Control Plane Policing Implementation Best Practices

Contents

Introduction: Network Device Operations

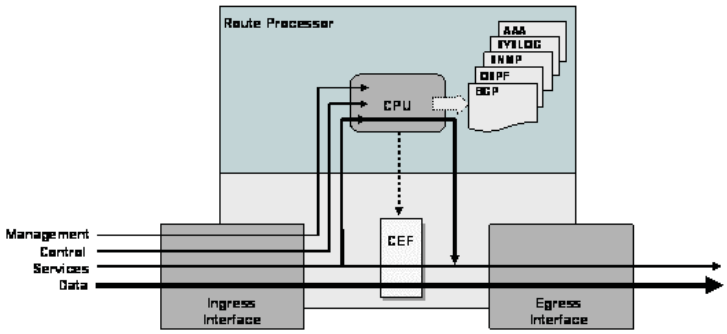
IP networks provide users with connectivity to networked resources such as corporate servers, extranet partners, multimedia content, the Internet, and any other application envisioned within IP networks. While these networks function to carry data plane (user-generated) packets, they are also created and operated by control plane and management plane packets. Unlike legacy network technologies such as ISDN, Frame Relay, and ATM that defined separate data and control channels, IP carries all packets within a single pipe. Thus, IP network devices such as routers and switches must be able to distinguish between data plane, control plane, and management plane packets to treat each packet appropriately.

From an IP traffic plane perspective, packets may be divided into four distinct, logical groups:

1. Data plane packets – End-station, user-generated packets that are always forwarded by network devices to other end-station devices. From the perspective of the network device, data plane packets always have a transit destination IP address and can be handled by normal, destination IP address-based forwarding processes.
2. Control plane packets – Network device generated or received packets that are used for the creation and operation of the network itself. From the perspective of the network device, control plane packets always have a receive destination IP address and are handled by the CPU in the network device route processor. Examples include protocols such as ARP, BGP, OSPF, and other protocols that glue the network together.
3. Management plane packets – Network device generated or received packets, or management station generated or received packets that are used to manage the network. From the perspective of the network device, management plane packets always have a receive destination IP address and are handled by the CPU in the network device route processor. Examples include protocols such as Telnet, Secure Shell (SSH), TFTP, SNMP, FTP, NTP, and other protocols used to manage the device and/or network.
4. Services plane packets – A special case of data plane packets, services plane packets are also user-generated packets that are also forwarded by network devices to other end-station devices, but that require high-touch handling by the network device (above and beyond normal, destination IP address-based forwarding) to forward the packet. Examples of high-touch handling include such functions as GRE encapsulation, QoS, MPLS VPNs, and SSL/IPsec encryption/decryption, etc. From the perspective of the network device, services plane packets may have a transit destination IP address, or may have a receive destination IP address (for example, in the case of a VPN tunnel endpoint).

Each of these types represents a specific group of packets that a network device will receive on ingress from network interfaces and be required to process. This concept is illustrated in Figure 1.

Figure 1. IP Network Traffic Planes and Their Handling within the Router



From the local perspective of the network device, three general types of packets exist:

1. Transit packets – These include data plane and some services plane packets that are subjected to standard, destination IP-based forwarding functions. In most networks and under normal operating conditions, transit packets are typically forwarded by Cisco Express Forwarding mechanisms, either in the interrupt process within CPU-based (software switched) platforms, or within specialized high-speed forwarding hardware (ASICs, FPGAs, or NPs) on high-end platforms. “Fast path” is most often used to describe this type of packet handling.
2. Receive packets – These include control plane and management plane packets that are destined to the network device itself. Receive packets must be handled by the CPU within the route processor, as they are ultimately destined to and handled by applications running at the process level within IOS or IOS XR. “Punt” is often used to describe the action of moving a packet from the fast path to the route processor for handling.
3. Exception IP and Non-IP packets – One special set of packets includes both exception IP packets and non-IP packets. Exception IP packets include, for example, IPv4 packets containing IP header options, IP packet TTL expires, and IP packets with unreachable destinations. Layer 2 keepalives, ISIS packets, Cisco Discovery Protocol (CDP) packets, and PPP Link Control Protocol (LCP) packets are examples of non-IP packets. All of the packets in this set must be handled by the route processor.

Under normal network operating conditions, the vast majority of packets handled by network devices are data plane packets. These packets are handled in the fast path. Network devices are optimized to handle these fast path packets efficiently. Typically, considerably fewer control and management plane packets are required to create and operate IP networks. Thus, the punt path and route processor are significantly less capable of handling the kinds of packets rates experienced in the

fast path since they are never directly involved in the forwarding of data plane packets.

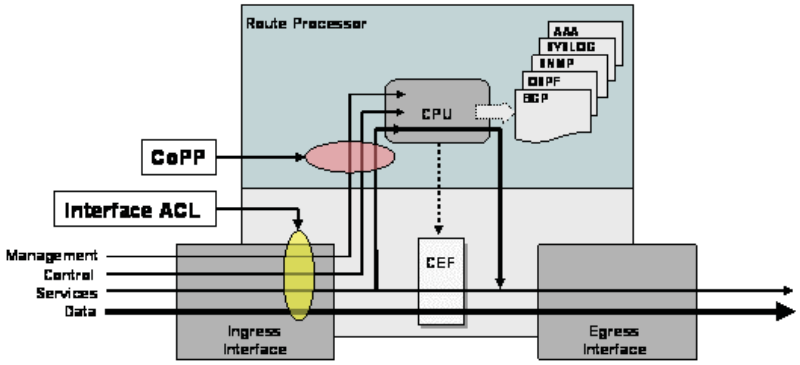
When high packet rates overload the control and/or management plane, route processor resources can be overwhelmed, reducing the availability of these resources for tasks critical to the operation and maintenance of the network. For example, if a high volume of rogue packets generated by a virus or worm is presented to the control plane, the router will spend an excessive amount of time processing and discarding unnecessary traffic. The route processor is thus not available to support its required tasks such as computing periodic routing table updates, maintaining the CEF table, which is actually used for the forwarding of data plane packets, and maintaining interface link states (see Figure 1).

When considering route processor attack scenarios, it is natural to think only of malicious events. However, both malicious and non-malicious events can overwhelm route processor resources. Malicious events include crafted packet attacks or simply high rates of packets directed at the control plane. Non-malicious events may result from router or network misconfigurations, software bugs, or in some circumstances, network failure reconvergence events. From the perspective of the router, the underlying condition is irrelevant when the result is the same. It is important to take appropriate steps to protect the route processor from being overwhelmed, whether by malicious or non-malicious events.

Many Cisco IOS™ and Cisco IOS XR software security features are available to protect the route processor of networking devices. Some of these features are generic and applicable to a broad range of security functions. Others are specifically designed to protect the route processor. In brief, important features include the following:

- Interface ACL – The interface access control list (iACL) is the traditional and most generally available approach for managing all packets entering or exiting a network device. The iACLs are well understood and are generally applicable to data, services, control, and management plane packets. However, as illustrated in Figure 2, iACLs are applied at the interface level to each packet ingressing (or egressing) the interface—not just control plane packets, for example. In addition, iACLs must be applied to every individual interface to which the policy is to be applied. On large routers, this can be an onerous task.
- Receive ACL. The receive path ACL (rACL) feature was developed for Cisco 7500 and Cisco 12000 Series routers as the initial step toward achieving a Cisco IOS-wide route processor protection mechanism. These routers are widely deployed in very large service provider networks, can have an extensive number of interfaces, and can see very high packet rates. Thus, the rACL concept was developed as an efficient mechanism to protect the route processor on these high-end routers. Unlike iACLs, which are applied on individual interfaces, rACLs are applied once to the receive path of the router. In this way, only a single instance of the rACL must be configured and applied only to those packets with receive destination IP addresses. This covers most (but not all) packets that may punt and require handling by the route processor. In addition, as with all ACLs, rACLs only provide permit and deny granularity in their actions. (Considering Figure 2, rACLs operate where Control Plane Policing (CoPP) is shown in the figure, but only on IP packets with a receive destination IP address.)
- Control Plane Policing – CoPP is the Cisco IOS-wide route processor protection mechanism. As illustrated in Figure 2, and similar to rACLs, CoPP is deployed once to the punt path of the router. However, unlike rACLs that only apply to receive destination IP packets, CoPP applies to all packets that punt to the route processor for handling. CoPP therefore covers not only receive destination IP packets, it also exceptions IP packets and non-IP packets. In addition, CoPP is implemented using the Modular QoS CLI (MQC) framework for policy construction. In this way, in addition to simply permit and deny functions, specific packets may be permitted but rate-limited. This behavior substantially improves the ability to define an effective CoPP policy. (Note: that “Control Plane Policing” is something of a misnomer because CoPP generally protects the punt path to the route processor and not solely the control plane.)
- Local Packet Transport Services – Cisco IOS XR is the newest Cisco router operating system and is designed for the CRS-1 and XR 12000 routers. The Local Packet Transport Services (LPTS) feature is only implemented in Cisco IOS XR and takes the Cisco IOS CoPP concept to a new level by automatically applying rate limiting to all packets that must be handled by any route processor on the device. LPTS maintains tables describing all packet flows destined for the route processor and uses a set of predefined policers that are applied based on the flow type of the incoming control traffic. Automated control is vital to network health as it becomes increasingly difficult to rely on network engineers and operators to configure these functions manually in such large-scale, high-speed networks.

Figure 2. The Impact of Interface ACLs and Control Plane Policing Mechanism on Each IP Network Traffic Plane



These features and others are all part of the Cisco Network Foundation Protection (NFP), which is an umbrella strategy encompassing Cisco IOS and Cisco IOS XR software security features that provide the tools, technologies, and services that enable organizations to secure their network foundations. NFP helps to establish a methodical approach to protecting router planes, forming the foundation for continuous service delivery. Cisco IOS and Cisco IOS XR software both provide a rich set of security features to address complexity of attacks and help ensure the availability of network elements under any circumstances.

The remainder of this white paper provides deployment and operational guidance for the Cisco IOS CoPP feature. A short overview for comparison purposes is also provided for the Cisco IOS XR LPTS feature.

Control Plane Policing Feature Description

General Overview

Control Plane Policing (CoPP) is a Cisco IOS-wide feature designed to allow users to manage the flow of traffic handled by the route processor of their network devices. CoPP is designed to prevent unnecessary traffic from overwhelming the route processor that, if left unabated, could affect system performance. Route processor resource exhaustion, in this case, refers to all resources associated with the punt path and route processor(s) such as Cisco IOS process memory and buffers, and ingress packet queues.

More than just control plane packets can punt and affect the route processor and system resources. Management plane traffic, as well as certain data plane exceptions IP packets and some services plane packets, may also require the use of route processor resources. Even so, it is common practice to identify the resources associated with the punt path and route processor(s) as the Control Plane. The feature in Cisco IOS is CoPP.

CoPP protects the route processor on network devices by treating route processor resources as a separate entity with its own ingress interface (and in some implementations, egress also). Because of this behavior, a CoPP policy can be developed and applied only to those packets within the control plane. Unlike interface ACLs, for example, no effort is wasted investigating data plane (transit) packets that will never reach the control plane. This action has a significant simplifying implication on the construction of policies for CoPP.

CoPP is implemented using the Cisco IOS Modular QoS CLI (MQC), a highly flexible framework that allows users to create and attach traffic polices to interfaces The Cisco Modular QoS CLI (MQC) mechanisms are used by CoPP to define the classification and policing descriptions for its policies. In this way, in addition to the limited permit and deny actions associated with simple ACLs, specific packets may be permitted but rate-limited when using the MQC structure. For example, you may wish to permit certain ICMP packet types, but rate limit them so that the route processor is not adversely impacted. This action adds tremendously to the capabilities and flexibility of developing and deploying a useable CoPP policy.

In general, the configuration information presented here represents the basic set of CoPP functions that should be available across all platforms. Where appropriate, references will be made to areas that may have different behavior or enhanced feature availability. The remainder of this section covers the operational aspects involved with successful CoPP policy construction and deployment.

CoPP Policy Construction and Deployment Concepts

Before describing the details of CoPP policy construction and deployment, some of the important details related to MQC and its operation, especially within the context of CoPP are discussed.

class.

The instruction for evaluating match commands is specified as either **match-any** or **match-all**. When more than one match statement is included, match-any requires that a packet match at least one of the statements to be included in the class. If match-all is used, a packet must match all of the statements to be included in the class.

The **policy-map** command is used to associate a traffic class, defined by the **class-map** command, with one or more QoS policies. The result of this association is called a service policy. A service policy contains three elements: a name, a traffic class (specified with the **class** command), and the QoS policies. The purpose of the service policy is to associate a traffic class with one or more QoS policies. Classes included within policy maps are processed top-down. When a packet is found to match a class, no further processing is performed. That is, a packet can only belong to a single class, and it is the first one to which a match occurs. When a packet does not match any of the defined classes, it is automatically placed in the class **class-default**. The default class is always applied, whether it is explicitly configured or not.

The **service-policy** command is used to attach the service policy, as specified with the **policy-map** command, to an interface. In the case of CoPP, this is the control-plane interface. Because the elements of the service policy can be applied to packets entering, or in some versions of CoPP, leaving the interface, users are required to specify whether the service policy characteristics should be applied to incoming or outgoing packets.

It is important to note that MQC is a general framework used for enabling all QoS throughout Cisco IOS, and not exclusively for CoPP. Not all features available within the MQC framework are available or applicable to CoPP policies. For example, only certain classification (match) criteria are applicable to CoPP. In some instances, there are MQC platform and/or IOS-dependencies that may apply to CoPP. Consult the appropriate product references and configuration guides for any CoPP-specific dependencies.

1. Constructing the CoPP Policy
2. Deploying the CoPP Policy
3. Verifying the CoPP Policy
4. Tuning the CoPP Policy

Each of these steps is discussed in detail below. For each step, guidance based on deployment experience is provided.

1. Constructing the CoPP Policy
For CoPP policy construction, several steps are required to create the MQC classification and policing functions. These include: **access-list** construction, **class-map** construction, and finally, **policy-map** construction.

Access List Construction

To define appropriate policies for your CoPP configuration, you need to identify all of the traffic flows and packet rates for those flows that may be seen by CoPP. Typically, ACLs are used for the traffic flow identification task and, in most cases, the protocols as well as the source and destination IP addresses are well known. It is still quite likely that some surprise traffic flows will arise. The definition of these ACLs is one of the most critical steps in the CoPP process. MQC uses these ACLs to define the traffic classes, which in turn become the object of the policy actions (policing). Appropriate granularity in the distribution of protocols within these ACLs allows for better protection of the RP.

By recognizing that certain classes will be created, you will see that it is important to define distinct, granular access lists that will be used to define the CoPP classes. Some traffic types are easy; BGP is critical, for example. Other traffic types may not be as straightforward. The typical approach involves grouping traffic flows based on function within the network. Such an approach may include classes:

- Routing: Routing protocol traffic is crucial to the creation and operation of the network. BGP (TCP/179), OSPF, and EIGRP are all potential candidates. LDP (TCP/646) and MSDP (TCP/639) may also be required (implementation specific). (IS-IS cannot be classified, however, as it is not an IP protocol).
- Management: Management protocols necessary for the day-to-day operation of the network includes protocols such as: SSH (TCP/22), HTTP/HTTPS (TCP/80 and TCP/443), TFTP (UDP/69), SNMP (UDP/161), NTP (UDP/123), DNS (UDP/53), etc.
- Normal: Other less essential traffic can also be expected to be required, but would be prudent to rate-limit. This class mainly includes ICMP protocol packets such as ICMP Time-Exceeded, Echo-Request, Echo-Reply, Packet-too-big, Port-unreachable, etc. This class might also include things like GRE traffic, etc.
- Undesirable: This class covers explicitly bad or malicious traffic that should always be denied access to the RP. This includes all IP fragments (fragments should never be seen in the control plane), certain TCP RESET packets, and other specific, identifiable attack packets (such as SQL-Slammer) that may randomly target router receive addresses.
- Catch-All-IP: A catch-all IP class is required to collect any remaining traffic that has not matched any other class and that is destined for the RP. This class prevents these packets from ending up in class-default. (This is discussed in further detail under the **policy-map** discussion).

There are several caveats and key points to keep in mind when constructing your access lists.

- The **log** or **log-input** keywords must never be used in access-lists that are used within MQC policies for CoPP. The use of these keywords may cause unexpected result in the functionality of CoPP.
- The use of the deny rule in access lists used in MQC is somewhat different to regular interface ACLs. Packets that match a deny rule are excluded from that class and cascade to the next class (if one exists) for classification. This is in contrast to packets matching a permit rule, which are then included in that class and no further comparisons are performed.

Based on the above guidance, you can create ACLs to define traffic. Separate ACLs with unique numbers (or names, if allowed) should be used to represent each class. For initial or pilot CoPP deployments, you may wish to be less restrictive on IP source and destination addresses. Over time and with more experience, refinements may be appropriate. Example ACLs for the above categories follow.

Routing – ACL 120

```
!-- ACL for CoPP Routing class-map
!
access-list 120 permit tcp any gt 1024 <router receive block> eq bgp
access-list 120 permit tcp any eq bgp <router receive block> gt 1024 established
access-list 120 permit tcp any gt 1024 <router receive block> eq 639
access-list 120 permit tcp any eq 639 <router receive block> gt 1024 established
access-list 120 permit tcp any <router receive block> eq 646
access-list 120 permit udp any <router receive block> eq 646
access-list 120 permit ospf any <router receive block>
access-list 120 permit ospf any host 224.0.0.5
access-list 120 permit ospf any host 224.0.0.6
access-list 120 permit eigrp any <router receive block>
access-list 120 permit eigrp any host 224.0.0.10
access-list 120 permit udp any any eq pim-auto-rp
---etc--- for other routing protocol traffic...
!
```

Management – ACL 121

```
! – ACL for CoPP Management class
!
access-list 121 permit tcp <NOC block> <router receive block> eq telnet
access-list 121 permit tcp <NOC block> eq telnet <router receive block> established
access-list 121 permit tcp <NOC block> <router receive block> eq 22
access-list 121 permit tcp <NOC block> eq 22 <router receive block> established
access-list 121 permit udp <NOC block> <router receive block> eq snmp
access-list 121 permit tcp <NOC block> <router receive block> eq www
access-list 121 permit udp <NOC block> <router receive block> eq 443
access-list 121 permit tcp <NOC block> <router receive block> eq ftp
access-list 121 permit tcp <NOC block> <router receive block> eq ftp-data
```

```
---etc--- for known good management traffic...
!
```

Normal – ACL 122

```
!-- ACL for CoPP Normal class-map
!
access-list 122 permit icmp any <router receive block> echo
access-list 122 permit icmp any <router receive block> echo-reply
access-list 122 permit icmp any <router receive block> ttl-exceeded
access-list 122 permit icmp any <router receive block> packet-too-big
access-list 122 permit icmp any <router receive block> port-unreachable
access-list 122 permit icmp any <router receive block> unreachable
access-list 122 permit pim any any
access-list 122 permit igmp any any
access-list 122 permit gre any any
---etc--- for other known good traffic...
!
```

Undesirable – ACL 123

```
! -- ACL for CoPP Undesirable class-map
!
access-list 123 permit icmp any any fragments
access-list 123 permit udp any any fragments
access-list 123 permit tcp any any fragments
access-list 123 permit ip any any fragments
access-list 123 permit udp any any eq 1434
access-list 123 permit tcp any any eq 639 rst
access-list 123 permit tcp any any eq bgp rst
--- etc. all other known bad things here–
!
```

Catch-All IP – ACL 124

```
! -- ACL for CoPP Catch-All class-map
!
access-list 124 permit tcp any any
access-list 124 permit udp any any
access-list 124 permit icmp any any
access-list 124 permit ip any any
!
```

The next step in CoPP policy construction is that of class-map construction.

Class Map Construction

In MQC, the **class-map** statement defines the classes by name, and includes one or several match statements that indicate the classification mechanisms to be used to determine which packets are in the class. The **match** keyword supports the following classification mechanisms for CoPP:

- Standard and extended numbered or named IP access lists (ACLs) using the match access-group keyword. (In certain Cisco IOS releases, named ACLs may not be available for CoPP use).
- IP TOS values, including match ip dscp and match ip precedence keywords.
- ARP protocol packets using match protocol arp command. (The Cisco IOS 12.2SX release does not support the match protocol arp command.)

Thus, the syntax used within MQC for creating a CoPP class map is as follows:

```
router(config) class-map [match-any | match-all] class-name
router(config-cmap) match [access-group | protocol | ip prec | ip dscp]
```

Based on the ACL construction phase, the following classes of traffic are defined for the CoPP policy.

```
!
! – CoPP Routing class-map
!
class-map match-all Routing
 match access-group 120
!
! – CoPP Management class-map
!
class-map match-all Management
 match access-group 121
!
! – CoPP Normal class-map
!
class-map match-all Normal
 match access-group 122
!
! – CoPP Undesirable class-map
!
class-map match-all Undesirable
 match access-group 123
!
! – CoPP Catch-All-IP class-map
!
class-map match-all Catch-All-IP
 match access-group 124
!
```

The final step in CoPP policy construction is that of policy-map construction.

Policy Map Construction

In MQC, the **policy-map** statement is used to define a service policy. After defining the service policy using the **policy-map** statement, the **class** command is used within the **policy-map** definition to specify the name of a class, as defined by the **class-map** phase previously, and the traffic policy to be associated with that class. For CoPP deployments, the **police** keyword is typically used to define the traffic policy. Thus, the syntax used within MQC for creating a CoPP policy map is as follows:

```
 exceed-action [transmit | drop]

where:
* cir – Committed information rate (bits per second)
* rate – Policy rate in packets per second (pps)
```

When more than one class of traffic is defined within a policy-map, the order of classes is important, as traffic is compared against successive classes, top-down, until a match is recorded. Once a packet has matched a class, no further comparisons are made. If no match is found after processing all classes, packets automatically match the always-defined class, **class-default**. The class **class-default** is special in MQC because it is always automatically placed at the end of every policy map. Match criteria cannot be configured for **class-default** because it automatically includes an implied match for all packets. Only a traffic policy can be configured for **class-default**.

The following are several caveats and key points to keep in mind when constructing policy maps.

- After the **class** keyword has been used to specify a class of traffic within the policy-map, if the desired policy is to permit all traffic at an unpoliced rate, it is allowable in some versions of IOS to simply omit the police keyword and action. For example:

```
!
policy-map CoPP
 class ONE
 police 10000 1500 1500 conform-action transmit exceed-action transmit
 class TWO
 police 10000 1500 1500 conform-action transmit exceed-action transmit
 class THREE
 police 10000 1500 1500 conform-action transmit exceed-action drop
```

Is equivalent to:

```
!
policy-map CoPP
 class ONE
 class TWO
 class THREE
 police 10000 1500 1500 conform-action transmit exceed-action drop
```

As shown in the example, traffic matching class **ONE** or class **TWO** is permitted without using a **police** statement with each class. Check the release notes for your version of IOS to determine whether this option is available for policing traffic.

- In some versions of IOS, the keyword **drop** may be used in place of the keyword **police** when the desired action is to deny all traffic within the affected class. For example, the following policy drops all traffic matching **class ONE**:

```
!
policy-map CoPP
 class ONE
 police 10000 1500 1500 conform-action drop exceed-action drop
 class TWO
 police 10000 1500 1500 conform-action transmit exceed-action drop
```

The above is equivalent to:

```
!
policy-map CoPP
 class ONE
 drop
 class TWO
 police 10000 1500 1500 conform-action transmit exceed-action drop
```

As shown in the above example, traffic matching **class ONE** is simply dropped. Using the drop **keyword** is equivalent to using the **police** statement with both conform and exceed actions of drop. While the police statement is available in all Cisco IOS releases, the **drop** keyword is only available in certain releases. Check the release notes for your version of Cisco IOS to determine which options are available for policing traffic.

- In most versions of IOS, policing can only be accomplished on a bandwidth basis. That is, policing can only be specified according to a rate measured in bits-per-second and not on packets-per-second. Check the release notes for your version of IOS to determine which options are available for policing traffic.
- The class **class-default** is automatically placed at the end of the policy map. By the nature of CoPP matching mechanisms, certain traffic types will always end up falling into the default class. This includes traffic such as Layer 2 keepalives and non-IP traffic such as certain ISIS packets. Because these traffic types are required to maintain the network control plane, **class-default** must never be policed with both conform and exceed being set with an action of drop. It is generally considered best practice never to rate-limit the class **class-default**.

Based on this, and the class-map construction phase, the following policy-map is defined for this example CoPP policy:

```
!
policy-map RTR_CoPP
 class Undesirable
 police 8000 1500 1500 conform-action drop exceed-action drop
 class Routing
 police 1000000 50000 50000 conform-action transmitexceed-action transmit
 class Management
 police 100000 20000 20000 conform-action transmit exceed-action drop
 class Normal
 police 50000 5000 5000 conform-action transmit exceed-action drop
 class Catch-All-IP
 police 50000 5000 5000 conform-action transmit exceed-action drop
 class class-default
 police 8000 1500 1500 conform-action transmit exceed-action transmit
!
```

Note that in **policy-map RTR_CoPP**, several of the police statements include **drop** actions. This represents the type of policy that could be used in an operationally effective CoPP deployment after careful consideration, lab testing, and/or pilot deployments have determined the policy to be operationally sound. It is typical for an initial policy to be developed for lab and/or pilot deployments that includes **transmit** actions only. Such a policy would be used to assess the effectiveness of ACLs, class-map choices, and **policy-map** traffic rates for each class of traffic. Once the CoPP policy is confirmed to be operationally effective, it would then be appropriate to convert the appropriate “exceed-action” **transmit** actions to **drop** actions.

Now that the CoPP policy has been built, it is must be applied to the control plane interface.

2. Deploying the CoPP Policy
As mentioned above, the CoPP policy is applied to the control plane interface. Only traffic destined for the route processor will be affected by the CoPP policy. The

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.