White Paper

Understanding ACL Merge Algorithms and ACL Hardware Resources on Cisco Catalyst 6500 Switches

This document provides information to help you understand the Access Control List (ACL) merge algorithms and the hardware resources used in Cisco Catalyst 6500 switches to enforce security and apply quality of service (QoS) using router ACLs (RACLs), VLAN ACLs (VACLs), and QoS ACLs.

Introduction

The purpose of this document is to provide information on how the Cisco® Catalyst® 6500 Series switches program ACLs in hardware, and how in certain configurations, careful planning is needed to prevent exhaustion of limited hardware resources relating to ACLs.

This document discusses the ACL merge algorithms and ACL-related hardware resources that apply to the Cisco Catalyst 6500 switches running hybrid software (Cisco Catalyst OS on the Supervisor Engine and IOS on the Multilayer Switch Feature Card [MSFC], if present) or Cisco IOS® Software on the Supervisor software (single combined Cisco IOS Software image running on the Supervisor Engine and MSFC).

The information discussed in this document is general and does not apply to specific software releases. However, later software releases are likely to include more ACL merge, feature manager, and resource-usage optimizations. Be sure to read the release notes for your platform before upgrading Supervisor Engine or MSFC software.

Note: This document is not intended to provide specific configuration examples for each type of ACL. For specific configuration information, see the "References and Related Information" section on page 31.

Overview of ACLs on Cisco Catalyst 6500 Series Switches

The main issue users face when configuring ACLs on the Cisco Catalyst 6500 Series switches are resource contention and exhaustion. Because the platform enforces several types of ACLs in hardware rather than in software, the switch programs hardware lookup tables and various hardware registers in the Policy Feature Card (PFC), PFC2, or PFC3, so that when a packet arrives, the switch can perform a hardware table lookup and perform the appropriate action.

For typical configurations, the main hardware resources we are concerned with are the mask-value entries (which we will call masks through the rest of this paper) and pattern-value entries (which we will call patterns) in the ternary content addressable memory (TCAM); the Logical Operation Units (LOUs); and the ACL-to-switch interface mapping labels.

TCAM entries, LOUs, and ACL labels are limited resources. Therefore, depending on your ACL configuration, you might need to

be careful not to exhaust the available resources. In addition, with large QoS ACL and VACL configurations, you also might need to consider Non-Volatile Random Access Memory (NVRAM) space.

The available hardware resources differ on Supervisor 1a with PFC, Supervisor 2 with PFC2, and Supervisor 720 with PFC3. The resources shown in Table 1 are discussed in detail later in this document.

Table 1 ACL Hardware Resources by Supervisor Engine

Supervisor Model	TCAM	LOUs	L4Ops[1] per ACL	ACL Labels[2]	Sup NVRAM
Supervisor 1a and PFC	• 2K masks and 16K patterns shared between RACLs, VACLs and QoS ACLs	• 32 LOUs (64 registers) shared between RACLs, VACLs, and QoS ACLs	• 9 per ACL (further L4Ops result in LOU expansion)	• 512 ACL labels shared between RACLs, VACLs, and QoS ACLs	• 512 KB
Supervisor 2 and PFC2	• 4K mask values and 32K pattern values for RACLs (including policy-based routing ACLs) and VACLs • 4K masks and 32K patterns for QoS ACLs	• 32 LOUs (64 registers) shared between RACLs, VACLs, and QoS ACLs	• 10 per ACL (further L4Ops result in LOU expansion)	• 512 ACL labels shared between RACLs, VACLs, and QoS ACLs	• 512KB
Supervisor 720 and PFC3	• 4K mask values and 32K pattern values for RACLs (including policy-based routing ACLs) and VACLs • 4K masks and 32K patterns for QoS ACLs	• 32 source LOUs (64 registers) and 32 destination LOUs (64 registers), shared between RACLs, VACLs, and QoS ACLs	• 10 per ACL (further L4Ops result in LOU expansion)	• 512 ACL labels shared between RACLs, VACLs, and QoS ACLs	• 2MB

1. L4Ops=Layer 4 operations

2. Independent of the 512 ACL label limit, there is an additional software limit in Cisco Catalyst OS of 250 QoS ACLs system-wide when using the default (binary) configuration mode. This restriction is removed in text configuration mode.

ACLs Processed in Hardware in Cisco Catalyst 6500 Series Switches

The Cisco Catalyst 6500 Series switches support the following types of ACLs and ACL-based features in hardware with no performance impact:

- IP standard and extended ACLs permit and deny (with Supervisor 1a with PFC use the **no ip unreachable**s command on VLAN and other interfaces)
- VACLs
  - Supervisor 720—Supports IP and MAC VACLs
  - Supervisor 1a with PFC and Supervisor 2 with PFC2—Support IP, IPX, and MAC VACLs

- Reflexive (session-based) ACLs—Limit of 512 sessions on Supervisor 1a with PFC; 32K sessions on Supervisor 2 with PFC2; 96K sessions on Supervisor 720 with PFC3
- Policy routed traffic
  - Supervisor Engine 2 with PFC2 and Supervisor 720 with PFC3—Supported in hardware unless `match length`, **`set ip precedence`**, or other unsupported parameters are used. In addition, the set **`interface Null0`** parameter is supported in hardware; other **`set interface`** parameters are handled in software
  - Supervisor 1a with PFC—Same restrictions as Supervisor 2, except does not support the set interface `Null0` command in hardware. Requires the `mls ip pbr` command with caveats.
- Unicast Reverse Path Forwarding (RPF) check with ACL—When you configure ACL-based uRPF check on Supervisor 2 with PFC2 or Supervisor 720 with PFC3, only traffic matching a "permit" access control entry (ACE) in the uRPF ACL is handled in hardware
- Transmission Control Protocol (TCP) intercept
  - Supervisor 1a with PFC—Traffic denied in a TCP intercept ACL is handled in hardware
  - Supervisor 2 with PFC2 and Supervisor 720 with PFC3—All TCP session traffic, except for the TCP three-way handshake (SYN, SYN/ACK, ACK) and session close (FIN/RST), is handled in hardware
- Web Cache Communication Protocol (WCCP) redirection for HyperText Transfer Protocol (HTTP) requests (Supervisor 2 with PFC2 and Supervisor 720 with PFC3 only)—HTTP packets with the SYN, FIN, or RST bits in the TCP header set are redirected to the CPU; other packets are processed in hardware
- Traffic requiring NAT, provided the NAT translation and NetFlow setup is already completed (Supervisor 720 with PFC3 only)
- Internetwork Packet Exchange (IPX) standard ACLs—Matching on IPX source and destination net, and/or destination node only (Supervisor 1a and Supervisor 2 only)
- IPX extended ACLs—Matching on IPX source and destination net, destination node, and/or destination protocol only (Supervisor 1a and Supervisor 2 only)
- Time-based ACLs—Processed in hardware as long as the time-based ACEs are supported in hardware

ACLs Processed in Software in Cisco Catalyst 6500 Series Switches

It is important to understand that some ACL and ACL-based feature configurations are not supported in hardware on the PFC/PFC2/PFC3, and will cause some traffic to be redirected to the MSFC for ACL processing in software. The following configurations will result in software processing:

- ACL denied traffic
  - Supervisor 1a with PFC—ACL denied packets are processed in software if interface does not have the **`no ip unreachables`** command configured
  - Supervisor 2 with PFC2—ACL denied packets are leaked to the MSFC2 if unreachables are enabled. Packets are leaked at 10 packets per second (pps) per VLAN (Catalyst OS software with Cisco IOS Software) or one packet every two seconds per VLAN (Cisco IOS Software)
  - Supervisor 720 with PFC3—ACL denied packets are leaked to the MSFC3 if unreachables are enabled. Packets requiring ICMP unreachables are leaked at a user-configurable rate (500 pps by default)
- Traffic denied in an output ACL (Supervisor 1a with PFC only)—If traffic is denied in an output ACL, an MLS cache entry is never created for the flow. Therefore, subsequent packets do not match a hardware cache entry and are sent to the MSFC where they are denied in software

- IPX filtering based on unsupported parameters (such as source host); on Supervisor 720, Layer 3 IPX traffic is always processed in software
- ACEs requiring logging (**log** keyword)—ACEs in the same ACL that do not require logging are still processed in hardware; Supervisor 2 with PFC2 and Supervisor 720 with PFC3 support rate-limiting of packets redirected to the MSFC for ACL logging.
- Transmission Control Protocol (TCP) intercept
  - Supervisor 1a with PFC—Traffic permitted in a TCP intercept ACL is handled in software
  - Supervisor 2 with PFC2 and Supervisor 720 with PFC3—The TCP three-way handshake (SYN, SYN/ACK, ACK) and session close (FIN/RST) are handled in software; all remaining traffic is handled in hardware
- Policy routed traffic (if **match length**, **set ip precedence**, or other unsupported parameters are used; if the **mls ip pbr** command is not configured with Supervisor 1a with PFC). The **set interface** parameter is supported in software, with the exception of the **set interface Null0** parameter, which is handled in hardware on Supervisor 2 with PFC2 and Supervisor 720 with PFC3.
- WCCP redirection for HTTP requests (Supervisor 1a with PFC only)
- Traffic requiring Network Address Translation (NAT) (Supervisor 1a with PFC and Supervisor 2 with PFC2); traffic requiring NAT translation or NetFlow setup (Supervisor 720 with PFC3)
- Unicast RPF check
  - Supervisor 2 with PFC2 and Supervisor 720 with PFC3—Traffic denied in a uRPF check ACL ACE
  - Supervisor 1a with PFC—Any uRPF check configuration
- Non-IP (all Supervisors) and non-IPX (Supervisor 1a with PFC and Supervisor 2 with PFC2 only) RACLs
- Broadcast traffic denied in a RACL

ACL Performance Characteristics

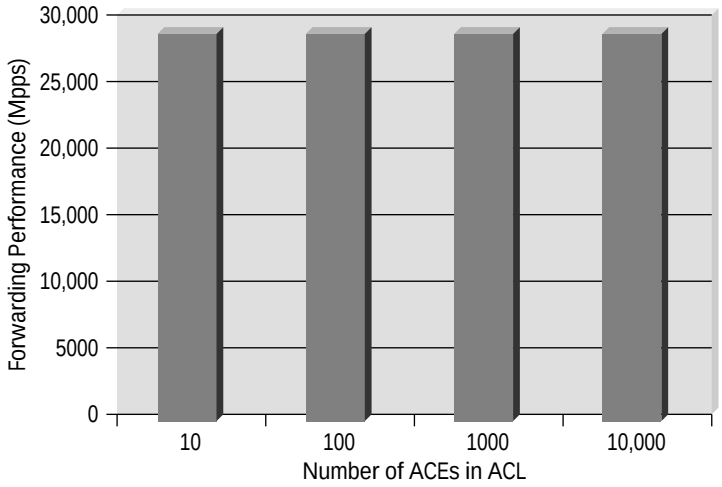
Forwarding performance on the Cisco Catalyst 6500 Series switches remains 100 percent consistent regardless of the ACL configuration, assuming the following is true:

- The configuration consists of feature ACLs that are supported in hardware (see the section “ACLs Processed in Hardware in Cisco Catalyst 6500 Series Switches” section on page 2)
- Hardware resources are not exhausted

Figure 1 and Figure 2 illustrate the deterministic performance characteristics of the Cisco Catalyst 6500 Series switches with an ACL configuration applied.

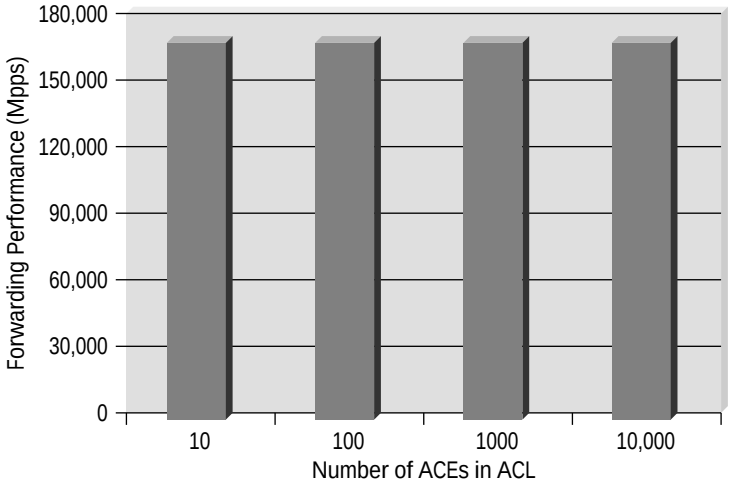
In a test of the Supervisor Engine 2 using the centralized forwarding engine (PFC2), 29.1 million packets per second (Mpps) performance is maintained through the switch regardless of whether a 10-line ACL or a 10,000-line ACL is applied (Figure 1).

Figure 1
Supervisor 2 with PFC2 ACL Performance 20 Gigabit Ethernet Ports, Central Forwarding



Similarly, in a test of the Supervisor Engine 2 using modules equipped with the Distributed Forwarding Cards (DFCs), 169.46 Mpps performance is maintained through the switch, again regardless of the size of the ACL configuration (Figure 2).

Figure 2
Supervisor 2 with PFC2 ACL Performance—114 Gigabit Ethernet Ports, Distributed Forwarding



The remainder of this document discusses how the ACL hardware is managed, what the limitations are, and how to get the most out of the ACL hardware on the Cisco Catalyst 6500 Series switches.

ACL Merge Algorithms

The Catalyst 6500 Supervisor Engines support a limited number of lookups in the ACL TCAMs per packet. On Supervisor 1a with PFC and Supervisor 2 with PFC2, four lookups are performed on each packet passing through the system:

- Ingress security lookup
- Egress security lookup

- Ingress QoS lookup
- Egress QoS lookup

On Supervisor 720 with PFC3, two security lookups and one QoS lookup are supported in each direction, for a total of six TCAM lookup results.

The limited number of lookups in the hardware dictates how the system must treat certain configurations. Consider the case where you configure two "ingress security" features on an interface on a Supervisor 1a with PFC or Supervisor 2 with PFC2 (for example, a VLAN interface with **`ip access-group in`** and **`ip nat outside`**).

Because the PFC/PFC2 can only perform one ingress security lookup per packet, the software must combine the different access control entries (ACEs) in each ACL to arrive at a single group of ACEs and the correct corresponding results based on the two ACLs. The merge process is also responsible for other functions such as expanding ACEs (due to lack of L4Op Pointers or LOUs).

The Supervisor 720 with PFC3 architecture is optimized for scenarios where you configure two "ingress security" or two "egress security" features on a single interface. These improvements are described later in this paper. However, even on Supervisor 720 with PFC3, in some situations, the system must combine the ACEs from multiple ACLs (for example, if you configure three "ingress security" features on a single interface).

The process of combining the ACEs from multiple feature ACLs is known as the "ACL merge." Catalyst 6500 switches support two different merge algorithms, both of which accomplish the goal of generating a single list of ACEs and their corresponding results:

- Order Independent Merge—Supported on Supervisor 1a with PFC and Supervisor 2 with PFC2
- Order Dependent Merge—Supported on all Supervisor Engines

Regardless of the algorithm used, the result of the ACL merge is a list of "value, mask, result" (VMR) entries. See the "Ternary CAM (TCAM)" section on page 11 for more details. Note that, assuming the entries are successfully installed in the TCAM, neither algorithm affects the hardware forwarding performance of the system.

Order Independent Merge

The order independent merge (OIM) is based on Binary Decision Diagrams (BDD), a bit-based method of representing Boolean expressions. Simply stated, the Catalyst 6500 system uses BDDs to represent the VMRs for the combined ACLs in an *order independent* fashion. The OIM is only supported on the Supervisor 1a with PFC and Supervisor 2 with PFC2.

In theory, order independence is very desirable for a TCAM based ACL solution because of the limitations in the number of supported masks in current TCAM designs (see the "Ternary CAM (TCAM)" section on page 11). BDD produces the most efficient order-independent list of VMRs in terms of mask sharing.

However, the OIM has several disadvantages. These include:

- The OIM represents ACLs as BDDs, which can use a considerable amount of system memory
- The OIM is slow due to memory allocations, building BDDs, and processing intermediate results
- Order independent results can require many unique masks that in fact reduce mask sharing—with large or complex combinations of ACLs, this can result in "TCAM blowup," the case where there are an insufficient number of masks available to install the entire group of VMRs in the TCAM

The order independent merge is enabled by default in all 12.1E and 12.1EX releases.

Order Dependent Merge

Unlike the OIM, the order dependent merge (ODM) is not bit-based. The algorithm does not represent ACLs as BDDs but rather manipulates the VMRs themselves, using truth tables and merge result rules. The ODM addresses the key weaknesses of the OIM. For example:

- Order dependent results do not require as many unique masks
- The ODM represents ACLs as groups of VMRs, not as BDDs, which reduces the system memory usage for the merge
- The use of VMR format also makes the ODM faster than the OIM

The ODM process does not require the large number of intermediate results produced by BDD, and typically reduces the number of masks used compared to the OIM, which produces order-independent results.

For security ACLs, the order dependent merge is disabled by default in Supervisor 1a with PFC and Supervisor 2 with PFC2 running 12.1E based software, but can be enabled in Cisco IOS Software on the Supervisor releases 12.1(8b)EX, 12.1(11b)E, and later releases. In hybrid systems, Catalyst OS support for ODM is available in release 7.1(1) and later.

For QoS ACLs, ODM is supported in Cisco IOS Software Release 12.1(12c)E1 and later.

Supervisor 720 with PFC3 supports only ODM. In addition, Supervisor 1a with PFC and Supervisor 2 with PFC2 systems running 12.2S or 12.2SX based Cisco IOS Software images have ODM enabled by default.

Beginning in Cisco IOS Software Release 12.1(13)E, and in all 12.2S and 12.2SX based Cisco IOS Software images, an improved version of ODM is implemented. This version, known as ODM v2, has several advantages over ODM v1:

- Smaller merge results (fewer VMRs for the same merge)
- Lower CPU and memory usage and faster execution of the ACL merge
- Extensible and flexible platform independent design

Also beginning in 12.1(13)E, and in all 12.2S and 12.2SX based Cisco IOS Software images, several ODM optimizations are introduced that improve ODM ACL merge results, particularly for certain feature combinations such as security ACLs with PBR, and security ACLs with Cisco IOS SLB.

In 12.1E based Cisco IOS Software images, the ODM optimizations are enabled using a hidden command, **`mls aclmerge odm optimizations`**. These optimizations are enabled by default in all 12.2S and 12.2SX based Cisco IOS Software images, but can be disabled using the **`no mls aclmerge odm optimizations`** command. For QoS ACLs, the optimizations are available beginning in Cisco IOS Software Release 12.1(12c)E1.

Comparing ODM to OIM

In general, experience has shown that with real-world ACL feature configurations, the ODM produces more efficient results than the OIM. Because the ODM produces fewer VMRs, fewer unique masks are required in the TCAM, and therefore ACLs are more likely to fit within the limited number of entries in the TCAM.

Note: While ACLs are more likely to fit in the TCAM when using the ODM, using the ODM does not change the other restrictions imposed by the hardware—for example, LOUs, L4Ops, and ACL labels (these hardware elements are discussed elsewhere in this document).

To illustrate the superior capabilities of the ODM compared to the OIM, consider a feature ACL configuration, in a switch with Supervisor 2 with PFC2 running Cisco IOS Software, consisting of an interface with two input features enabled—**ip access-group in** and **ip nat outside**:

```
interface Vlan100
   description Interface with Two Input Features
   ip address 10.1.1.1 255.255.255.0
   ip access-group TestACL in
   ip nat outside
  !
  ip access-list extended TestACL
   permit udp host 10.1.1.3 host 224.0.0.2 eq 1985
   permit icmp any any
   permit tcp 10.1.1.0 0.0.0.63 range 3000 3100 172.16.0.0 0.0.255.255 gt 1023
   permit tcp 10.1.1.0 0.0.0.63 range 3000 3100 172.18.0.0 0.0.255.255 gt 1023
   permit tcp 10.1.1.0 0.0.0.63 range 3000 3100 172.19.0.0 0.0.255.255 gt 1023
   permit tcp 10.1.1.0 0.0.0.63 range 3000 3100 10.15.0.0 0.0.255.255 gt 1023
   permit tcp 10.1.1.0 0.0.0.63 10.15.168.0 0.0.0.255 eq bgp
  !
  ip nat outside source static 204.175.41.9 10.160.19.205
  ip nat outside source static 204.175.40.9 10.160.19.204
  ip nat outside source static 204.175.45.102 10.160.19.203
  ip nat outside source static 204.175.44.102 10.160.19.202
  ip nat outside source static 204.175.41.102 10.160.19.201
  ip nat outside source static 204.175.40.102 10.160.19.200
  ip nat outside source static 207.248.0.155 10.160.17.216
  ip nat outside source static 148.246.250.70 10.160.17.215
  ip nat outside source static 63.75.63.46 10.160.16.39
  ip nat outside source static 63.75.63.49 10.160.16.40
  ip nat outside source static 63.75.63.14 10.160.16.41
  ip nat outside source static 63.75.63.121 10.160.16.42
  ip nat outside source static 63.75.63.126 10.160.16.43
  ip nat outside source static 198.207.140.100 10.160.16.44
  ip nat outside source static 209.208.175.209 10.160.19.23
  ip nat outside source static 209.208.175.210 10.160.19.24
  ip nat outside source static 209.208.175.211 10.160.19.25
  ip nat outside source static 209.208.175.219 10.160.19.26
  ip nat outside source static 209.208.175.220 10.160.19.27
  ip nat outside source static 209.208.175.221 10.160.19.28
  ip nat outside source static 209.208.175.222 10.160.19.29
```

The output of the **show tcam counts** command shows that with the OIM enabled (**mls aclmerge algorithm bdd** global configuration command), 176 masks and 210 patterns are consumed with this configuration.

```
6509#show tcam counts
            Used        Free        Percent Used       Reserved
            ----        ----        ------------       --------
 Labels:       3         509            0

ACL_TCAM
  Masks:     176        3920            4                    0
Entries:     210       32558            0                    0

QOS_TCAM
  Masks:       1        4095            0                    0
Entries:       8       32760            0                    0

    LOU:       3          61            4
  ANDOR:       0          16            0
  ORAND:       0          16            0
    ADJ:       0        1024            0
```

Note: For more information about the show tcam counts command, see the "Monitoring TCAM and Other Hardware Resource Usage" section on page 19.

With the ODM (**mls aclmerge algorithm odm**), however, only 22 masks and 73 patterns are consumed:

```
6509#show tcam counts
            Used        Free        Percent Used       Reserved
            ----        ----        ------------       --------
 Labels:       3         509            0

ACL_TCAM
  Masks:      22        4074            0                    0
Entries:      73       32695            0                    0

QOS_TCAM
  Masks:       1        4095            0                    0
Entries:       8       32760            0                    0

    LOU:       3          61            4


  ANDOR:       0          16            0
  ORAND:       0          16            0
    ADJ:       0        1024            0
```

The previous example clearly illustrates that the ODM significantly improves the efficiency of the merge process, saving limited TCAM resources and typically supporting much larger and more complex ACL feature configurations.

Table 2 shows the commands used to enable OIM and ODM in hybrid and Cisco IOS Software on the Supervisor systems.

Table 2 Commands to Configure ACL Merge Algorithm

	OIM	ODM
Cisco Catalyst OS software on the Supervisor, Cisco IOS software on the MSFC	Supervisor: • `set aclmerge algo bdd` MSFC: • `mls acl algorithm bdd`	Supervisor: • `set aclmerge algo odm` MSFC: • `mls aclmerge algorithm odm` • `mls aclmerge odm optimizations`
Cisco IOS Software on the Supervisor	• `mls acl algorithm bdd`	• `mls aclmerge algorithm odm` • `mls aclmerge odm optimizations`

ACL Feature Manager and ACL Optimizations

The ACL feature manager is the software responsible for converting the individual ACEs of each configured ACL into the actual VMRs that will be installed in the TCAM. These VMRs might be generated directly from ACEs you configure or from ACEs that are the result of the ACL merge being invoked.

The core function of the ACL feature manager is to perform the feature merge and generate VMRs based on the ACL configuration. However, the feature manager also performs certain optimizations on the ACLs and VMRs, regardless of the merge algorithm used in the system.

One important optimization eliminates ACEs that will never be checked. For example, a given ACE could be a subset of a preceding or subsequent ACE, or it could be a subset of multiple preceding or subsequent ACEs. If an ACE does not match any packets that are not already covered by previous or subsequent ACEs (that is, it is a complete subset of those ACEs), that ACE can be removed.

For example, if you have an ACL that includes various **`permit`** and **`deny`** statements, and that ends in a **`permit any`** statement, the ACL feature manager might be able to save TCAM space by never actually installing VMRs into the TCAM for the permit statements you configured, because the final **`permit any`** may suffice.

Another important optimization is removing overridden ACEs. For example, suppose you configure two input security features, **`ip access-group in`** (security ACL) and **`ip policy route-map`** (policy routing). If the result for an ACE in the first feature ACL (security ACL) is deny, then there is no need to consider any ACEs in the second feature ACL (policy routing) that match the same packets, so those ACEs can be removed.

A final optimization consists of creating “blocks” of order-dependent VMRs (ACEs) in the TCAM and then reorganizing these blocks in an order-independent manner. This optimization can further increase TCAM mask sharing and thereby conserve valuable TCAM resources. For more information about the importance of TCAM mask sharing, see the “TCAM Mask Sharing” section on page 15.

Obviously, none of these optimizations will occur unless the final lookup result (permit or deny) will still be returned correctly after the optimizations are complete. If the optimizations would result in an incorrect result, the ACL feature manager will always install the ACEs in question so that the correct result is returned.

Ternary CAM

The Ternary CAM (TCAM) is a specialized piece of memory designed for rapid table lookups, based on packets passing through the switch, performed by the ACL engine on the PFC, PFC2, and PFC3. The result of the ACL engine lookup into the TCAM determines how the switch handles a packet. For example, the packet might be permitted or denied.

Figure 3
A Typical Value, Mask, Result (VMR) Entry

```
Mask  [0x7f9e]: 00000000 ffffffff 00000000 05ff0000 20
Value [0x7f9e]: 00000000 0a010164 00000000 04000000 00
Result: 0006
```

The TCAM has a limited number of entries that are populated with pattern values and mask values, each with an associated result. These are known as Value, Mask, Result entries, or VMRs. The term VMR simply refers to the format in which ACEs are represented in the PFC/PFC2/PFC3 TCAM. Figure 3 illustrates a typical VMR (represented as hexadecimal digits).

The "Value" in VMR refers to the pattern that is to be matched (such as IP addresses, protocol ports, and so on). The "Mask" refers to the mask bits associated with the pattern. The "Result" refers to the result or action that occurs in the case where a lookup returns a hit for the pattern and mask. This result might be a simple "permit" or "deny," or it might be a pointer to other more complex information (for example, in policy based routing [PBR], the result is a pointer to an entry in the hardware Adjacency table that contains information about the next-hop).

The ACL feature manager produces the set of VMRs that are installed in the TCAM. In some cases, the ACEs you configure may be subjected to the ACL merge algorithm during this process (see the "ACL Merge Algorithms" section on page 5 for more information). For a given VLAN in a given direction (input or output), a lookup in the TCAM will produce one or more results based on the longest-match hit. Therefore, the entries in the TCAM must either be arranged in a specific order, or must be represented in an order-independent manner.

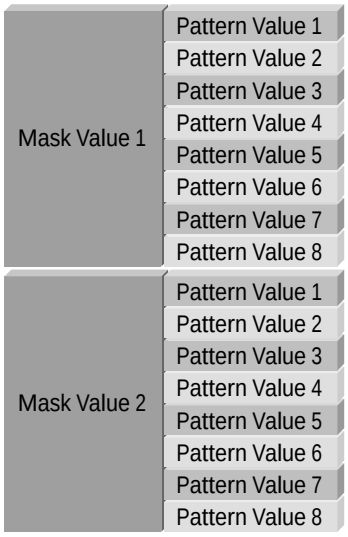
In the case of the order independent merge (OIM), the VMRs are represented as order-independent entries. In the case of the order dependent merge (ODM), the VMRs are arranged such that the first hit represents the longest match hit. It is important to note that the order-independent or order-dependent arrangement of the TCAM entries does not affect the ultimate permit or deny result for a particular packet. The proper permit or deny result as configured in the access lists will be returned by a TCAM lookup.

Installing VMRs in the TCAM

For a single mask-value entry in the TCAM, there are eight associated entries that can be populated with pattern values. The mask-value and pattern-value entries contain the VMRs generated by the ACL feature manager. The VMRs may directly correspond to user-configured access control entries (ACEs, that is, the individual lines of an access list), to VMRs generated in the process of the ACL merge algorithm, or to both.

Figure 4 illustrates the relationship between masks and patterns in the TCAM.

Figure 4
TCAM Masks and Patterns



The following discussion is oversimplified, but provides the basic idea of how the ACL TCAM entries are populated with VMRs.

After the ACL feature manager and merge have produced a list of VMRs for the configured ACLs, the VMRs are installed in the TCAM. Each specific pattern value is associated with a mask value:

- Each pattern value includes information (explicit and implicit) specified in the ACE, such as VLAN, source and destination IP addresses, protocol IDs, port numbers, IP and TCP flag settings, and so forth
- The mask value with which the pattern is associated specifies which bits in the pattern should be matched exactly (mask bits are set) and which bits we don't care about (mask bits are not set)

Up to eight patterns can be associated with a single mask. If a ninth pattern using the same mask needs to be installed, a new mask is populated (setting the same match and don't care bits as the first mask), and that pattern becomes the first pattern associated with the second mask.

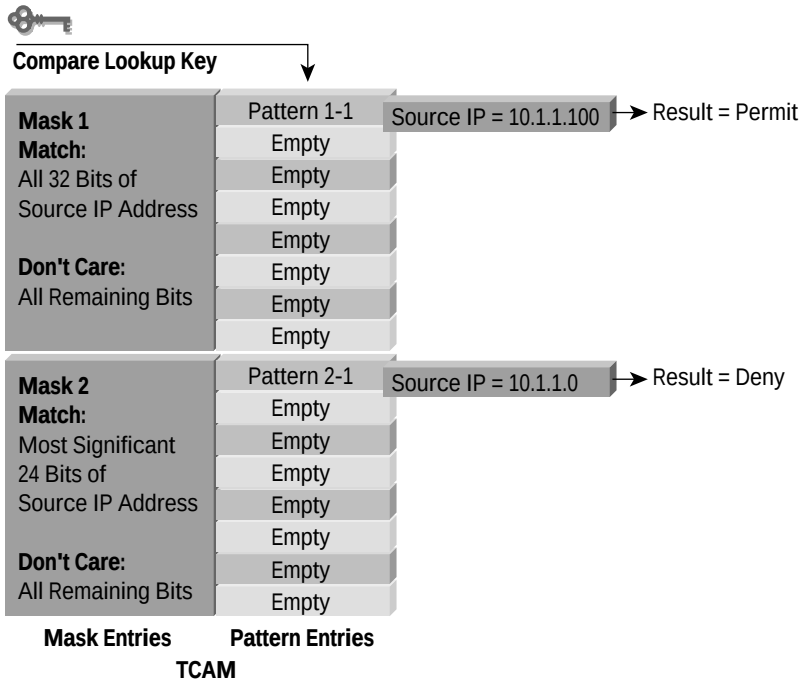
For example, consider these example ACEs from a VACL configuration (assume that there are other ACEs in this VACL that we are not considering):

```
set security acl ip Control_Access permit host 10.1.1.100
set security acl ip Control_Access deny 10.1.1.0 255.255.255.0
```

From these two ACEs, two VMRs are generated (assuming the ACL merge is not invoked, and no feature manager optimizations result in one of the entries being removed—events that are outside the scope of this discussion). When these VMRs are installed in the TCAM, two masks and two patterns are consumed.

Figure 5 shows how the VMRs are stored in the TCAM for the two ACEs in this VACL.

Figure 5
ACL VMRs Installed in the TCAM for a VACL (Example Part 1)



The first mask (Mask 1) has 32 match bits that correspond to the source IP address. One pattern is associated with the mask—Pattern 1-1, using 10.1.1.100 as the source IP. The remaining mask bits are don't care bits, corresponding to the destination IP address, port numbers, etc. This pattern points to a result for a hit, in this case, permit.

The second mask (Mask 2) has 24 match bits and eight don't care bits over the source IP address. The associated pattern (Pattern 2-1) has 10.1.1.0 as the source IP. Similar to the first mask, this mask also has the remaining mask bits flagged as don't care bits. This pattern points to a result for a hit, in this case, deny.

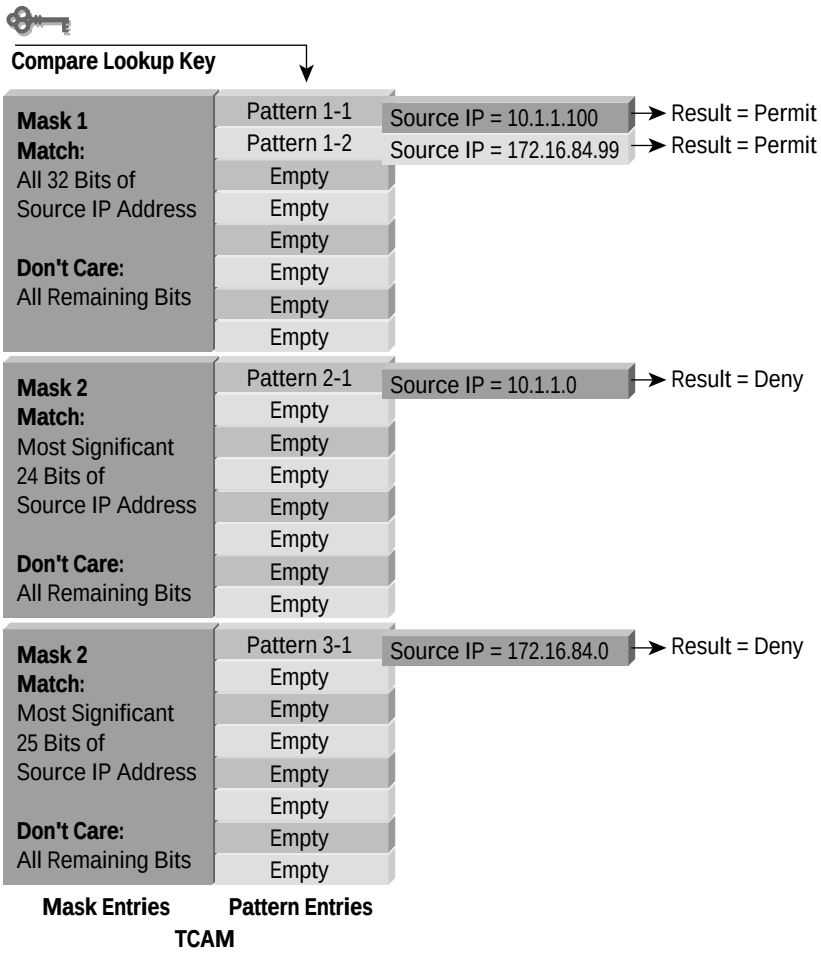
Now, suppose we add two new ACEs (indicated in bold type) to the VACL:

```
set security acl ip Control_Access permit host 10.1.1.100
set security acl ip Control_Access deny 10.1.1.0 255.255.255.0
set security acl ip Control_Access permit host 172.16.84.99

set security acl ip Control_Access deny 172.16.84.0 255.255.255.128
```

These new entries consume two additional pattern value entries and one additional mask value entry in the TCAM.

Figure 6 shows how the masks and patterns are stored in the TCAM for the additional ACEs in this VACL.

Figure 6
TCAM Mask and Patterns for VACL (Example Part 2)



The first new entry uses the same mask (Mask 1) as the first entry in the VACL (that is, match the 32-bit source IP in the lookup key, don't care for the rest). Therefore, a second pattern (Pattern 1-2) is populated for that mask.

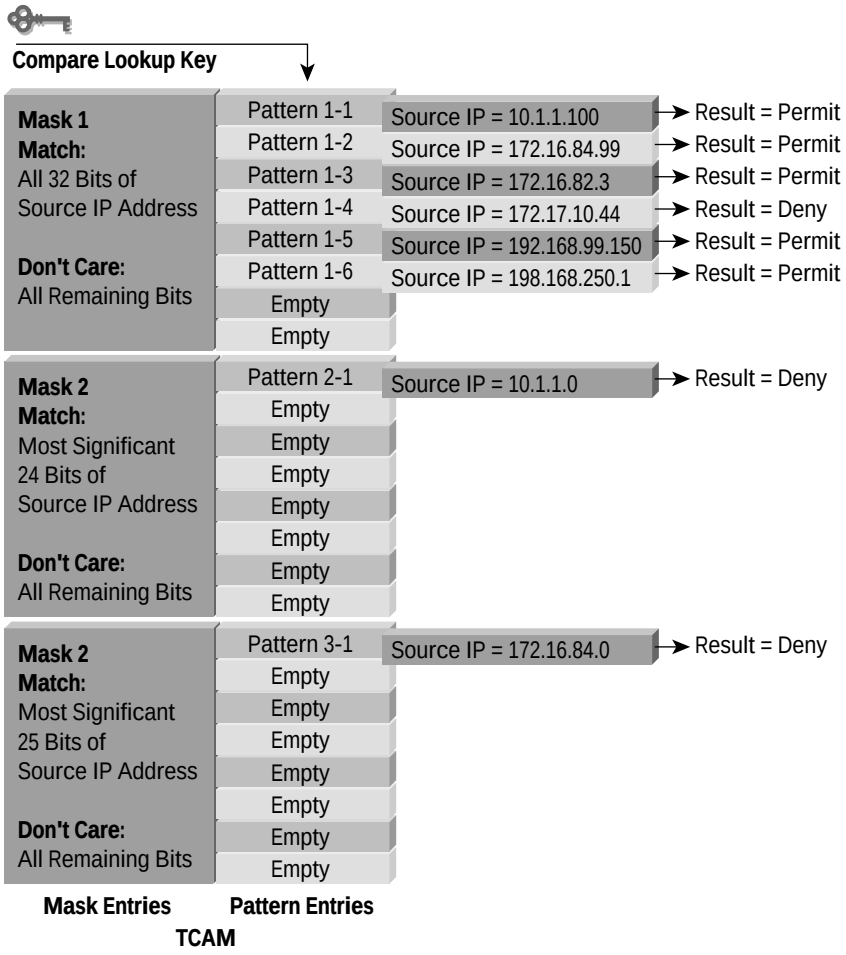
The second new entry requires a new mask (match 25 bits of the source IP address, don't care for the rest) and therefore, a new mask (Mask 3) and one of its associated patterns (Pattern 3-1) are populated.

Now, suppose we add four more ACEs (indicated in bold type) to the VACL:

```
set security acl ip Control_Access permit host 10.1.1.100
set security acl ip Control_Access deny 10.1.1.0 255.255.255.0
set security acl ip Control_Access permit host 172.16.84.99
set security acl ip Control_Access deny 172.16.84.0 255.255.255.128
set security acl ip Control_Access permit host 172.16.82.3

set security acl ip Control_Access deny host 172.17.10.44

set security acl ip Control_Access permit host 192.168.99.150

set security acl ip Control_Access deny host 192.168.250.1
```

Figure 7 shows how the masks and patterns are stored in the TCAM for the additional ACEs in this VACL.

Figure 7
TCAM Mask and Patterns for VACL (Example Part 3)



Notice that the four new entries use the same mask already used by the first and third ACEs in the VACL (Mask 1, match the 32-bit source IP address, don't care for the rest). Therefore, these four new patterns (Patterns 1-3, 1-4, 1-5, and 1-6 can be associated with the existing mask.

TCAM Mask Sharing

The VACL example shown in the preceding section ("Installing VMRs in the TCAM," beginning on page 11) demonstrates why sharing of masks among ACE patterns directly affects the total number of ACEs that can fit in the TCAM. With the ACEs in the example VACL, six of the entries share the same mask, while the other two require their own mask, resulting in a total of three masks and eight patterns in the TCAM.

Recall that each mask can have up to eight associated patterns. If, for example, there were 12 patterns associated with a single mask, 2 masks would be consumed, 1 for the first 8 patterns and 1 for the remaining 4 patterns (with 4 unused patterns remaining).

It is also important to recall that the ACL feature manager and merge algorithm produce a set of VMRs derived from the ACLs configured in the system. These VMRs may or may not have a one-to-one correspondence with the actual ACEs in the configured ACLs. This is an important point because it is the VMRs, not the ACEs themselves, which are installed in the TCAM.

In the most inefficient mask sharing case, where no mask sharing is possible, the total number of VMRs (not necessarily ACEs) that can fit in the TCAM is equal to the total number of masks in the TCAM. In the most efficient mask sharing case, the total number of VMRs that can fit in the TCAM is equal to the total number of patterns in the TCAM.

On the Supervisor Engine 1a with PFC, with the most inefficient mask sharing, 2K VMRs could fit in the TCAM. With the most efficient mask sharing, a total of 16K VMRs could fit in the TCAM.

On the Supervisor Engine 2 with PFC2 and the Supervisor 720 with PFC3, with the most inefficient mask sharing, 4K VMRs could fit in the TCAM for security ACLs and the same number (4K) could fit in the TCAM for QoS ACLs. With the most efficient mask sharing, a total of 32K VMRs for security ACLs and 32K VMRs for QoS ACLs could fit in the TCAM.

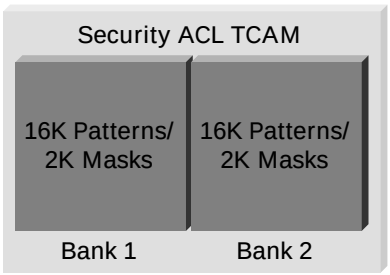
Note: Recall that on PFC2/PFC3, the security ACL and QoS ACL VMRs are installed in different TCAMs, and therefore the masks and patterns are not shared.

Typically, you will be able to fit a number of VMRs in the TCAM somewhere between the total masks and the total patterns. For example, on PFC, perhaps 8K VMRs might fit in the TCAM, depending on how efficiently masks are shared.

Supervisor 720 Dual-Bank Security ACL TCAM Architecture

The security ACL TCAM architecture on the Supervisor 720 with PFC3 is more sophisticated than on earlier supervisor engine generations. The PFC3 implements a dual-bank TCAM for security and feature ACLs consisting of two TCAMs of 16K patterns and 2K masks each. Figure 8 illustrates the dual-bank architecture.

Figure 8
Supervisor 720 with PFC3 Dual-Bank Security ACL TCAM Architecture



Unlike earlier supervisor engines, Supervisor 720 with PFC3 can perform two parallel security ACL lookups and return two independent results in each direction (ingress and egress). Therefore, these dual TCAM banks can be treated as separate TCAMs containing different feature ACLs and returning different results.

The final result is returned using priority bits for each ACE. For example, suppose that a security ACL (such as a RACL) is stored in Bank 1 and a NAT ACL is stored in Bank 2. Stated generally, the priority bits are programmed in such a way that the NAT ACL result is returned only if the RACL result first permits the traffic. If the RACL returns a deny result, the priority bit for that result is set, overriding the parallel lookup result from the NAT ACL.

Optionally, the two banks can be chained together and treated as a single large TCAM bank in cases where a single feature ACL requires more than 16K patterns or 2K masks.

The advantage of the dual-bank architecture is that the ACL merge can be completely avoided in some cases. For example, if you configure two ingress ACL features (such as a RACL and NAT), the VMRs for the RACL can be installed in one TCAM bank and the VMRs for NAT can be installed in the other.

Avoiding the ACL merge has several advantages:

- Reduces the likelihood of ACL merge blowup
- Increases the likelihood that multi-feature configurations can be applied in hardware
- Avoids the increased CPU and memory usage associated with merging ACL features

However, there are certain cases where a merge is still required:

- More than two ACL features are applied to an interface in the same direction. In this case, VMRs for one feature (for example, a VACL) are installed in the first TCAM bank, but the other two features (for example, PBR and NAT) must be merged before installing them in the second TCAM bank.
- One or more interfaces require TCAM-bank chaining. In this case, the dual TCAM banks are treated as a single TCAM and therefore the merge is triggered for any interface that has more than one ACL feature applied.
- If a security ACL feature (such as PBR) uses multi-statement route-maps or otherwise uses multiple ACLs to classify traffic. In this case, all of the route-map ACLs must be merged before installing them in the hardware.

TCAM Lookup Process

When the ACL engine needs to perform a lookup on a packet, it creates a lookup key. The lookup key contains the same type of information as the patterns in the TCAM (IP addresses, ports, and so on) but is based on the contents of the packet passing through the switch.

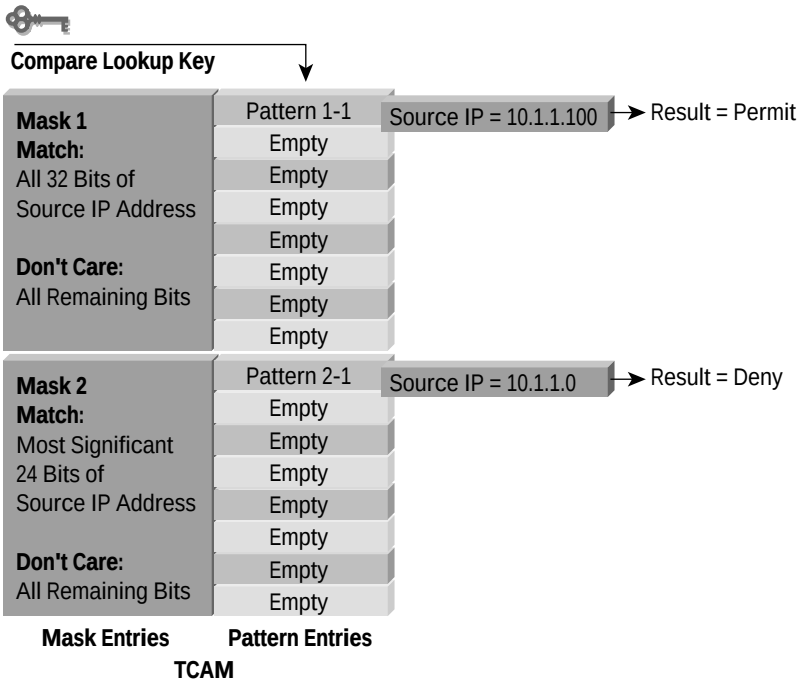
The ACL engine scans the patterns in the TCAM in parallel to see if the lookup key matches. The key matches if the appropriate bits in the key match all the bits in the pattern that are masked with a match bit (using the mask associated with that pattern). Bits that are masked with a don't care bit are ignored—bits in the lookup key do not have to match those bits. The result returned is always the longest match in the TCAM.

Consider this example VACL, given earlier in the "Installing VMRs in the TCAM" section on page 12 (Figure 4):

```
set security acl ip Control_Access permit host 10.1.1.100
set security acl ip Control_Access deny 10.1.1.0 255.255.255.0
```

Now suppose a TCP packet with source IP 10.1.1.100/255.255.255.0 and destination IP 192.168.16.100/255.255.255.0 enters the switch and requires ACL enforcement for this VACL. The ACL engine generates a lookup key based on the packet contents and initiates a TCAM lookup using this key.

Figure 9
VMRs Installed in the TCAM for a VACL



The ACL engine compares the lookup key to all of the patterns in the TCAM in parallel, masking the appropriate bits according to the mask associated with each pattern. In this example, the lookup key will match the first pattern (Pattern 1-1) and return a hit (we assume there are no more specific entries in the TCAM). The result for a hit on this entry is permit. This lookup returned a hit because every bit in the source IP matches exactly. How the remaining bits in the pattern are set does not matter (recall that the first pattern is masked with 32 match bits over the source IP, with the remaining bits set as don't care).

Now, suppose another TCP packet with source IP 10.1.1.55/255.255.255.0 and destination IP 192.168.20.12/255.255.255.0 enters the switch and requires ACL enforcement for the same VACL. Again, the ACL engine generates a lookup key based on the packet contents and initiates a TCAM lookup.

The ACL engine compares this new lookup key to all of the patterns in the TCAM in parallel, with the associated mask bits applied for each pattern. In this example, the 32-bit source IP address of the lookup key does not match Pattern 1-1 so no hit occurs (recall that the first pattern is masked with 32 match bits over the source IP, with the remaining bits set as don't care).

Now, the ACL engine compares the lookup key to the second pattern (Pattern 2-1) using the associated mask (recall that the mask for the second pattern has 24 match bits over the source IP, with the remaining bits of the mask set to "don't care"). The first 24 bits of the source IP in the lookup key (10.1.1) match the 24 bits in the second pattern that are masked with match bits. The result for a hit on this entry is deny and the packet is dropped.

The total number of mask and pattern value entries that can be stored in the TCAM depends on your hardware configuration, as shown in Table 3.

Table 3 TCAM Masks and Patterms by Surpervisor Engine

Supervisor Model	TCAM Masks	TCAM Patterns
Supervisor 1a with PFC	• 2K masks, shared between security ACLs, feature ACLs, and QoS ACLs	• 16K patterns shared between security ACLs, feature ACLs, and Qos ACLs
Supervisor 2 with PFC2 **Supervisor 720 with PFC3**	• 4K masks for security ACLs and feature ACLs • 4K masks for QoS ACLs	• 32K patterns for security ACLs and feature ACLs • 32K patterns for QoS ACLs

The *actual* number of patterns that are populated in the TCAM depends on how efficiently the mask values are shared among entries. Consider the example given previously. The VMRs for the two VACL ACEs consumed one mask and one pattern each, for a total of two masks and two patterns. Each pattern required a new mask because the number and location of match bits and don't care bits in the mask value were different for each pattern (32 match bits versus 24 match bits over the source IP).

Because hardware resources are limited, the Cisco Catalyst 6500 series switches support a mechanism to give one of three priority values to each interface to which an ACL feature configuration is applied. The **`tcam priority`** {**`high|medium|low`**} interface configuration command allows you to specify which interfaces have the highest priority when the system attempts to install a security ACL configuration in the hardware.

Monitoring TCAM and Other Hardware Resource Usage

You can check what percentage of the TCAM mask and patterns are currently populated by issuing one of the commands shown in Table 4, depending on the operating system running on your Cisco Catalyst 6500 switch.

Table 4 TCAM Masks and Patterns by Supervisor Engine

Supervisor Model	Hybrid System (Cisco Catalyst OS + IOS)	Native System (IOS Only)
Supervisor 1a and PFC	• `show security acl resource-usage` or `show qos acl resource-usage` (both show the same values)	• `show tcam counts`
Supervisor 2 and PFC2 **Supervisor 720 with PFC3**	• `show security acl resource-usage` (for security ACLs and feature ACLs) • `show qos acl resource-usage` (for QoS ACLs)	• `show tcam counts`

Use the **show security acl resource-usage** and **show qos acl resource-usage** commands on a hybrid system:

```
6509> (enable) show security acl resource-usage

Security ACL resource usage:
ACL storage (mask/value): 5.21%/1.9%
ACL to switch interface mapping table: 0.98%
ACL layer 4 port operators: 26.56%
6509> (enable) show qos acl resource-usage

QoS ACL resource usage:
ACL storage (mask/value): 0.2%/0.0%
ACL to switch interface mapping table: 0.98%
ACL layer 4 port operators: 26.56%
6509> (enable)
```

Use the **show tcam counts** command on a Supervisor IOS system (on a system with Supervisor 1a with PFC, the QOS TCAM section is not included in the output because QoS ACLs, security ACLs, and feature ACLs all share the same TCAM entries):

```
6509-IOS#show tcam counts

           Used            Free            Percent  Used
           ----            ----            -------------
 Labels:      3             509                0

ACL_TCAM
  Masks:     33            4063                0
Entries:    164           32604                0

QOS_TCAM
  Masks:      1            4095                0
Entries:      1           32767                0

    LOU:     10              54               15
 ANDOR:      0              16                0
 ORAND:      0              16                0

6509-IOS#
```

If the TCAM is full and you attempt to add new ACLs, or ACEs to existing ACLs, the commit or map process will fail, and any prior configuration will remain in effect. In the case of RACLs, the ACL will be enforced in software on the MSFC, with the corresponding performance penalty.

On a switch running hybrid software, if you configure VACL or QoS ACL ACEs that exceed the pattern or mask capacity of the TCAM, a syslog message similar to the following will be printed to the console:

%ACL-5-TCAMFULL: acl engine TCAM table is full

On Supervisor IOS systems, or on the MSFC in a hybrid system, if you configure RACL ACEs that exceed the capacity of the TCAM, a syslog message similar to the following will be printed to the console:

%FM-4-TCAM_ENTRY: Hardware TCAM entry capacity exceeded

On Supervisor IOS systems, or on the MSFC in a hybrid system, issue the show fm summary command to see which interfaces are enforcing ACLs in hardware (ACTIVE) and which are enforcing ACLs in software (INACTIVE).

Note: "fm" stands for Feature Manager, the subsystem responsible for programming ACLs in the PFC/PFC2 hardware.

```
6509-IOS#show fm summary
Interface: GigabitEthernet7/16
  TCAM screening for features is ACTIVE outbound
  TCAM screening for features is ACTIVE inbound
```

Note: If a RACL configuration exceeds the hardware capacity of the PFC/PFC2/PFC3, the output of the **show security acl resource-usage** or **show tcam counts** commands will not show that the TCAM capacity is exceeded. This is because when the VMR installation into the TCAM fails, the VMRs for that ACL are completely removed from the TCAM and the traffic is switched in software.

Logical Operation Units and Layer 4 Operations

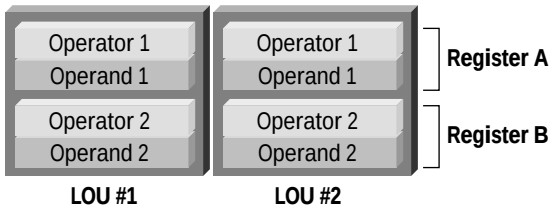
LOUs are hardware registers used to store {operator, operand} tuples for Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) port numbers specified in an IP extended ACL, VACL, or QoS ACL. These tuples are called Layer 4 Operations, or L4Ops.

The operator portion of an L4Op is one of the **lt, gt, neq,** and **range** operators. The operand is the source or destination TCP or UDP port number.

L4Ops consume LOU registers. As an example, the L4Op {gt, 1023} consumes one register (one half) of an LOU. The same rule applies for the **lt** and **neq** operators. An L4Op using the **range** operator requires two LOU registers (the entire LOU). Tuples using the **eq** operator (for example, {eq, 5000}) do not consume LOU registers. Figure 10 illustrates the structure of the LOUs.

Note: In addition to the **lt, gt, neq,** and **range** operators, an ACE matching on TCP flags also consumes an L4Op for the ACL. However, these ACEs do *not* consume LOU registers. Instead, there is a separate hardware table used for storing various TCP flag combinations specified in ACEs. However, because most ACLs do not match on TCP flags other than the "established" flag, exhausting this particular hardware resource is uncommon, and the details of this resource are not discussed here. The important thing to note is, one of the L4Ops for the ACL is consumed if one or more ACEs matches on the TCP "established" flag.

Figure 10
LOU Structure and {operator, operand} Tuples



Consider the following ACL:

```
access-list 101 permit tcp host 10.1.1.1 host 10.2.2.2 gt 1023

access-list 101 permit tcp host 10.3.3.3 lt 1023 host 10.4.4.4
access-list 101 permit tcp host 10.5.5.5 host 10.6.6.6 gt 5000

access-list 101 permit tcp host 10.7.7.7 host 10.8.8.8 neq 2000

access-list 101 permit tcp host 10.9.9.9 lt 1023 host 10.10.10.10 gt 1023
```

There are four different L4Ops in this ACL:

- gt 1023
- lt 1023
- gt 5000
- neq 2000

L4Ops are different if the operation (for example, **gt**, **lt**, etc.) is different, or the operand (the TCP or UDP port number) is different, or both. Therefore, {gt, 1000} and {gt 1001} are two different L4Ops.

For example, consider the following ACL:

```
access-list 101 permit tcp host 10.3.3.3 lt 1023 host 10.4.4.4
access-list 101 permit tcp host 10.5.5.5 host 10.6.6.6 lt 1023
```

If the same {operator, operand} tuple is applied to a source TCP or UDP port and then later to a destination TCP or UDP port, it counts as a different L4Op. In this case, the two {lt, 1023} tuples count as two different L4Ops because one applies to an IP source address and one applies to an IP destination address.

Now consider the following ACL:

```
access-list 101 permit tcp host 10.1.1.1 host 10.2.2.2 gt 1023

access-list 101 deny tcp host 10.3.3.3 gt 1023 host 10.4.4.4
access-list 101 deny tcp host 10.5.5.5 neq 4000 host 10.6.6.6
access-list 101 permit tcp host 10.7.7.7 host 10.8.8.8 range 5000 6000
```

There are four L4Ops in this ACL. The LOU register consumption for this ACL depends on the supervisor engine version.

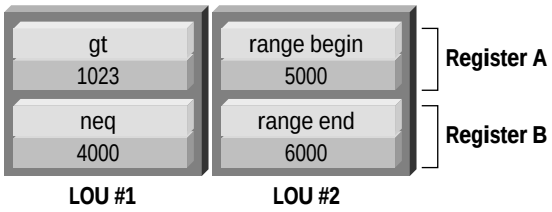
With Supervisor 1a with PFC and Supervisor 2 with PFC2, LOUs are shared between source ports and destination ports. Unlike with L4Ops, the same {operator, operand} tuple applied to both a source address and a destination address only consumes one LOU register (half of one LOU). In the case of the range operator, two LOU registers are consumed (one full LOU).

In the preceding example:

- Register A of the first LOU contains {gt, 1023}
- Register B of the first LOU contains {neq, 4000}
- The entire second LOU (both registers) contains {range, 5000 6000}

Figure 11 illustrates how the LOU registers in this example are populated on Supervisor 1a with PFC and Supervisor 2 with PFC2.

Figure 11
LOU Registers Example 1—Supervisor 1a with PFC and Supervisor with PFC2



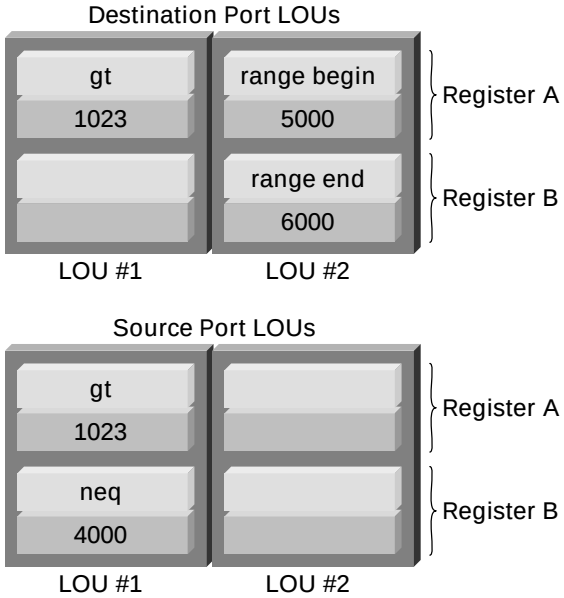
With Supervisor 720 with PFC3, LOUs are *not* shared between source ports and destination ports. Therefore, the same {operator, operand} tuple applied to both a source address and a destination address consumes two LOU registers (half of one LOU each). In the case of the range operator, the same range applied to both a source and destination address consumes four LOU registers (two full LOUs).

In the preceding example:

- Register A of the first source port LOU contains {gt, 1023}
- Register B of the first source port LOU contains {neq, 4000}
- Register A of the first destination port LOU contains {gt, 1023}
- The entire second destination LOU (both Register A and Register B) contains {range, 5000 6000}

Register B of the first destination port LOU is empty and is available for another destination port **`lt, gt, or neq`** L4Op. Figure 12 illustrates how the LOU registers in this example are populated on Supervisor 720 with PFC3.

Figure 12
LOU Registers Example 1—Supervisor 720 with PFC3



To summarize, consider the following ACL we used earlier:

```
access-list 101 deny tcp host 10.3.3.3 lt 1023 host 10.4.4.4
access-list 101 deny tcp host 10.5.5.5 host 10.6.6.6 lt 1023
```

In this example, on Supervisor 1a with PFC and Supervisor 2 with PFC2, there are two different L4Ops, but only one LOU register (half of one LOU) is consumed. On Supervisor 720 with PFC3, there are two different L4Ops, and one source port LOU register and one destination port LOU register are consumed.

There are 32 LOUs (64 registers) available on Supervisor 1a with PFC and Supervisor 2 with PFC2 systems. There are 64 LOUs (128 registers) available on Supervisor 720 with PFC3; 32 LOUs are reserved for source ports and 32 LOUs are reserved for destination ports. All ACLs in the system use the same pool of LOUs, and as described earlier, different ACLs using the same L4Ops can share the same LOU registers.

There is a limit of nine L4Ops for a given ACL in Supervisor 1a with PFC systems. For Supervisor Engine 2 with PFC2 and Supervisor 720 with PFC3 systems, there is a limit of ten L4Ops for a given ACL. This is because there are a limited number of LOU pointers available per ACL. If more than nine/ten L4Ops are configured, subsequent L4Ops must be expanded, which can greatly increase the TCAM usage in the PFC/PFC2.

Note: In certain configurations, the number of L4Ops supported per ACL is reduced to seven/eight per ACL due to internal constructs for handling IP packet fragments.

In simple terms, when an L4Op is expanded, the system installs multiple VMRs in the TCAM that are equivalent to a single VMR that uses a LOU pointer. The system attempts to make the least possible impact when expanding L4Ops (that is, it will try to expand ACEs that require the least number of expanded TCAM entries). In the worst case, an expanded ACE can consume a large number of TCAM pattern and masks.

When using more than nine/ten L4Ops in an ACL, keep a careful eye on the TCAM usage statistics using the **show security acl resource-usage** or **show tcam counts** commands. These commands show the total LOU usage for both security ACLs (RACLs and VACLs) and QoS ACLs.

Note: The **show qos acl resource-usage** command shows the same usage percentage for LOUs as the **show security acl resource-usage** because LOUs are shared between security and QoS ACLs.

Consider the following VACL, which uses nine L4Ops (this poorly written VACL is used merely to illustrate the effect of L4Op expansion):

```
set security acl ip CA deny tcp host 10.1.1.10 host 20.1.1.10 neq 1042
set security acl ip CA deny tcp host 10.1.1.10 host 20.1.1.10 neq 1043
set security acl ip CA deny tcp host 10.1.1.10 host 20.1.1.10 neq 1044
set security acl ip CA deny tcp host 10.1.1.10 host 20.1.1.10 neq 1045
set security acl ip CA deny tcp host 10.1.1.10 host 20.1.1.10 neq 1046
set security acl ip CA deny tcp host 10.1.1.10 host 20.1.1.10 neq 1047
set security acl ip CA deny tcp host 10.1.1.10 host 20.1.1.10 neq 1048
set security acl ip CA deny tcp host 10.1.1.10 host 20.1.1.10 neq 1049
set security acl ip CA deny tcp host 10.1.1.10 host 20.1.1.10 neq 1050
set security acl ip CA permit any
```

Here is the output of the **show security acl resource-usage** for this VACL:

```
6509> (enable) show security acl usage-usage

Security ACL resource usage:
ACL storage (mask/value): 0.53%/0.6%
ACL to switch interface mapping table: 0.39%
ACL layer 4 port operators: 14.6%
6509> (enable)
```

Notice that 0.53 percent of the mask values, 0.6 percent of the pattern values, and 14.6 percent of the LOUs are currently utilized.

Note: Do not be confused by the label "ACL layer 4 port operators"—this value represents the percentage of LOUs used system-wide, not L4Ops.

Now, suppose we add this ACE (using a tenth L4Op) to the VACL before the permit any parameter:

```
set security acl ip CA deny tcp host 10.1.1.10 host 20.1.1.10 neq 2000
```

Because we have exceeded the limit of nine L4Ops per ACL, the switch must expand the L4Op into multiple TCAM entries. Here is the output of the **show security acl usage-usage** command for this VACL after the expansion:

```
6509> (enable)  show security acl usage-usage
Security ACL resource usage:
ACL storage (mask/value): 1.32%/0.16%
ACL to switch interface mapping table: 0.39%
ACL layer 4 port operators: 14.6%
```

Adding this ACE to the VACL more than doubles the mask and pattern usage (when using the BDD merge algorithm). This is because the system must convert the single ACE into multiple pattern and masks in the TCAM to return the correct result but without using an L4Op.

On the switch, if you configure VACLs or QoS ACLs that exhaust the LOU registers in the switch, a syslog message similar to the following is printed to the console:

%ACL-5-NOLOU: acl engine is out of logical operation unit

On the MSFC, or on a Cisco IOS Software on the Supervisor system, if you configure ACLs that exhaust the LOU registers in the switch, a syslog message similar to the following is printed to the console:

%FM-4-TCAM_CAPMAP: Interface Vlan10 hardware TCAM LOU usage capability exceeded

Estimating L4Op and LOU Usage

This section shows you how to estimate LOU and L4Op usage when configuring ACLs. Due to a limited number of LOUs available to the system (32 or 64 LOUs system-wide), and because the number of L4Ops you can use on a per-ACL basis (nine/ten per ACL without ACE expansion) is limited, in some cases it might be important to estimate the total LOU and L4Op usage in your configuration, especially for larger and more complex ACL configurations.

Note: These examples consider the worst-case values. In many situations, the ACL feature manager and the ACL merge algorithm can optimize the individual ACEs and VMRs as they are prepared for installation in the TCAM. See the "ACL Feature Manager and ACL Optimizations" section on page 10 for more information.

Consider the following RACLs (access-list 101 and access-list 102):

```
access-list 101 permit tcp host 10.1.1.1 host 10.2.2.2 gt 1023

access-list 101 deny   tcp host 10.3.3.3 host 10.4.4.4 lt 1023

access-list 101 deny   tcp host 10.5.5.5 host 10.6.6.6 gt 5000

access-list 101 permit tcp host 10.7.7.7 host 10.8.8.8 neq 4000

access-list 101 permit tcp host 10.9.9.9 neq 4000 host 10.10.10.10
access-list 101 deny   tcp host 10.11.11.11 host 10.12.12.12 gt 1023


access-list 102 deny   tcp host 10.1.1.1 host 10.2.2.2 gt 2400

access-list 102 deny   tcp host 10.3.3.3 lt 1023 host 10.4.4.4
access-list 102 permit tcp host 10.5.5.5 range 1200 1300 host 10.6.6.6
access-list 102 permit tcp host 10.7.7.7 host 10.8.8.8 neq 4000
```

The L4Op usage for these ACLs is as follows:

- ACL 101—five L4Ops
- ACL 102—four L4Ops

Note: ACL 101 has five L4Ops because the first and last ACEs specify the same L4Op ({gt, 1023} for the destination TCP port).

On Supervisor 1a with PFC and Supervisor 2 with PFC2, the LOU usage for these ACLs is 3.5 (7 LOU registers). On Supervisor 720 with PFC3, the LOU usage for these ACLs is 4.5 (9 LOU registers). Figure 13 illustrates how the LOU registers in this example are populated in Supervisor 1a with PFC and Supervisor 2 with PFC2. Figure 14 illustrates how the LOU registers are populated in Supervisor 720 with PFC3.

Figure 13
LOU Registers Example 2—Supervisor 1a with PFC and Supervisor 2 with PFC2

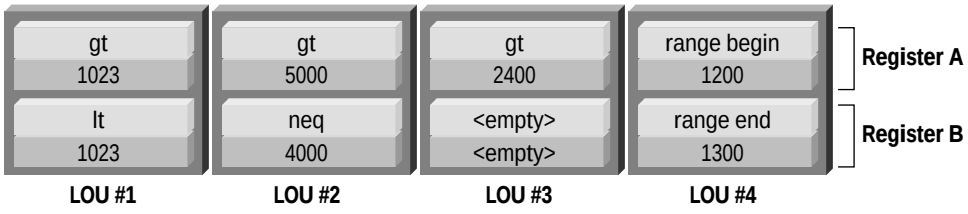
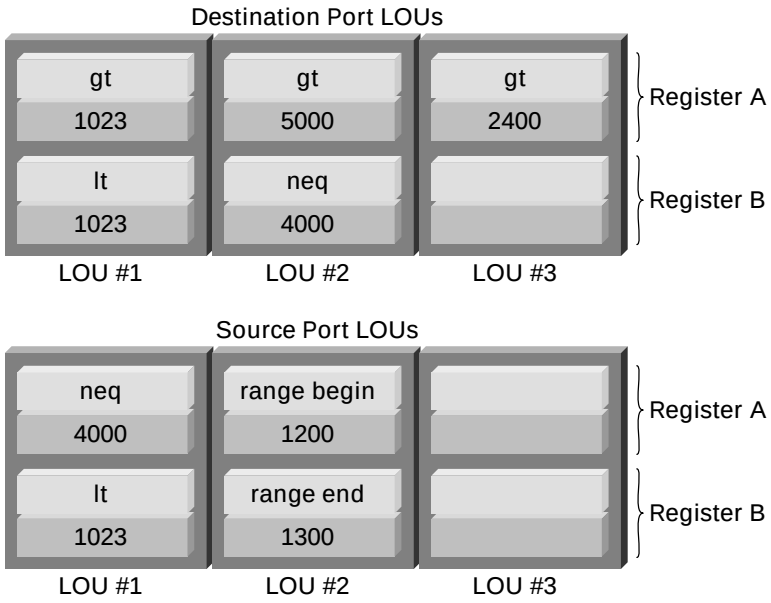


Figure 14
LOU Registers Example 2—Supervisor 720 with PFC3



Recall that there is a limit of 9/10 L4Ops per ACL, and a system-wide limit of 32 or 64 LOUs. In some cases, you might want to replace ACEs that use L4Ops with several ACEs that use the **eq** operator instead, in order to save L4Ops and LOU registers.

For example, consider this ACE:

```
access-list 199 permit tcp any any range 5500 5502
```

This ACE could be configured as follows:

```
access-list 199 permit tcp any any eq 5500
access-list 199 permit tcp any any eq 5501
access-list 199 permit tcp any any eq 5502
```

Doing so saves one L4Op (LOU pointer) for access-list 199 and one LOU (recall that the range operator uses two LOU registers, consuming one full LOU).

We do not recommend expanding L4Ops manually if there are many ports in the range because the TCAM compiler will probably expand the L4Op more efficiently. For example, do not enter 100 ACEs using the eq operator to replace an ACE using range 1100 1200 because the ACL feature manager can expand this ACE into far fewer than 100 ACEs using efficient TCAM mask and pattern values. As a general rule, do not manually expand ranges unless they have ten or fewer ports.

ACL to Switch Interface Mapping Labels

The ACL-to-switch interface mapping table, like TCAM entries, L4Ops, and LOUs, is another limited hardware resource used for ACL processing. This table represents the total number of ACLs you can apply to interfaces, VLANs, or switch ports system wide.

The Cisco Catalyst 6500 Series supervisor engines support up to 512 unique unidirectional ACL labels system-wide, each of which consumes an entry in the mapping table when it is applied to an interface, VLAN, or switch port. There are three types of labels: input, output, and port.

It is important not to confuse the 512 ACL label limit with a limit on the total number of ACLs supported in the system. The total labels and the total supported ACLs are not necessarily the same thing. The following discussion describes the details of how ACL labels are allocated and how they relate to the total number of ACLs the system can support.

Each label type is allocated based on how the ACL or ACLs are mapped. The 512 available labels can be any combination of these label types. The pool of available labels is shared between RACLs, VACLs, and QoS ACLs.

The rules of label consumption are somewhat complicated, but can be summarized in the following manner:

- An input RACL or other input feature ACL on a MSFC interface consumes one input label
- An output RACL or other output feature ACL on a MSFC interface consumes one output label
- A VACL or group of VACLs mapped to a VLAN consumes one input and one output label
- A QoS ACL mapped to a VLAN consumes one input label
- A Qos ACL mapped to a port consumes one port label

These rules are, however, over-simplified—in the list above, some labels might be shared between RACLs, VACLs, and QoS ACLs. The following example will better illustrate the manner in which ACL labels are allocated.

Note: The 512 ACL label limit exists in both hybrid and Cisco IOS Software on the Supervisor systems. The following output examples use the **show security acl resource-usage** command available in hybrid mode. To check the label usage in a Cisco IOS Software on the Supervisor system, issue the **show tcam counts** command.

Suppose you configure a single VACL, VACL-A, and map it to VLAN 1. This consumes two labels, one input, and one output, which we will call Label In-1 and Label Out-1, respectively. Both labels contain "VACL-A." Checking the resource usage, we can see that a percentage of the total label pool has been consumed:

```
6509> (enable) show security acl map VACL-A

ACL VACL-A is mapped to VLANs:
1
6509> (enable) show security acl resource-usage

Security ACL resource usage:
ACL storage (mask/value): 3.32%/0.41%
ACL to switch interface mapping table: 0.39% ‹  Label usage is 0.39%
ACL layer 4 port operators: 1.56%
6509> (enable)
```

Now, suppose that you map the same VACL to VLANs 2-1000. The same two labels (Label In-1 and Label Out-1) are used for all of the VLANs, so the resource consumption remains the same:

```
6509> (enable) show security acl map VACL-A

ACL VACL-A is mapped to VLANs:
1-1000
6509> (enable) show security acl resource-usage

Security ACL resource usage:
ACL storage (mask/value): 3.32%/0.41%
ACL to switch interface mapping table: 0.39% <  Label usage remains at 0.39%
ACL layer 4 port operators: 1.56%
6509> (enable)
```

Now, we configure input IP ACL 103 on the VLAN 10 interface of the MSFC (**ip access-group 103 in**). This consumes a new input ACL label (we will refer to it as Label In-2), containing input RACL 103 and VACL-A. We have now consumed three labels from the pool (Label In-1, Label In-2, and Label Out-1). We can see that the resource usage percentage has increased:

```
6509> (enable) show security acl resource-usage

Security ACL resource usage:
ACL storage (mask/value): 3.57%/0.64%
ACL to switch interface mapping table: 0.58% <  Label usage is now 0.58%
ACL layer 4 port operators: 26.56%
6509> (enable)
```

Now, suppose we apply input IP ACL 103 on the VLAN 20, 30, and 40 interfaces of the MSFC. Notice that the applied ACLs are the same as those applied to VLAN 10 (input RACL 103 and VACL-A). Therefore, no new labels are consumed—Label In-2 can be reused.

Now, suppose we configure output IPX ACL 801 on the VLAN 10 and 20 MSFC interfaces. This will consume one additional output label, Label Out-2, containing output IPX RACL 801 and IP VACL-A. The resource usage reflects the additional usage:

```
6509> (enable) show security acl resource-usage

Security ACL resource usage:
ACL storage (mask/value): 5.21%/0.85%
ACL to switch interface mapping table: 0.78% <  Label usage is now 0.78%
ACL layer 4 port operators: 26.56%
6509> (enable)
```

There are now four labels in use in the system, Label In-1, Label In-2, Label Out-1, and Label Out-2.

Note: On Supervisor 2 with PFC2 running Cisco IOS Software, and on the Supervisor 720 with PFC3, IPX ACLs are not supported in the hardware. Therefore, when you apply an IPX ACL, hardware resources are not consumed because IPX access control is handled in software on the MSFC.

Now, suppose we configure a QoS ACL (QOS-ACL-A) that polices traffic (according to an aggregate policer rule) on VLAN 10 and VLAN 20. Doing so consumes a new input ACL label (we will call it Label In-3) containing input IP RACL 103, IP VACL-A, and IP QOS-ACL-A. Recall that the pool of labels is shared between security ACLs and QoS

ACLs on both Supervisor Engine 1a and Supervisor Engine 2. The usage value for the ACL-to-switch interface mapping table shown in the **show security acl resource-usage** and **show qos acl resource-usage** command output is always the same:

```
6509> (enable) show qos acl resource-usage

QoS ACL resource usage:
ACL storage (mask/value): 0.2%/0.0%
ACL to switch interface mapping table: 0.98% ‹  Label usage is now 0.98%
ACL layer 4 port operators: 26.56%
6509> (enable) show security acl resource-usage

Security ACL resource usage:
ACL storage (mask/value): 5.21%/1.9%
ACL to switch interface mapping table: 0.98% ‹  Same usage percentage here
ACL layer 4 port operators: 26.56%
6509> (enable)
```

Finally, suppose we configure port-based QoS on a group of ports (port 3/1-16) by applying a QoS ACL (PORT-QOS-ACL-A) to the ports. This consumes one port-based ACL label containing PORT-QOS-ACL-A.

```
6509> (enable) show qos acl resource-usage

QoS ACL resource usage:
ACL storage (mask/value): 0.7%/0.0%
ACL to switch interface mapping table: 1.17% ‹  Label usage is now 1.17%
ACL layer 4 port operators: 26.56%
6509> (enable)
```

Note: It does not matter to which VLAN the ports belong—the port-based ACL label allocation is independent of the input- and output-based ACL label allocation.

This example illustrates the basic method used for ACL label allocation. In a system with a large number of ACLs applied to interfaces or VLANs, it is important to try to share labels as much as possible, and to monitor the ACL label usage as you apply ACL configuration.

On a system running Cisco Catalyst OS, if you configure VACLs or QoS ACLs that exhaust the ACL labels in the switch, a syslog message similar to the following is printed to the console:

%ACL-5-NOLABEL: acl engine is out of label

On the MSFC or on Cisco IOS Software on the Supervisor system, if you make configuration changes that exhaust the ACL labels in the switch, a syslog message similar to the following is printed to the console:

%FM-4-TCAM_LABEL: Hardware TCAM label capacity exceeded

Policy-Based Routing TCAM and Adjacency Utilization

In Supervisor 2 with PFC2 and Supervisor 720 with PFC3, policy-based routing (PBR) is fully supported in hardware using a combination of the security and feature ACL TCAM and the hardware adjacency table.

While a discussion of PBR is outside the scope of this document, it is important to understand that PBR does consume ACL TCAM resources and adjacency table entries.

When you configure PBR, the PBR ACLs are installed in the security ACL TCAM. When a packet matches a PBR ACE, an adjacency pointer is returned as the result. The pointer references a location in the hardware adjacency table that contains the rewrite information necessary to forward the packet to the correct PBR next hop.

In Supervisor 2 with PFC2, 1024 of the 256 K available hardware adjacencies are reserved for PBR. In Supervisor 720 with PFC3, 2048 of the 1 million available hardware adjacencies are reserved for PBR.

NVRAM Space with QoS ACLs and VACLs

In hybrid systems with very large QoS ACL and VACL configurations on the Supervisor Engine, the NVRAM space may be the gating factor in whether you can apply the desired ACL configuration. For this reason, beginning with Cisco Catalyst OS release 5.4(1), you can choose to store your QoS and VACL configuration in Flash memory rather than in NVRAM.

The NVRAM available on the supervisor engine is adequate for most configurations. However, when NVRAM space is exhausted, the system automatically attempts to store the ACL configuration in Flash memory and sets the CONFIG_FILE variable to automatically load the ACL configuration at boot up.

If desired, you can manually configure the system to store your ACL configuration in Flash. See the “References and Related Information” section on page 31 for detailed information on how ACLs are stored in Flash, and how to configure the switch to store your ACL configuration in Flash.

References and Related Information

This section contains information about other reference documents and related information pertaining to the ACL implementation and configuration on Cisco Catalyst 6500 Series switches.

- Configuring Access Control on Cisco Catalyst 6500 Series Switches—Cisco Catalyst OS 7.6(1)

  http://www.cisco.com/univercd/cc/td/doc/product/lan/cat6000/sw_7_6/confg_gd/acc_list.htm
- Configuring QoS on Cisco Catalyst 6500 Series Switches—Cisco Catalyst OS 7.6(1)

  http://www.cisco.com/univercd/cc/td/doc/product/lan/cat6000/sw_7_6/confg_gd/qos.htm
- Configuring Network Security on Cisco Catalyst 6500 Series Switches—Cisco IOS Software on the Supervisor 12.1E

  http://www.cisco.com/univercd/cc/td/doc/product/lan/cat6000/12_1e/swconfig/secure.htm
- Configuring Network Security on Cisco Catalyst 6500 Series Switches—Cisco IOS Software on the Supervisor 12.2SX

  http://www.cisco.com/univercd/cc/td/doc/product/lan/cat6000/122sx/swcg/secure.htm
- Configuring QoS on Cisco Catalyst 6500 Series Switches—Cisco IOS Software on the Supervisor 12.1E

  http://www.cisco.com/univercd/cc/td/doc/product/lan/cat6000/12_1e/swconfig/qos.htm
- Configuring QoS on Cisco Catalyst 6500 Series Switches—Cisco IOS Software on the Supervisor 12.2SX

  http://www.cisco.com/univercd/cc/td/doc/product/lan/cat6000/122sx/swcg/qos.htm
- Cisco Catalyst 6000 Family Release Notes

  http://www.cisco.com/univercd/cc/td/doc/product/lan/cat6000/relnotes/index.htm

Glossary

ACE—Access Control Entry (one line of an Access Control List)

ACL—Access Control List (an access list in IOS, or a security or QoS access list in Cisco Catalyst OS)

BDD—Binary Decision Diagram (a method for representing ACEs using binary trees and used to perform the ACL merge when using the OIM)

Feature ACL—Feature Access Control List (any ACL used to match traffic in order to apply or not apply a configured feature to the traffic—such features include NAT, PBR, access control, and so on)

Feature manager—The software responsible for converting individual ACEs into the actual VMRs that will be installed in the TCAM, and for performing ACL optimizations

L4Op—Layer 4 Operation (the {operator, operand} tuples used to perform Layer 4 protocol operations—the L4Ops are lt, gt, neq, and range)

LOU—Logical Operation Unit (hardware register in the PFC/PFC2 used to store L4Op information)

ODM—Order Dependent Merge (one of two ACL merge algorithms available in Cisco Catalyst 6500 switches, characterized by more efficient merge results)

OIM—Order Independent Merge (one of the two ACL merge algorithms available in Cisco Catalyst 6500 switches, producing order independent TCAM entries but at the cost of using more unique mask values)

PBR—Policy-based routing (a mechanism for making a routing decision based on the source rather than the destination IP address of a packet)

QoS ACL—QoS Access Control List (in Cisco Catalyst OS, an ACL for QoS configured using the set qos acl commands)

RACL—Router Access Control List (an access-list applied to an interface in IOS, using the ip access-group {in|out} command)

TCAM—Ternary Content Addressable Memory (specialized piece of memory for storing complex tabular data and supporting very rapid parallel lookups)

uRPF check—Unicast Reverse Path Forwarding check (method of verifying the reachability of source IP addresses in incoming packets)

VACL—VLAN Access Control List (a security ACL mapped to a VLAN on a Cisco Catalyst OS switch, using the set security acl commands)

VMR—Value, Mask, Result (format, used to represent entries in the TCAM, that consists of a pattern value, the associated mask value, and a result for lookups returning a hit for the entry)

Packets requiring ICMP unreachables are leaked at ten packets per second per VLAN (hybrid) or one packet every two seconds per VLAN (native)

Corporate Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 526-4100

European Headquarters
Cisco Systems International BV
Haarlerbergpark
Haarlerbergweg 13-19
1101 CH Amsterdam
The Netherlands
www-europe.cisco.com
Tel: 31 0 20 357 1000
Fax: 31 0 20 357 1100

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
www.cisco.com
Tel: 408 526-7660
Fax: 408 527-0883

Asia Pacific Headquarters
Cisco Systems, Inc.
Capital Tower
168 Robinson Road
#22-01 to #29-01
Singapore 068912
www.cisco.com
Tel: +65 6317 7777
Fax: +65 6317 7799

Cisco Systems has more than 200 offices in the following countries and regions. Addresses, phone numbers, and fax numbers are listed on the

Cisco Web site at www.cisco.com/go/offices

Argentina • Australia • Austria • Belgium • Brazil • Bulgaria • Canada • Chile • China PRC • Colombia • Costa Rica • Croatia Czech Republic • Denmark • Dubai, UAE • Finland • France • Germany • Greece • Hong Kong SAR • Hungary • India • Indonesia • Ireland Israel • Italy • Japan • Korea • Luxembourg • Malaysia • Mexico • The Netherlands • New Zealand • Norway • Peru • Philippines • Poland Portugal • Puerto Rico • Romania • Russia • Saudi Arabia • Scotland • Singapore • Slovakia • Slovenia • South Africa • Spain • Sweden Switzerland • Taiwan • Thailand • Turkey • Ukraine • United Kingdom • United States • Venezuela • Vietnam • Zimbabwe

EL/LW4761 09/03