

Service Policy Using the Modular Policy Framework

Released: April 24, 2014
Updated: September 16, 2014

Service policies using Modular Policy Framework provide a consistent and flexible way to configure ASA features. For example, you can use a service policy to create a timeout configuration that is specific to a particular TCP application, as opposed to one that applies to all TCP applications. A service policy consists of multiple actionsapplied to an interface or applied globally.

This chapter includes the following sections:

- Information About Service Policies, page 1-1
- Licensing Requirements for Service Policies, page 1-6
- Guidelines and Limitations, page 1-6
- Default Settings, page 1-8
- Task Flows for Configuring Service Policies, page 1-9
- Identifying Traffic (Layer 3/4 Class Maps), page 1-12
- Defining Actions (Layer 3/4 Policy Map), page 1-15
- Applying Actions to an Interface (Service Policy), page 1-17
- Monitoring Modular Policy Framework, page 1-18
- Configuration Examples for Modular Policy Framework, page 1-18
- Feature History for Service Policies, page 1-21

Information About Service Policies

This section describes how service policies work and includes the following topics:

- Supported Features, page 1-2
- Feature Directionality, page 1-2
- Feature Matching Within a Service Policy, page 1-3
- Order in Which Multiple Feature Actions are Applied, page 1-4
- Incompatibility of Certain Feature Actions, page 1-5

- Feature Matching for Multiple Service Policies, page 1-6

Supported Features

Table 1-1 lists the features supported by Modular Policy Framework.

Table 1-1 Modular Policy Framework

Feature	For Through Traffic?	For Management Traffic?	See:
Application inspection (multiple types)	**All** except RADIUS accounting	**RADIUS accounting** only	• Chapter 7, "Getting Started with Application Layer Protocol Inspection." • Chapter 8, "Inspection of Basic Internet Protocols." • Chapter 9, "Inspection for Voice and Video Protocols." • Chapter 10, "Inspection of Database and Directory Protocols." • Chapter 11, "Inspection for Management Application Protocols." • Chapter 21, "ASA and Cisco Cloud Web Security."
ASA IPS	**Yes**	No	Chapter 26, "ASA IPS Module."
ASA CX	**Yes**	No	Chapter 25, "ASA CX Module."
ASA FirePOWER (ASA SFR)	**Yes**	No	Chapter 24, "ASA FirePOWER (SFR) Module."
NetFlow Secure Event Logging filtering	**Yes**	**Yes**	See the general operations configuration guide.
QoS input and output policing	**Yes**	No	Chapter 19, "Quality of Service."
QoS standard priority queue	**Yes**	No	Chapter 19, "Quality of Service."
QoS traffic shaping, hierarchical priority queue	**Yes**	**Yes**	Chapter 19, "Quality of Service."
TCP and UDP connection limits and timeouts, and TCP sequence number randomization	**Yes**	**Yes**	Chapter 18, "Connection Settings."
TCP normalization	**Yes**	No	Chapter 18, "Connection Settings."
TCP state bypass	**Yes**	No	Chapter 18, "Connection Settings."
User statistics for Identity Firewall	**Yes**	**Yes**	See the **user-statistics** command in the command reference.

Feature Directionality

Actions are applied to traffic bidirectionally or unidirectionally depending on the feature. For features that are applied bidirectionally, all traffic that enters or exits the interface to which you apply the policy map is affected if the traffic matches the class map for both directions.

Note When you use a global policy, all features are unidirectional; features that are normally bidirectional when applied to a single interface only apply to the ingress of each interface when applied globally. Because the policy is applied to all interfaces, the policy will be applied in both directions so bidirectionality in this case is redundant.

For features that are applied unidirectionally, for example QoS priority queue, only traffic that enters (or exits, depending on the feature) the interface to which you apply the policy map is affected. See Table 1-2 for the directionality of each feature.

Table 1-2 Feature Directionality

Feature	Single Interface Direction	Global Direction
Application inspection (multiple types)	Bidirectional	Ingress
ASA CSC	Bidirectional	Ingress
ASA CX	Bidirectional	Ingress
ASA CX authentication proxy	Ingress	Ingress
ASA FirePOWER (ASA SFR)	Bidirectional	Ingress
ASA IPS	Bidirectional	Ingress
NetFlow Secure Event Logging filtering	N/A	Ingress
QoS input policing	Ingress	Ingress
QoS output policing	Egress	Egress
QoS standard priority queue	Egress	Egress
QoS traffic shaping, hierarchical priority queue	Egress	Egress
TCP and UDP connection limits and timeouts, and TCP sequence number randomization	Bidirectional	Ingress
TCP normalization	Bidirectional	Ingress
TCP state bypass	Bidirectional	Ingress
User statistics for Identity Firewall	Bidirectional	Ingress

Feature Matching Within a Service Policy

See the following information for how a packet matches class maps in a policy map for a given interface:

1. A packet can match only one class map in the policy map for each feature type.
2. When the packet matches a class map for a feature type, the ASA does not attempt to match it to any subsequent class maps for that feature type.
3. If the packet matches a subsequent class map for a different feature type, however, then the ASA also applies the actions for the subsequent class map, if supported. See Incompatibility of Certain Feature Actions, page 1-5 for more information about unsupported combinations.

Note Application inspection includes multiple inspection types, and most are mutually exclusive. For inspections that can be combined, each inspection is considered to be a separate feature.

For example, if a packet matches a class map for connection limits, and also matches a class map for an application inspection, then both actions are applied.

If a packet matches a class map for HTTP inspection, but also matches another class map that includes HTTP inspection, then the second class map actions are not applied.

If a packet matches a class map for HTTP inspection, but also matches another class map that includes FTP inspection, then the second class map actions are not applied because HTTP and FTP inspections cannpt be combined.

If a packet matches a class map for HTTP inspection, but also matches another class map that includes IPv6 inspection, then both actions are applied because the IPv6 inspection can be combined with any other type of inspection.

Order in Which Multiple Feature Actions are Applied

The order in which different types of actions in a policy map are performed is independent of the order in which the actions appear in the policy map.

Note NetFlow Secure Event Logging filtering and User statistics for Identity Firewall are order-independent.

Actions are performed in the following order:

1. QoS input policing
2. TCP normalization, TCP and UDP connection limits and timeouts, TCP sequence number randomization, and TCP state bypass.

Note When a the ASA performs a proxy service (such as AAA or CSC) or it modifies the TCP payload (such as FTP inspection), the TCP normalizer acts in dual mode, where it is applied before and after the proxy or payload modifying service.

3. ASA CSC
4. Application inspections that can be combined with other inspections:
   a. IPv6
   b. IP options
   c. WAAS
5. Application inspections that cannot be combined with other inspections. See Incompatibility of Certain Feature Actions, page 1-5 for more information.
6. ASA IPS
7. ASA CX
8. ASA FirePOWER (ASA SFR)
9. QoS output policing
10. QoS standard priority queue
11. QoS traffic shaping, hierarchical priority queue

Incompatibility of Certain Feature Actions

Some features are not compatible with each other for the same traffic. The following list may not include all incompatibilities; for information about compatibility of each feature, see the chapter or section for your feature:

- You cannot configure QoS priority queueing and QoS policing for the same set of traffic.
- Most inspections should not be combined with another inspection, so the ASA only applies one inspection if you configure multiple inspections for the same traffic. HTTP inspection can be combined with the Cloud Web Security inspection. Other exceptions are listed in the Order in Which Multiple Feature Actions are Applied, page 1-4.
- You cannot configure traffic to be sent to multiple modules, such as the ASA CX and ASA IPS.
- HTTP inspection is not compatible with the ASA CX or the ASA FirePOWER.
- The ASA CX and ASA FirePOWER modules are not compatible with Cloud Web Security.

Note The **match default-inspection-traffic** command, which is used in the default global policy, is a special CLI shortcut to match the default ports for all inspections. When used in a policy map, this class map ensures that the correct inspection is applied to each packet, based on the destination port of the traffic. For example, when UDP traffic for port 69 reaches the ASA, then the ASA applies the TFTP inspection; when TCP traffic for port 21 arrives, then the ASA applies the FTP inspection. So in this case only, you can configure multiple inspections for the same class map. Normally, the ASA does not use the port number to determine which inspection to apply, thus giving you the flexibility to apply inspections to non-standard ports, for example.

This traffic class does not include the default ports for Cloud Web Security inspection (80 and 443).

An example of a misconfiguration is if you configure multiple inspections in the same policy map and do not use the default-inspection-traffic shortcut. In Example 1-1, traffic destined to port 21 is mistakenly configured for both FTP and HTTP inspection. In Example 1-2, traffic destined to port 80 is mistakenly configured for both FTP and HTTP inspection. In both cases of misconfiguration examples, only the FTP inspection is applied, because FTP comes before HTTP in the order of inspections applied.

Example 1-1 Misconfiguration for FTP packets: HTTP Inspection Also Configured

```
class-map ftp
   match port tcp eq 21
class-map http
   match port tcp eq 21 [it should be 80]
policy-map test
   class ftp
     inspect ftp
   class http
     inspect http
```

Example 1-2 Misconfiguration for HTTP packets: FTP Inspection Also Configured

```
class-map ftp
   match port tcp eq 80 [it should be 21]
class-map http
   match port tcp eq 80
policy-map test
   class http
     inspect http
```

```
class ftp
  inspect ftp
```

Feature Matching for Multiple Service Policies

For TCP and UDP traffic (and ICMP when you enable stateful ICMP inspection), service policies operate on traffic flows, and not just individual packets. If traffic is part of an existing connection that matches a feature in a policy on one interface, that traffic flow cannot also match the same feature in a policy on another interface; only the first policy is used.

For example, if HTTP traffic matches a policy on the inside interface to inspect HTTP traffic, and you have a separate policy on the outside interface for HTTP inspection, then that traffic is not also inspected on the egress of the outside interface. Similarly, the return traffic for that connection will not be inspected by the ingress policy of the outside interface, nor by the egress policy of the inside interface.

For traffic that is not treated as a flow, for example ICMP when you do not enable stateful ICMP inspection, returning traffic can match a different policy map on the returning interface. For example, if you configure IPS on the inside and outside interfaces, but the inside policy uses virtual sensor 1 while the outside policy uses virtual sensor 2, then a non-stateful Ping will match virtual sensor 1 outbound, but will match virtual sensor 2 inbound.

Licensing Requirements for Service Policies

Model	License Requirement
ASAv	Standard or Premium License.
All other models	Base License.

Guidelines and Limitations

This section includes the guidelines and limitations for this feature.

Context Mode Guidelines

Supported in single and multiple context mode.

Firewall Mode Guidelines

Supported in routed and transparent firewall mode.

IPv6 Guidelines

Supports IPv6 for the following features:

- Application inspection for DNS, FTP, HTTP, ICMP, ScanSafe, SIP, SMTP, IPsec-pass-thru, and IPv6.
- ASA IPS
- ASA CX
- ASA FirePOWER

- NetFlow Secure Event Logging filtering
- TCP and UDP connection limits and timeouts, TCP sequence number randomization
- TCP normalization
- TCP state bypass
- User statistics for Identity Firewall

Class Map Guidelines

The maximum number of class mapsof all types is 255 in single mode or per context in multiple mode. Class maps include the following types:

- Layer 3/4 class maps (for through traffic and management traffic).
- Inspection class maps
- Regular expression class maps
- **match** commands used directly underneath an inspection policy map

This limit also includes default class maps of all types, limiting user-configured class mapsto approximately 235. See Default Class Maps, page 1-9.

Policy Map Guidelines

See the following guidelines for using policy maps:

- You can only assign one policy map per interface. (However you can create up to 64 policy maps in the configuration.)
- You can apply the same policy map to multiple interfaces.
- You can identify up to 63 Layer 3/4 class maps in a Layer 3/4 policy map.
- For each class map, you can assign multiple actions from one or more feature types, if supported. See Incompatibility of Certain Feature Actions, page 1-5.

Service Policy Guidelines

- Interface service policies take precedence over the global service policy for a given feature. For example, if you have a global policy with FTP inspection, and an interface policy with TCP normalization, then both FTP inspection and TCP normalization are applied to the interface. However, if you have a global policy with FTP inspection, and an interface policy with FTP inspection, then only the interface policy FTP inspection is applied to that interface.
- You can only apply one global policy. For example, you cannot create a global policy that includes feature set 1, and a separate global policy that includes feature set 2. All features must be included in a single policy.
- When you make service policy changes to the configuration, all *new* connections use the new service policy. Existing connections continue to use the policy that was configured at the time of the connection establishment. **show** command output will not include data about the old connections.

  For example, if you remove a QoS service policy from an interface, then re-add a modified version, then the **show service-policy** command only displays QoS counters associated with new connections that match the new service policy; existing connections on the old policy no longer show in the command output.

  To ensure that all connections use the new policy, you need to disconnect the current connections so they can reconnect using the new policy. See the **clear conn** or **clear local-host** commands.

Default Settings

The following topics describe the default settings for Modular Policy Framework:

- Default Configuration, page 1-8
- Default Class Maps, page 1-9

Default Configuration

By default, the configuration includes a policy that matches all default application inspection traffic and applies certain inspections to the traffic on all interfaces (a global policy). Not all inspections are enabled by default. You can only apply one global policy, so if you want to alter the global policy, you need to either edit the default policy or disable it and apply a new one. (An interface policy overrides the global policy for a particular feature.)

The default policy includes the following application inspections:

- DNS
- FTP
- H323 (H225)
- H323 (RAS)
- RSH
- RTSP
- ESMTP
- SQLnet
- Skinny (SCCP)
- SunRPC
- XDMCP
- SIP
- NetBios
- TFTP
- IP Options

The default policy configuration includes the following commands:

```
class-map inspection_default
 match default-inspection-traffic
policy-map type inspect dns preset_dns_map
 parameters
    message-length maximum client auto
    message-length maximum 512
    dns-guard
    protocol-enforcement
    nat-rewrite
policy-map global_policy
 class inspection_default
  inspect dns preset_dns_map
  inspect ftp
  inspect h323 h225 _default_h323_map
  inspect h323 ras _default_h323_map
```

```
  inspect ip-options _default_ip_options_map
  inspect netbios
  inspect rsh
  inspect rtsp
  inspect skinny
  inspect esmtp _default_esmtp_map
  inspect sqlnet
  inspect sunrpc
  inspect tftp
  inspect sip
  inspect xdmcp
service-policy global_policy global
```

Note See Incompatibility of Certain Feature Actions, page 1-5 for more information about the special **match default-inspection-traffic** command used in the default class map.

Default Class Maps

The configuration includes a default Layer 3/4 class map that the ASA uses in the default global policy called default-inspection-traffic; it matches the default inspection traffic. This class, which is used in the default global policy, is a special shortcut to match the default ports for all inspections. When used in a policy, this class ensures that the correct inspection is applied to each packet, based on the destination port of the traffic. For example, when UDP traffic for port 69 reaches the ASA, then the ASA applies the TFTP inspection; when TCP traffic for port 21 arrives, then the ASA applies the FTP inspection. So in this case only, you can configure multiple inspections for the same class map. Normally, the ASA does not use the port number to determine which inspection to apply, thus giving you the flexibility to apply inspections to non-standard ports, for example.

```
class-map inspection_default
 match default-inspection-traffic
```

Another class map that exists in the default configuration is called class-default, and it matches all traffic. This class map appears at the end of all Layer 3/4 policy maps and essentially tells the ASA to not perform any actions on all other traffic. You can use the class-default class if desired, rather than making your own **match any** class map. In fact, some features are only available for class-default, such as QoS traffic shaping.

```
class-map class-default
 match any
```

Task Flows for Configuring Service Policies

This section includes the following topics:

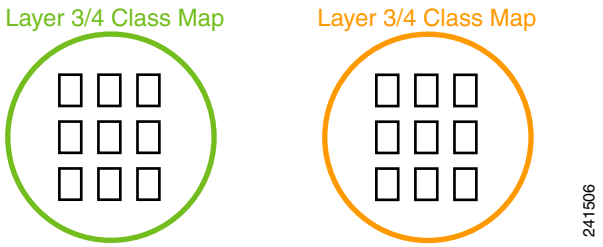
- Task Flow for Using the Modular Policy Framework, page 1-9
- Task Flow for Configuring Hierarchical Policy Maps for QoS Traffic Shaping, page 1-11

Task Flow for Using the Modular Policy Framework

To configure Modular Policy Framework, perform the following steps:

Step 1 Identify the traffic—Identify the traffic on which you want to perform Modular Policy Framework actions by creating Layer 3/4 class maps.

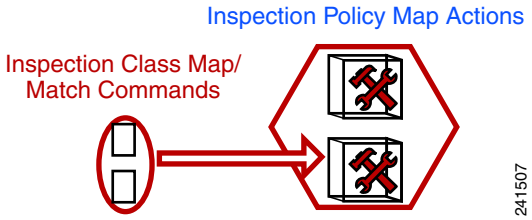
For example, you might want to perform actions on all traffic that passes through the ASA; or you might only want to perform certain actions on traffic from 10.1.1.0/24 to any destination address.



See Identifying Traffic (Layer 3/4 Class Maps), page 1-12.

Step 2 Perform additional actions on some inspection traffic—If one of the actions you want to perform is application inspection, and you want to perform additional actions on some inspection traffic, then create an inspection policy map. The inspection policy map identifies the traffic and specifies what to do with it.

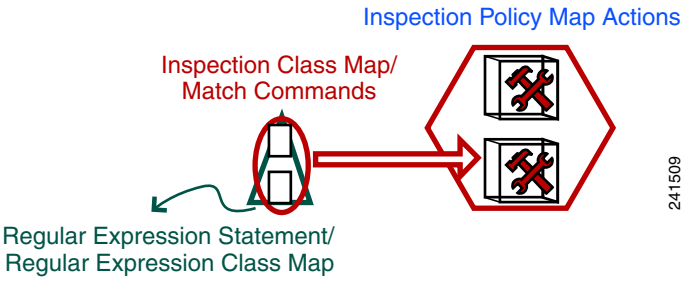
For example, you might want to drop all HTTP requests with a body length greater than 1000 bytes.



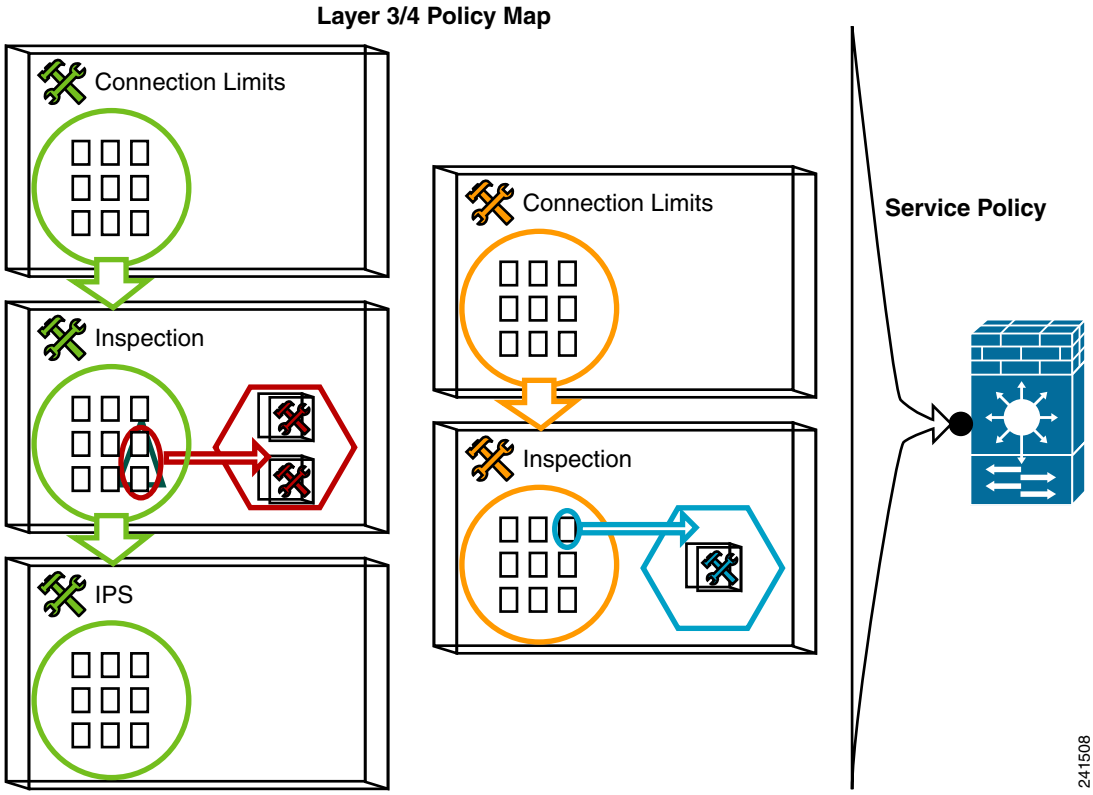
You can create a self-contained inspection policy map that identifies the traffic directly with **match** commands, or you can create an inspection class map for reuse or for more complicated matching. See Defining Actions in an Inspection Policy Map, page 2-4 and the Identifying Traffic in an Inspection Class Map, page 2-5.

Step 3 Create a regular expression—If you want to match text with a regular expression within inspected packets, you can create a regular expression or a group of regular expressions (a regular expression class map). Then, when you define the traffic to match for the inspection policy map, you can call on an existing regular expression.

For example, you might want to drop all HTTP requests with a URL including the text “example.com.”



Step 4 Define the actions you want to perform and determine on which interfaces you want to apply the policy map—Define the actions you want to perform on each Layer 3/4 class map by creating a Layer 3/4 policy map. Then, determine on which interfaces you want to apply the policy map using a service policy.



See Defining Actions (Layer 3/4 Policy Map), page 1-15 and the Applying Actions to an Interface (Service Policy), page 1-17.

Task Flow for Configuring Hierarchical Policy Maps for QoS Traffic Shaping

If you enable QoS traffic shaping for a class map, then you can optionally enable priority queuing for a subset of shaped traffic. To do so, you need to create a policy map for the priority queuing, and then within the traffic shaping policy map, you can call the priority class map. Only the traffic shaping class map is applied to an interface.

See Chapter 19, "Information About QoS," for more information about this feature.

Hierarchical policy maps are only supported for traffic shaping and priority queuing.

To implement a hierarchical policy map, perform the following steps:

Step 1 Identify the prioritized traffic according to the Identifying Traffic (Layer 3/4 Class Maps), page 1-12.

You can create multiple class maps to be used in the hierarchical policy map.

Step 2 Create a policy map according to the Defining Actions (Layer 3/4 Policy Map), page 1-15, and identify the sole action for each class map as **priority**.

Step 3 Create a separate policy map according to the Defining Actions (Layer 3/4 Policy Map), page 1-15, and identify the **shape** action for the **class-default** class map.

Traffic shaping can only be applied the to **class-default** class map.

Step 4 For the same class map, identify the priority policy map that you created in Step 2 using the **service-policy** *priority_policy_map* command.

Step 5 Apply the shaping policy map to the interface according to Applying Actions to an Interface (Service Policy), page 1-17.

Identifying Traffic (Layer 3/4 Class Maps)

A Layer 3/4 class map identifies Layer 3 and 4 traffic to which you want to apply actions. You can create multiple Layer 3/4 class maps for each Layer 3/4 policy map.

This section includes the following topics:

- Creating a Layer 3/4 Class Map for Through Traffic, page 1-12
- Creating a Layer 3/4 Class Map for Management Traffic, page 1-14

Creating a Layer 3/4 Class Map for Through Traffic

A Layer 3/4 class map matches traffic based on protocols, ports, IP addresses and other Layer 3 or 4 attributes.

Detailed Steps

	Command	Purpose
Step 1	**class-map** *class_map_name* **Example:** `hostname(config)# class-map all_udp`	Creates a Layer 3/4 class map, where *class_map_name* is a string up to 40 characters in length. The name "class-default" is reserved. All types of class maps use the same name space, so you cannot reuse a name already used by another type of class map. The CLI enters class-map configuration mode.
Step 2	(Optional) **description** *string* **Example:** `hostname(config-cmap)# description All UDP traffic`	Adds a description to the class map.
Step 3	Match traffic using one of the following:	Unless otherwise specified, you can include only one **match** command in the class map.
	match any **Example:** `hostname(config-cmap)# match any`	Matches all traffic.

Command	Purpose
match access-list *access_list_name* **Example:** `hostname(config-cmap)# match access-list udp`	Matches traffic specified by an extended ACL. If the ASA is operating in transparent firewall mode, you can use an EtherType ACL.
match port {**tcp** | **udp**} {**eq** *port_num* | range *port_num port_num*} **Example:** `hostname(config-cmap)# match tcp eq 80`	Matches TCP or UDP destination ports, either a single port or a contiguous range of ports. **Tip** For applications that use multiple, non-contiguous ports, use the **match access-list** command and define an ACE to match each port.
match default-inspection-traffic **Example:** `hostname(config-cmap)# match default-inspection-traffic`	Matches default traffic for inspection: the default TCP and UDP ports used by all applications that the ASA can inspect. This command, which is used in the default global policy, is a special CLI shortcut that when used in a policy map, ensures that the correct inspection is applied to each packet, based on the destination port of the traffic. For example, when UDP traffic for port 69 reaches the ASA, then the ASA applies the TFTP inspection; when TCP traffic for port 21 arrives, then the ASA applies the FTP inspection. So in this case only, you can configure multiple inspections for the same class map (with the exception of WAAS inspection, which can be configured with other inspections. See Incompatibility of Certain Feature Actions, page 1-5 for more information about combining actions). Normally, the ASA does not use the port number to determine the inspection applied, thus giving you the flexibility to apply inspections to non-standard ports, for example. See Default Settings and NAT Limitations, page 7-4 for a list of default ports. Not all applications whose ports are included in the **match default-inspection-traffic** command are enabled by default in the policy map. You can specify a **match access-list** command along with the **match default-inspection-traffic** command to narrow the matched traffic. Because the **match default-inspection-traffic** command specifies the ports and protocols to match, any ports and protocols in the ACL are ignored. **Tip** We suggest that you only inspect traffic on ports on which you expect application traffic; if you inspect all traffic, for example using **match any**, the ASA performance can be impacted.
match dscp *value1* [*value2*] [...] [*value8*] **Example:** `hostname(config-cmap)# match dscp af43 cs1 ef`	Matches DSCP value in an IP header, up to eight DSCP values.

Command	Purpose
match precedence *value1* [*value2*] [*value3*] [*value4*] **Example:** `hostname(config-cmap)# match precedence 1 4`	Matches up to four precedence values, represented by the TOS byte in the IP header, where *value1* through *value4* can be 0 to 7, corresponding to the possible precedences.
match rtp *starting_port range* **Example:** `hostname(config-cmap)# match rtp 4004 100`	Matches RTP traffic, where the *starting_port* specifies an even-numbered UDP destination port between 2000 and 65534. The *range* specifies the number of additional UDP ports to match above the *starting_port*, between 0 and 16383.

match tunnel-group *name*
 	
(Optional)
match flow ip destination-address
 	
Example:
`hostname(config-cmap)# match tunnel-group group1`
`hostname(config-cmap)# match flow ip destination-address`
	Matches VPN tunnel group traffic to which you want to apply QoS.
	
	You can also specify one other **match** command to refine the traffic match. You can specify any of the preceding commands, except for the **match any**, **match access-list**, or **match default-inspection-traffic** commands. Or you can also enter the **match flow ip destination-address** command to match flows in the tunnel group going to each IP address.

Examples

The following is an example for the **class-map** command:

```
hostname(config)# access-list udp permit udp any any
hostname(config)# access-list tcp permit tcp any any
hostname(config)# access-list host_foo permit ip any 10.1.1.1 255.255.255.255

hostname(config)# class-map all_udp
hostname(config-cmap)# description "This class-map matches all UDP traffic"
hostname(config-cmap)# match access-list udp

hostname(config-cmap)# class-map all_tcp
hostname(config-cmap)# description "This class-map matches all TCP traffic"
hostname(config-cmap)# match access-list tcp

hostname(config-cmap)# class-map all_http
hostname(config-cmap)# description "This class-map matches all HTTP traffic"
hostname(config-cmap)# match port tcp eq http

hostname(config-cmap)# class-map to_server
hostname(config-cmap)# description "This class-map matches all traffic to server 10.1.1.1"
hostname(config-cmap)# match access-list host_foo
```

Creating a Layer 3/4 Class Map for Management Traffic

For management traffic to the ASA, you might want to perform actions specific to this kind of traffic. You can specify a management class map that can match an ACL or TCP or UDP ports. The types of actions available for a management class map in the policy map are specialized for management traffic. See Supported Features, page 1-2.

Detailed Steps

	Command	Purpose
Step 1	**class-map type management** *class_map_name* **Example:** `hostname(config)# class-map type management all_mgmt`	Creates a management class map, where *class_map_name* is a string up to 40 characters in length. The name "class-default" is reserved. All types of class maps use the same name space, so you cannot reuse a name already used by another type of class map. The CLI enters class-map configuration mode.
Step 2	(Optional) **description** *string* **Example:** `hostname(config-cmap)# description All management traffic`	Adds a description to the class map.
Step 3	Match traffic using one of the following:	Unless otherwise specified, you can include only one **match** command in the class map.
	match access-list *access_list_name* **Example:** `hostname(config-cmap)# match access-list udp`	Matches traffic specified by an extended ACL. If the ASA is operating in transparent firewall mode, you can use an EtherType ACL.
	match port {**tcp** | **udp**} {**eq** *port_num* | range *port_num port_num*} **Example:** `hostname(config-cmap)# match tcp eq 80`	Matches TCP or UDP destination ports, either a single port or a contiguous range of ports. **Tip** For applications that use multiple, non-contiguous ports, use the **match access-list** command and define an ACE to match each port.

Defining Actions (Layer 3/4 Policy Map)

This section describes how to associate actions with Layer 3/4 class maps by creating a Layer 3/4 policy map.

Restrictions

The maximum number of policy maps is 64, but you can only apply one policy map per interface.

Detailed Steps

	Command	Purpose
Step 1	**policy-map** *policy_map_name* **Example:** `hostname(config)# policy-map global_policy`	Adds the policy map. The *policy_map_name* argument is the name of the policy map up to 40 characters in length. All types of policy maps use the same name space, so you cannot reuse a name already used by another type of policy map. The CLI enters policy-map configuration mode.
Step 2	(Optional) **class** *class_map_name* **Example:** `hostname(config-pmap)# description global policy map`	Specifies a previously configured Layer 3/4 class map, where the *class_map_name* is the name of the class map. See Identifying Traffic (Layer 3/4 Class Maps), page 1-12 to add a class map. **Note** If there is no **match default-inspection-traffic** command in a class map, then at most one **inspect** command is allowed to be configured under the class. For QoS, you can configure a hierarchical policy map for the traffic shaping and priority queue features. See Task Flow for Configuring Hierarchical Policy Maps for QoS Traffic Shaping, page 1-11 for more information.
Step 3	Specify one or more actions for this class map.	See Supported Features, page 1-2.
Step 4	Repeat Step 2 and Step 3 for each class map you want to include in this policy map.	

Examples

The following is an example of a **policy-map** command for connection policy. It limits the number of connections allowed to the web server 10.1.1.1:

```
hostname(config)# access-list http-server permit tcp any host 10.1.1.1
hostname(config)# class-map http-server
hostname(config-cmap)# match access-list http-server

hostname(config)# policy-map global-policy
hostname(config-pmap)# description This policy map defines a policy concerning connection
to http server.
hostname(config-pmap)# class http-server
hostname(config-pmap-c)# set connection conn-max 256
```

The following example shows how multi-match works in a policy map:

```
hostname(config)# class-map inspection_default
hostname(config-cmap)# match default-inspection-traffic
hostname(config)# class-map http_traffic
hostname(config-cmap)# match port tcp eq 80

hostname(config)# policy-map outside_policy
hostname(config-pmap)# class inspection_default
hostname(config-pmap-c)# inspect http http_map
hostname(config-pmap-c)# inspect sip
hostname(config-pmap)# class http_traffic
hostname(config-pmap-c)# set connection timeout idle 0:10:0
```

The following example shows how traffic matches the first available class map, and will not match any subsequent class maps that specify actions in the same feature domain:

```
hostname(config)# class-map telnet_traffic
hostname(config-cmap)# match port tcp eq 23
hostname(config)# class-map ftp_traffic
hostname(config-cmap)# match port tcp eq 21
hostname(config)# class-map tcp_traffic
hostname(config-cmap)# match port tcp range 1 65535
hostname(config)# class-map udp_traffic
hostname(config-cmap)# match port udp range 0 65535
hostname(config)# policy-map global_policy
hostname(config-pmap)# class telnet_traffic
hostname(config-pmap-c)# set connection timeout idle 0:0:0
hostname(config-pmap-c)# set connection conn-max 100
hostname(config-pmap)# class ftp_traffic
hostname(config-pmap-c)# set connection timeout idle 0:5:0
hostname(config-pmap-c)# set connection conn-max 50
hostname(config-pmap)# class tcp_traffic
hostname(config-pmap-c)# set connection timeout idle 2:0:0
hostname(config-pmap-c)# set connection conn-max 2000
```

When a Telnet connection is initiated, it matches **class telnet_traffic**. Similarly, if an FTP connection is initiated, it matches **class ftp_traffic**. For any TCP connection other than Telnet and FTP, it will match **class tcp_traffic**. Even though a Telnet or FTP connection can match **class tcp_traffic**, the ASA does not make this match because they previously matched other classes.

Applying Actions to an Interface (Service Policy)

To activate the Layer 3/4 policy map, create a service policy that applies it to one or more interfaces or that applies it globally to all interfaces.

Restrictions

You can only apply one global policy, so if you want to alter the global policy, you need to either edit the default policy or disable it and apply a new one. By default, the configuration includes a global policy that matches all default application inspection traffic and applies inspection to the traffic globally. The default service policy includes the following command:

```
service-policy global_policy global
```

Detailed Steps

Command	Purpose
service-policy *policy_map_name* **interface** *interface_name* [**fail-close**] **Example:** `hostname(config)# service-policy inbound_policy interface outside`	Creates a service policy by associating a policy map with an interface. Specify the **fail-close** option to generate a syslog (767001) for IPv6 traffic that is dropped by application inspections that do not support IPv6 traffic. By default, syslogs are not generated. For a list of inspections that support IPv6, see IPv6 Guidelines, page 1-6.
service-policy *policy_map_name* **global** [**fail-close**] **Example:** `hostname(config)# service-policy inbound_policy global`	Creates a service policy that applies to all interfaces that do not have a specific policy. Specify the **fail-close** option to generate a syslog (767001) for IPv6 traffic that is dropped by application inspections that do not support IPv6 traffic. By default, syslogs are not generated. For a list of inspections that support IPv6, see IPv6 Guidelines, page 1-6.

Examples

For example, the following command enables the inbound_policy policy map on the outside interface:

```
hostname(config)# service-policy inbound_policy interface outside
```

The following commands disable the default global policy, and enables a new one called new_global_policy on all other ASA interfaces:

```
hostname(config)# no service-policy global_policy global
hostname(config)# service-policy new_global_policy global
```

Monitoring Modular Policy Framework

To monitor Modular Policy Framework, enter the following command:

Command	Purpose
`show service-policy`	Displays the service policy statistics.

Configuration Examples for Modular Policy Framework

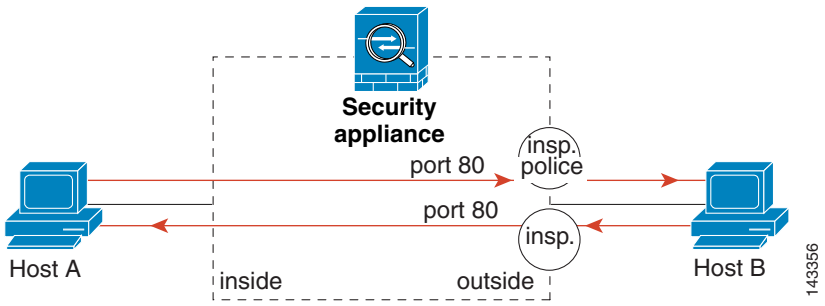
This section includes several Modular Policy Framework examples and includes the following topics:

- Applying Inspection and QoS Policing to HTTP Traffic, page 1-18
- Applying Inspection to HTTP Traffic Globally, page 1-19
- Applying Inspection and Connection Limits to HTTP Traffic to Specific Servers, page 1-20
- Applying Inspection to HTTP Traffic with NAT, page 1-21

Applying Inspection and QoS Policing to HTTP Traffic

In this example (see Figure 1-1), any HTTP connection (TCP traffic on port 80) that enters or exits the ASA through the outside interface is classified for HTTP inspection. Any HTTP traffic that exits the outside interface is classified for policing.

Figure 1-1 HTTP Inspection and QoS Policing



See the following commands for this example:

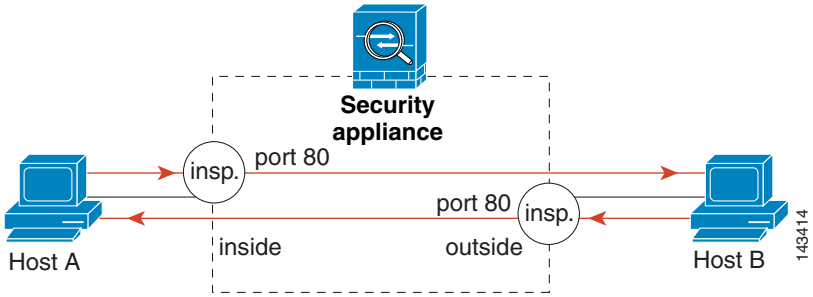
```
hostname(config)# class-map http_traffic
hostname(config-cmap)# match port tcp eq 80

hostname(config)# policy-map http_traffic_policy
hostname(config-pmap)# class http_traffic
hostname(config-pmap-c)# inspect http
hostname(config-pmap-c)# police output 250000
hostname(config)# service-policy http_traffic_policy interface outside
```

Applying Inspection to HTTP Traffic Globally

In this example (see Figure 1-2), any HTTP connection (TCP traffic on port 80) that enters the ASA through any interface is classified for HTTP inspection. Because the policy is a global policy, inspection occurs only as the traffic enters each interface.

Figure 1-2 Global HTTP Inspection



See the following commands for this example:

```
hostname(config)# class-map http_traffic
hostname(config-cmap)# match port tcp eq 80

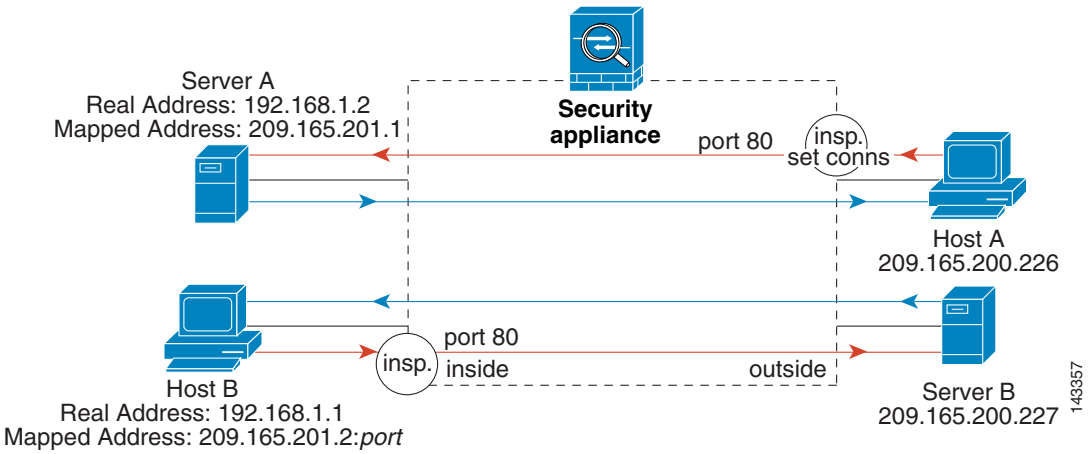
hostname(config)# policy-map http_traffic_policy
hostname(config-pmap)# class http_traffic
hostname(config-pmap-c)# inspect http
hostname(config)# service-policy http_traffic_policy global
```

Applying Inspection and Connection Limits to HTTP Traffic to Specific Servers

In this example (see Figure 1-3), any HTTP connection destined for Server A (TCP traffic on port 80) that enters the ASA through the outside interface is classified for HTTP inspection and maximum connection limits. Connections initiated from Server A to Host A does not match the ACL in the class map, so it is not affected.

Any HTTP connection destined for Server B that enters the ASA through the inside interface is classified for HTTP inspection. Connections initiated from Server B to Host B does not match the ACL in the class map, so it is not affected.

Figure 1-3 HTTP Inspection and Connection Limits to Specific Servers



See the following commands for this example:

```
hostname(config)# object network obj-192.168.1.2
hostname(config-network-object)# host 192.168.1.2
hostname(config-network-object)# nat (inside,outside) static 209.165.201.1
hostname(config)# object network obj-192.168.1.0
hostname(config-network-object)# subnet 192.168.1.0 255.255.255.0
hostname(config-network-object)# nat (inside,outside) dynamic 209.165.201.2
hostname(config)# access-list serverA extended permit tcp any host 209.165.201.1 eq 80
hostname(config)# access-list ServerB extended permit tcp any host 209.165.200.227 eq 80

hostname(config)# class-map http_serverA
hostname(config-cmap)# match access-list serverA
hostname(config)# class-map http_serverB
hostname(config-cmap)# match access-list serverB

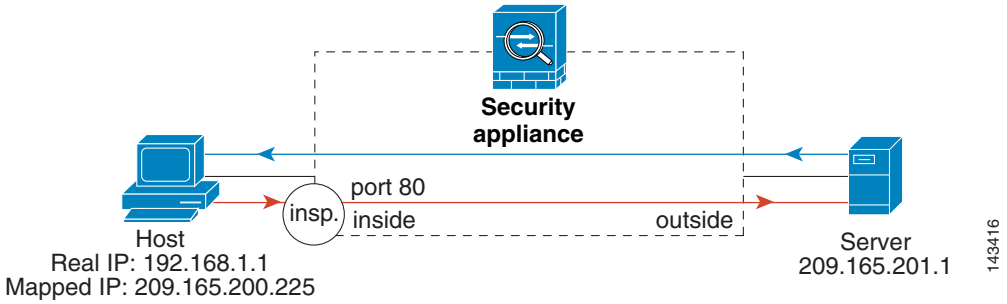
hostname(config)# policy-map policy_serverA
hostname(config-pmap)# class http_serverA
hostname(config-pmap-c)# inspect http
hostname(config-pmap-c)# set connection conn-max 100
hostname(config)# policy-map policy_serverB
hostname(config-pmap)# class http_serverB
hostname(config-pmap-c)# inspect http

hostname(config)# service-policy policy_serverB interface inside
hostname(config)# service-policy policy_serverA interface outside
```

Applying Inspection to HTTP Traffic with NAT

In this example, the Host on the inside network has two addresses: one is the real IP address 192.168.1.1, and the other is a mapped IP address used on the outside network, 209.165.200.225. You must use the real IP address in the ACL in the class map. If you applied it to the outside interface, you would also use the real address.

Figure 1-4 HTTP Inspection with NAT



See the following commands for this example:

```
hostname(config)# object network obj-192.168.1.1
hostname(config-network-object)# host 192.168.1.1
hostname(config-network-object)# nat (VM1,outside) static 209.165.200.225

hostname(config)# access-list http_client extended permit tcp host 192.168.1.1 any eq 80

hostname(config)# class-map http_client
hostname(config-cmap)# match access-list http_client

hostname(config)# policy-map http_client
hostname(config-pmap)# class http_client
hostname(config-pmap-c)# inspect http

hostname(config)# service-policy http_client interface inside
```

Feature History for Service Policies

Table 1-3 lists the release history for this feature.

Table 1-3 Feature History for Service Policies

Feature Name	Releases	Feature Information
Modular Policy Framework	7.0(1)	Modular Policy Framework was introduced.
Management class map for use with RADIUS accounting traffic	7.2(1)	The management class map was introduced for use with RADIUS accounting traffic. The following commands were introduced: **class-map type management**, and **inspect radius-accounting**.
Inspection policy maps	7.2(1)	The inspection policy map was introduced. The following command was introduced: **class-map type inspect**.

Table 1-3 Feature History for Service Policies (continued)

Feature Name	Releases	Feature Information
Regular expressions and policy maps	7.2(1)	Regular expressions and policy maps were introduced to be used under inspection policy maps. The following commands were introduced: **class-map type regex**, **regex**, **match regex**.
Match any for inspection policy maps	8.0(2)	The **match any** keyword was introduced for use with inspection policy maps: traffic can match one or more criteria to match the class map. Formerly, only **match all** was available.