  MLD Snooping        0
  Total             128 / 1024 (12.50%)

  UDB Usage:
  Legend of Offset Type) 1:Ether 2:IP 3:TCP/UDP
  UDB Set        Offset Type      Used / Total
  -------------- 0------8------15 ------------
  IPv4_TCP       0000000000000000    0 / 16
  IPv4_UDP       0000000000000000    0 / 16
  MPLS           0000000000000000    0 / 16
  IPv4_Frag      0000000000000000    0 / 16
  IPv4           0000000000000000    0 / 16
  Ethernet       0000000000000000    0 / 16
  IPv6           0000000000000000    0 / 16
...
```

Filter limitations on SBx8100 Series switches

The number of ACLs you can create on SBx8100 Series switches depends on the number of filters available for the card that the egress port is on and the protocol component you want to filter on.

This section first lists the number of filters and then discusses the protocol components. Then it gives examples of rule and user-defined byte usage for different ACLs.

Number of filters per card

The following table lists the maximum number of filters that can be created per line card:

Table 6: The maximum number of filters

LINE CARD	MAXIMUM FILTER NUMBER
SBx81GP24	1536
SBx81GT24	1536
SBx81GT40	1536
SBx81XS16	1536
SBx81CFC960	1536
SBx81XLEM	5120
SBx81GS24a	7168
SBx81XS6	7168

Number of filters created per ACL or class-map

Typically, the switch creates one filter in hardware for each ACL or class-map. The exceptions are when:

- you filter by TCP and UDP port ranges by using the **lt**, **gt**, **ne**, and **range** parameters of the **access-list** command. In this case, the switch might create multiple filters, in order to cover the specified range of port numbers.
- an ACL is neither IPv6 nor non-IPv6 specific, because it only contains Layer 2 protocol components. In that case, two filters are added to hardware, one for IPv6 and another for non-IPv6. An example of this is the hardware MAC numbered option in the **access-list** command, which only matches on MAC address.

  If an ACL's parameters enable the switch to calculate that a match will be only be seen in an IPv6 packet, then the switch will only create an IPv6 filter in hardware.
  Similarly, if an ACL's parameters enable the switch to calculate that a match will **not** be seen in an IPv6 packet, then the switch will only create a non-IPv6 filter.

Protocol components

On SBx8100 Series switches, a filter is comprised of standard and optional protocol components. The standard protocol components are:

- Source and destination IPv4 addresses
- Source IPv6 address
- The first byte of the destination IPv6 address (matching on more than the first byte uses user-defined bytes from the IPv6 L2 UDB set - see below).
- MAC source and destination addresses

The optional protocol components are:

- At layer 4: ICMP packet type and ICMPv6 packet type
- At layer 3: Destination IPv6 address (excluding the first byte)
- At layer 2: IP precedence value, Tag Protocol Identifier (TPID), inner VLAN ID, inner CoS, inner TPID, and SNAP tagged and untagged packets

For standard protocol components, you are only limited by the total number of filters available. However, there are restrictions on the number of optional protocol components you can use.

Each optional protocol component consumes bytes from a shared pool of bytes, which are called user-defined bytes (UDBs). UDBs are combined into “UDB sets”, which apply to either IPv6 traffic or non-IPv6 traffic. Each UDB set contains a limited number of user-defined bytes - 14 for SBx81XLEM and 6 for other cards. You must design your ACLs so that they stay within that 6-byte or 14-byte limit.

Table 7 lists the optional protocol components, the number of bytes used by each protocol component, and the UDB sets that each protocol component applies to. This table shows, for example, that filtering on destination IPv6 address uses up to 6 (or 14) bytes from the IPv6 UDB sets and no bytes from the non-IPv6 UDB sets. Therefore, filtering on destination IPv6 address does not prevent you from filtering on other protocol components, so long as you choose other protocol components that only consume bytes from the non-IPv6 UDB sets.

Table 7: The protocol components and the number of bytes they consume

PROTOCOL COMPONENT	NUMBER OF UDB	UDB SETS IT CAN CONSUME
IP precedence value	1 byte	Both IPv6 and non-IPv6
TPID	2 bytes	Both IPv6 and non-IPv6
inner VLAN ID	2 bytes	Both IPv6 and non-IPv6
inner CoS	1 byte	Both IPv6 and non-IPv6
inner TPID	2 bytes	Both IPv6 and non-IPv6
SNAP tagged and untagged packets	2 bytes for SBx81GS24a & SBx81XS6 3 bytes for all other cards	Non-IPv6 sets only

Table 7: The protocol components and the number of bytes they consume (continued)

PROTOCOL COMPONENT	NUMBER OF UDB	UDB SETS IT CAN CONSUME
Destination IPv6 address	Up to the limit of the IPv6 UDB set, therefore up to 6 or 14 bytes depending on the card. You can specify the number of UDBs by specifying the prefix or creating a mask. E.g., to match on three bytes, set the prefix length to 24 (3 x 8 bits). This will use the standard protocol component and two UDBs.	IPv6 sets only
ICMP packet type	1 byte	Non-IPv6 sets only
ICMPv6 packet type	1 byte	IPv6 sets only

In reality, the IPv6 and non-IPv6 UDB sets each consist of multiple individual sets, as shown in Table 8. The names of these individual sets vary depending on the line card; Table 8 shows all possible names. Having multiple individual sets has no practical effect on the number of protocol components you can filter on, because all IPv6 UDB sets share a single pool of 6 (or 14) bytes, and all non-IPv6 UDB sets share a single pool of 6 (or 14) bytes.

Table 8: The individual UDB set names

INDIVIDUAL UDB SET NAME	IPV6 OR NON-IPV6
IPv6	IPv6
IPv6 L2	IPv6
IPv6 TCP	IPv6
IPv6 UDP	IPv6
Non-IPv6	Non-IPv6
IPv4	Non-IPv6
IPv4 Frag	Non-IPv6
IPv4 TCP	Non-IPv6
IPv4 UDP	Non-IPv6
MPLS	Non-IPv6
Ethernet	Non-IPv6
User-def	Non-IPv6

After you configure ACLs, you can see how many bytes are still available in each UDB set, by using the **show platform classifier statistics utilization brief** command. Figure 2 shows an example of the output for an ACL that is in a QoS class-map that matches on inner VLAN ID. Because it is impossible to know whether the inner VLAN ID will be seen in an IPv6 packet or a non-IPv6 packet, two filters have been generated and installed into the hardware to match against IPv6 and non-IPv6 packets. These filters have each consumed 2 bytes from the IPv6 and non-IPv6 UDB sets respectively.

Figure 2: Example output from **show platform classifier statistics utilization brief**

```
awplus#show platform classifier statistics utilization brief

Card 1.1:

[Instance 1]
  [port1.1.1-port1.1.16]                  Used / Total
  --------------------------------
  System              0
  MLD Snooping        0
  DHCP Snooping       0
  Loop Detection      0
  EPSR                0
           0
  ACL                 0
  QoS                 2
  RA Guard            0
  Total               2 / 1536 (0.13%)

  UDB Usage:
  Legend of Offset Type) 1:Ether 2:IP 3:TCP/UDP
  UDB Set        Offset Type      Used / Total
  -------------- 0------8------15 ------------
  IPv4 TCP       110000              2 /  6
  IPv4 UDP       110000              2 /  6
  MPLS           110000              2 /  6
  IPv4 Frag      110000              2 /  6
  IPv4           110000              2 /  6
  Ethernet       110000              2 /  6
  User-Def       110000              2 /  6
  IPv6 L2        110000              2 /  6
```

In the above output, the following fields describe the UDB usage:

Table 9: UDB usage information in **show platform classifier statistics utilization brief**

FIELD	MEANING
UDB Set	Collections of UDBs that apply to different types of traffic. For debugging purposes, this command output displays the number of bytes used from each individual UDB set, rather than the combined IPv6/non-IPv6 sets; see Table 8.
Offset type	The offset type describes where in the packet the UDB byte can be extracted from. The number 1 indicates Layer 2, 2 indicates Layer 3, and 3 indicates Layer 4. See Table 10 to see which offset type applies for each protocol component.
Used	The number of bytes that are currently used in each UDB set.
Total	The total number of UDBs available in each UDB set.

Table 10: The offset type for each protocol component

PROTOCOL COMPONENT	OFFSET TYPE
IP precedence value	Layer 2 (offset type = 1)
Tag Protocol Identifier (TPID)	Layer 2
inner VLAN ID	Layer 2
inner CoS	Layer 2
inner Tag Protocol Identifier (TPID)	Layer 2
SNAP tagged and untagged packets	Layer 2

Table 10: The offset type for each protocol component

PROTOCOL COMPONENT	OFFSET TYPE
Destination IPv6 address	Layer 3 (offset type = 2)
ICMP packet type	Layer 4 (offset type = 3)
ICMPv6 packet type	Layer 4

If an ACL requires more UDBs than are available, the switch will display the following error:

```
awplus(config-if)#ipv6 traffic-filter dest
% Insufficient space in the hardware packet classifier tables. The
total number of rules, bytes to match, or bytes to set has reached
the limit.
% Failed to attach ACL dest to port1.1.1
```

Note that the UDBs are consumed by the first use of a protocol component in a filter, but using the same protocol component in further filters does not consume extra UDBs. You can use a protocol component in additional ACLs without using any extra bytes from any UDB sets.

Layer 2 protocol components on SBx81GS24a and SBx81XS6 line cards

The SBx81GS24a and SBx81XS6 line cards have a further limitation in their UDB usage. Out of the 6 UDBs available in each UDB set, only 2 can be used for Layer 2 protocol components. See Table 10 to see which protocol components are at Layer 2.

Examples of rule and UDB usage

The following table lists some example ACLs, showing the number of filters that get placed into hardware, and the number of UDBs that get consumed.

Table 11: Examples showing UDB usage

ACL MATCHES ON THIS STANDARD PROTOCOL COMPONENT	AND THIS OPTIONAL PROTOCOL COMPONENT	AND THEREFORE ADDS THESE FILTERS IN HARDWARE	AND CONSUMES BYTES IN THESE UDB SETS
IPv4 source address	inner VLAN ID	1 non-IPv6 filter	2 bytes from the non-IPv6 UDB sets
IPv6 source address	inner VLAN ID	1 IPv6 filter	2 bytes from the IPv6 UDB sets
MAC address	inner VLAN ID	1 IPv6 and 1 non-IPv6 filter	2 bytes from both IPv6 and non-IPv6
none	inner VLAN ID	1 IPv6 and 1 non-IPv6 filter	2 bytes from both IPv6 and non-IPv6
IPv6 destination address with /56 prefix length		1 IPv6 filter	1st byte from the standard protocol component and 6 bytes from the IPv6 UDB sets

ACL memory optimization for SBx8100, x220, x320, and FS980M Series

Introduction

From software version 5.5.1-1.1 onwards, the SBx8100, x220, x320, and FS980 Series ports can share some rules to optimize total ACL memory usage.

Previously, ACL rules memory usage was not very efficient on these platforms due to rule duplication:

- Global rules were duplicated for each port that had a per-port rule applied
- The same ACL rules were duplicated for each port they were applied to

Now, a global ACL rule occupies a single entry in the device's ACL memory and applies to all ports within that device. If multiple ports share the same per-port rule list, then the rules within that list are shared and not duplicated for each port.

How it works and some limitations

Identical rule list

Multiple ports can share their per-port rules as long as the rule list on each port is identical, i.e. the **same number** of rules are added in the **same order**

For example, the following configuration consumes a total of 3 ACLs:

```
awplus(config)#int port1.0.1
awplus(config-if)#access-group 3001
awplus(config-if)#access-group 3002
awplus(config-if)#access-group 3003
awplus(config-if)#int port1.0.2
awplus(config-if)#access-group 3001
awplus(config-if)#access-group 3002
awplus(config-if)#access-group 3003
```

While this configuration consumes a total of 5 ACLs:

```
interface port1.0.1
switchport
switchport mode access
access-group 3001
access-group 3002
access-group 3003
!
interface port1.0.2
switchport
switchport mode access
access-group 3001
access-group 3002
!
```

Policy maps

This feature also applies to policy-maps; if the same policy-map is applied on multiple ports, as long as the rule list for all the ports is still the same then the acls/class-map rules for that policy-map will be shared.

Here's an example where a policy-map is used but rules are **not** shared:

```
interface port1.0.1
switchport
switchport mode access
service-policy input pmap1
!
interface port1.0.2
switchport
switchport mode access
service-policy input pmap1
access-group 3002
!
```

Maximum number of shared per-port rules

The maximum number of per-port rules that can be shared between ports is limited by the maximum number of rules per platform and how the rules are configured.

The following table lists the maximum number of filters/rules that can be added per platform.

Table 12: Shared per-port rules by platform

PLATFORM	MAXIMUM FILTER NUMBER
SBx81GP24	1536
SBx81GT24	1536
SBx81GT40	1536
SBx81GC40	1536
SBx81XS16	1536
SBx81CFC960	1536
SBx81XLEM	5120
SBx81GS24a[a]	7168
SBx81XS6	7168
x220	512
x320	1536
FS980	496

a. The SB81GS24a has limited RAM memory so the maximum number of shared rules of 7168 can only be shared for up to 7 ports. The maximum number of rules that can be shared across all 24 ports is 2304. If those limits are exceeded, this may result in low memory issue.

The maximum number of rules shown in Table 12 above can only be shared if you use a combined port configuration when attaching the rules, otherwise the maximum is limited to half of the hardware capacity.

For example, on the x220 Series, 512 rules can be shared across all ports if the following configuration is used:

```
awplus(config)#int port1.0.1-1.0.28
awplus(config)#access-group 3001
awplus(config)#access-group 3002
...
awplus(config)#access-group 3512
```

The config file should look like this after attaching the rules to ports:

```
interface port1.0.1-1.0.28
switchport
switchport mode access
access-group 3001
access-group 3002
...
access-group 3512
!
```

If the rules to be shared are applied to each port individually or if the ports have other config that is different (causing the config file to have separate port config) then the maximum number of rules that can be shared are limited to HALF the maximum hardware limits shown in Table 12 above.

For example on the x220 Series, only up to 256 rules (half the limit) can be shared across two ports if the following config is used:

```
awplus(config)#int port1.0.1
awplus(config)#access-group 3001
awplus(config)#access-group 3002
...
awplus(config)#access-group 3256
awplus(config)#int port1.0.2
awplus(config)#access-group 3001
awplus(config)#access-group 3002
...
awplus(config)#access-group 3256
```

Also up to 256 rules can be shared on the x220 Series if the config file looks like this:

```
interface port1.0.1
 description port1
 switchport
 switchport mode access
 access-group 3001
 access-group 3002
 ...
 access-group 3256
!
interface port1.0.2
 description port2
 switchport
 switchport mode access
 access-group 3001
 access-group 3002
 ...
 access-group 3256
!
```

If more than half of the HW limit is added using combined port config, the shared rules must also be deleted using the combined port config.

For example in the x220 Series example above, you must remove the rules using the following config:

```
awplus(config)#int port1.0.1-1.0.28
awplus(config)#no access-group 3001
awplus(config)#no access-group 3002
.
awplus(config)#no access-group 3512
```

Timing delays

ACL optimization increases the maximum number of rules that can be attached on the SBx8100, x220, x320 and FS980 Series. However, when the number of rules approach the limit, it can cause delays in processing of adding/deleting of ports. We recommend that the shared rules are split across multiple named ACLs or added as individual ACLs instead of using a single named ACL.

For example, use the following configuration:

```
interface port1.3.1-1.3.20
 description port1
 access-group acl1-250  <<< includes 250 rules
 access-group acl2-250
 access-group acl3-250
```

Instead of this one:

```
interface port1.3.1-1.3.20
 description port1
 access-group acl-750   <<< includes 750 rules
```

Monitoring ACL rule use

Use the command: **show platform classifier statistics utilisation brief** to display the number of ACL rules used per device. If the rules are shared across multiple ports then each shared rule is only counted once.

```
awplus#show platform classifier statistics utilization brief

[Instance 0]
  [port1.0.1-port1.0.10]
 Usage:
                  Used / Total
  --------------------------------
  System              0
  DHCP Snooping       2
  Loop Detection      0
  EPSR                0
  CFM                 0
  G8032               0
  Global ACL          2
  ACL                 2
  VACL                0
  QoS                 0
  RA Guard            0
  BEFD                0
  AMFAPPS             0
  Openflow Hybrid     0
  Openflow Flow       0
  Openflow Default    0
  Pre-Ingress         2
Egress:
  VLAN Xlate          1
  VLAN Isolate        0
  VLAN IsolateDef     0
  Total               9 / 1536 (0.59%)
```

ACL memory optimization on the x530 Series

From version 5.5.0 onwards, the x530 Series support ACL optimization but it works differently in that the rule list does not have to be identical for the rules to be shared, the rules have to be applied in the same order but the total number of rules per port doesn't have to be identical.

In the example below, two rules are shared between port1.0.2 and port1.0.3 (acl 3001 and 3000) and one rule is shared between port1.0.1 and port1.0.2 (acl 3003) but remaining rules on port1.0.1 are duplicated/not shared:

```
awplus(config-if)#int port1.0.1
awplus(config-if)#access-group 3000
awplus(config-if)#access-group 3001
awplus(config-if)#access-group 3002
awplus(config-if)#access-group 3003
awplus(config-if)#
awplus(config-if)#int port1.0.2
awplus(config-if)#access-group 3003
awplus(config-if)#access-group 3001
awplus(config-if)#access-group 3000
awplus(config-if)#
awplus(config-if)#int port1.0.3
awplus(config-if)#access-group 3001
awplus(config-if)#access-group 3000
```

TCAM entries will be used as in the following table:

```
awplus#show pl swtable qostcam inst 1
 ------------------------------------------------------------++
 Idx |precd            |     lport | Owner |      Name      ||
 ----+-----------------+-----------+-------+----------------++
    0|ACLP             |       285|      2|   3000 00040000||
  ..  ..............  ..........  .....  ..............  ||
   12|ACLP             |       285|      4|   3001 00040000||
  ..  ..............  ..........  .....  ..............  ||
   24|ACLP             |       285|      5|   3002 00040000||
  ..  ..............  ..........  .....  ..............  ||
   36|ACLP             |       284|      7|   3003 00040000||
   36|ACLP             |       285|      6|   3003 00040000||
  ..  ..............  ..........  .....  ..............  ||
   48|ACLP             |       287|     10|   3001 00040000||
   48|ACLP             |       284|      8|   3001 00040000||
  ..  ..............  ..........  .....  ..............  ||
   60|ACLP             |       287|     11|   3000 00040000||
   60|ACLP             |       284|      9|   3000 00040000||
  ..  ..............  ..........  .....  ..............  ||
```

ACL filter sequence numbers

To help you manage ACLs you can apply sequence numbers to filters. This allows you to remove filters from named and numbered ACLs without having to reconfigure an ACL.

The ability to add sequence numbers to filters simplifies updates through the ability to position a filter within an ACL. When you add a new filter, you can specify a sequence number to position the filter in the ACL and you can also remove a current filter in an ACL by specifying a sequence number.

ACL filter sequence number behavior

- If filters with no sequence numbers are applied, then the first filter is assigned a sequence number of 4, and successive filters are incremented by 4. Sequence numbers are generated automatically if they are not specified at entry.
- The maximum filter sequence number is 65535. If the sequence number exceeds this maximum, the command will not be recognized and will show the error message: **% Unrecognized command**
- If you enter a filter without a sequence number, it is assigned a sequence number that is 4 greater than the last sequence number and is placed at the end of the ACL.
- If you enter a filter with a sequence number which matches the sequence number on an existing filter within the same ACL, then the existing filter is overwritten with the subsequent filter.
- ACL sequence numbers determine the order of execution of filters in an ACL. Filters in a ACL with a lower value sequence number are executed before filters with a higher value.
- Output from the **show running-config** command displays ACL entries without filter sequence numbers. Output from relevant **show** commands displays ACL entries with their sequence numbers.
- ACL sequence numbers are re-numbered upon switch restart following a **reload** command, or after powering off and powering on the switch. ACL sequence numbers are renumbered starting from 4 and increment by 4 for each filter. See the sample output in the configuration section that follows for an illustration of this behavior. No ACL sequence number re-number command is available to perform this action.
- The ACL sequence number feature works with numbered and named standard and extended IPv4 and IPv6 access lists, plus named hardware IPv4 and IPv6 access lists.
- The name of an access list can be designated as a number. Number in named ACLs must not exist within the range of designated numbered ACLs. (where <1-99> and <1300-1999> are standard numbered ACLs, <100-199> and <2000-2699> are extended numbered ACLs, <3000-3699> and <4000-4699> are hardware numbered ACLs).

ACL filter sequence number applicability

Numbered hardware ACLs are available in the range <*4000-4699*>, which permit or deny MAC source addresses, MAC destination addresses, and VLAN IDs to control packets coming from and going to network devices and hosts, in hardware.

ACL filter sequence number types

There are ACL filter sequence numbers available for the following types of ACLs:

Table 13: ACL types and sequence numbers

ACL TYPE	ACL COMMAND SYNTAX
IPv4 Standard Numbered ACLs	access-list <1-99> access-list <***1300-1999***>
IPv4 Extended Numbered ACLs	access-list <***100-199***> access-list <***2000-2699***>
IPv4 Standard Named ACLs	access-list standard <***name***>
IPv4 Extended Named ACLs	access-list extended <***name***>
IPv4 Hardware Named ACLs	access-list hardware <***name***>
IPv6 Standard Named ACLs	ipv6 access-list standard <***name***>
IPv6 Extended Named ACLs	ipv6 access-list extended <***name***>
IPv6 Hardware Named ACLs	ipv6 access-list <***name***>

Note that ACL sequence number support for these ACL commands is optional not required. An ACL sequence number will be added automatically, starting at 10 and incrementing by 10.

ACL commands without ACL filter sequence numbers

ACL filter sequence numbers are not available for numbered hardware ACL commands:

```
access-list <3000-3699>
access-list <4000-4699>
```

When using numbered hardware ACLs, the numbered ACLs are all created individually, and applied to an interface in a series of **Access-Group** commands. The order in which the ACLs are actioned is governed by the order in which the **Access-Group** commands are configured.

But, with named hardware ACLs, the named ACL contains a series of ACL filters within it. This order in which these filters are actioned is defined by their sequence numbers.

ACL filter sequence configuration

First create a named or numbered ACL to enter ACL filters in the ACL sub-modes available:

Step 1: **Create a new ACL and add a new filter**

Create ACL **1** and then add a new filter to the access-list to permit all packets from the **192.168.1** subnet:

```
awplus# # configure terminal
awplus(config)# access-list
awplus(config-ip-std-acl)# permit 192.168.1.0 0.0.0.255
awplus(config-ip-std-acl)# end
awplus# show access-list 1
```

```
Standard IP access list 1
   4 permit 192.168.1.0, wildcard bits 0.0.0.255
```

In the output above, you can see that, even though no sequence number was included in the command that created the filter entry, the entry has been automatically assigned the sequence number 4.

Step 2: **Add another filter to the ACL**

Append to (or add at the end of) ACL ***1*** a new filter to deny all packets from the **192.168.2** subnet:

```
awplus# configure terminal
awplus(config)# access-list 1
awplus(config-ip-std-acl)# deny 192.168.2.0 0.0.0.255
awplus(config-ip-std-acl)# end
awplus# show access-list 1
```

```
Standard IP access list 1
   4 permit 192.168.1.0, wildcard bits 0.0.0.255
   8 deny   192.168.2.0, wildcard bits 0.0.0.255
```

In the output above, you can see that, even though no sequence number was included in the command that created the second filter entry, the entry has been automatically assigned the sequence number 8.

So, if you add a filter to an ACL without specifying a sequence number the new filter is automatically assigned a sequence number. Sequence numbers are assigned in multiples of four from the sequence number of the last filter.

Step 3: **Insert a filter into the ACL**

Insert a new filter with the sequence number **6** into ACL **1** to permit packets from the **192.168.3** subnet:

```
awplus# configure terminal
```

```
awplus(config)# access-list 1
awplus(config-ip-std-acl)# 6 permit 192.168.3.0 0.0.0.255
awplus(config-ip-std-acl)# end
awplus# show access-list 1
```

```
Standard IP access list 1
   4 permit 192.168.1.0, wildcard bits 0.0.0.255
   6 permit 192.168.3.0, wildcard bits 0.0.0.255
   8 deny   192.168.2.0, wildcard bits 0.0.0.255
```

The new filter has precedence over the filter with the sequence number **8**.

Step 4: **Remove a filter from the ACL by specifying a filter pattern**

Remove the filter with the IP address **192.168.2** from **ACL 1**:

```
awplus# configure terminal
awplus(config)# access-list 1
awplus(config-ip-std-acl)# no deny 192.168.2.0 0.0.0.255
awplus(config-ip-std-acl)# end
awplus# show access-list 1
```

```
Standard IP access list 1
   4 permit 192.168.1.0, wildcard bits 0.0.0.255
   8 permit 192.168.3.0, wildcard bits 0.0.0.255
```

Step 5: **Remove a filter from the ACL by specifying a sequence number**

Remove the filter with the sequence number **4** from **ACL 1**:

```
awplus# configure terminal
awplus(config)# access-list 1
awplus(config-ip-std-acl)# no 4
awplus(config-ip-std-acl)# end
awplus# show access-list
```

```
Standard IP access list 10
   6 permit 192.168.3.0, wildcard bits 0.0.0.255
```

Creating ACLs in Global Configuration mode

In the case of some software ACLs, you can add new filters in **Global Configuration** mode with the **access-list** command. In this mode the filters are assigned a sequence number corresponding to the order in which they are entered, i.e. the first filter entered has higher precedence in the ACL.

Note - this approach to adding filters to an ACL is not available for named hardware ACLs.

Step 1: **Add filters with the access-list command**

Add filters to ACL **1** using the **access-list** command:

```
awplus# configure terminal
awplus(config)# access-list 1 permit 192.168.1.0 0.0.0.255
awplus(config)# access-list 1 deny 192.168.2.0 0.0.0.255
awplus(config)# end
awplus# show access-list 1
```

```
Standard IP access list 1
   6  permit 192.168.3.0, wildcard bits 0.0.0.255
   8  permit 192.168.1.0, wildcard bits 0.0.0.255
   12 deny 192.168.2.0, wildcard bits 0.0.0.255
```

You can then enter the IPv4 **Standard ACL Configuration** mode and specify sequence numbers to reorder the filters, as shown in the next step.

Step 2: **Reorder the filters**

Reorder the filters in ACL **1** by specifying a sequence number for each filter. The specified sequence number will overwrite the previous sequence number assigned to the filter:

```
awplus# configure terminal
awplus(config)# access-list 1
awplus(config-ip-std-acl)# 1021 permit 192.168.1.0 0.0.0.255
awplus(config-ip-std-acl)# 3333 permit 192.168.3.0 0.0.0.255
awplus(config-ip-std-acl)# 2772 deny 192.168.2.0 0.0.0.255
awplus(config-ip-std-acl)# end
awplus# show access-list 1
```

```
Standard IP access list 1
 1021 permit 192.168.1.0, wildcard bits 0.0.0.255
 2772 deny   192.168.2.0, wildcard bits 0.0.0.255
 3333 permit 192.168.3.0, wildcard bits 0.0.0.255
```

Step 3: **Copy the running-config file into the startup-config file**

Copy the running-config into the file set as the current startup-config file and then reload the device. When the device has rebooted you can then enter **Global Configuration** mode and use the **show access-list** command to display **ACL 1**:

```
awplus(config)# exit
awplus# copy running-config startup-config
awplus# reload
awplus# show access-list 1
```

```
Standard IP access list 1
   4 permit 192.168.1.0, wildcard bits 0.0.0.255
   8 deny   192.168.2.0, wildcard bits 0.0.0.255
   12 permit 192.168.3.0, wildcard bits 0.0.0.255
```

After the device has rebooted, the sequence numbers of the filters in the ACL have been reassigned, incrementing from 4.

Display the ACL configuration details

Display the running system status and configuration details for ACLs:

```
awplus# show running-config access-list
```

```
access-list 1 deny 10.1.1.0 0.0.0.255
access-list 1 permit any
access-list 2
access-list 5
access-list 10 permit 192.168.1.0 0.0.0.255
access-list 10 deny 192.168.2.0 0.0.0.255
access-list 10 permit 192.168.3.0 0.0.0.255
access-list 20
access-list 25 permit 10.1.2.0 0.0.0.255
access-list 25 deny 192.168.1.0 0.0.0.255
access-list 50
access-list 95 permit any
access-list 100
access-list 1300
access-list 2000
access-list extended acl
access-list extended my-list
access-list extended name
access-list extended name1
access-list standard name3
ipv6 access-list extended ipv6_acl
ipv6 access-list standard ipv6_acl2
ipv6 access-list extended my-ipv6-list
ipv6 access-list extended my-list
ipv6 access-list standard my-new-list
ipv6 access-list standard name
ipv6 access-list standard name1 deny any
ipv6 access-list extended name5
ipv6 access-list standard name6
access-list hw_acl
access-list icmp
access-list my-hw-list
access-list name2
access-list name4
```

ACL source and destination addresses

Configure source addresses in ACL filters to filter packets coming **from** specified networking devices or hosts. Configure destination addresses in ACL filters to filter packets going **to** specified networking devices or hosts.

ACL reverse masking

ACLs uses reverse masking, also referred to as wildcard masking, to indicate to the switch whether to check or ignore corresponding IP address bits when comparing the address bits in an ACL filter to a packet being submitted to the ACL.

Reverse masking for IP address bits specify how the switch treats the corresponding IP address bits. A reverse mask is also called an inverted mask because a 1 and 0 mean the opposite of what they mean in a subnet or a network mask.

- A reverse mask bit 0 means check the corresponding bit value.
- A reverse mask bit 1 means ignore the corresponding bit value.

C613-22039-00 REV M

North America Headquarters | 19800 North Creek Parkway | Suite 100 | Bothell | WA 98011 | USA | T: +1 800 424 4284 | F: +1 425 481 3895
Asia-Pacific Headquarters | 11 Tai Seng Link | Singapore | 534182 | T: +65 6383 3832 | F: +65 6383 3830
EMEA & CSA Operations | Incheonweg 7 | 1437 EK Rozenburg | The Netherlands | T: +31 20 7950020 | F: +31 20 7950021

alliedtelesis.com