CHAPTER 30

Configuring a Service Policy Using the Modular Policy Framework

Service policies using Modular Policy Framework provide a consistent and flexible way to configure adaptive security appliance 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 30-1
- Licensing Requirements for Service Policies, page 30-6
- Guidelines and Limitations, page 30-6
- Default Settings, page 30-8
- Task Flows for Configuring Service Policies, page 30-9
- Identifying Traffic (Layer 3/4 Class Maps), page 30-12
- Defining Actions (Layer 3/4 Policy Map), page 30-15
- Applying Actions to an Interface (Service Policy), page 30-17
- Monitoring Modular Policy Framework, page 30-18
- Configuration Examples for Modular Policy Framework, page 30-18
- Feature History for Service Policies, page 30-21

Information About Service Policies

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

- Supported Features for Through Traffic, page 30-2
- Supported Features for Management Traffic, page 30-2
- Feature Directionality, page 30-2
- Feature Matching Within a Service Policy, page 30-3
- Order in Which Multiple Feature Actions are Applied, page 30-4
- Incompatibility of Certain Feature Actions, page 30-5
- Feature Matching for Multiple Service Policies, page 30-6

Supported Features for Through Traffic

Table 30-1 lists the features supported by Modular Policy Framework.

Table 30-1 Modular Policy Framework

Feature	See:
Application inspection (multiple types)	• Chapter 38, "Getting Started With Application Layer Protocol Inspection." • Chapter 39, "Configuring Inspection of Basic Internet Protocols." • Chapter 41, "Configuring Inspection of Database and Directory Protocols." • Chapter 42, "Configuring Inspection for Management Application Protocols." • Chapter 40, "Configuring Inspection for Voice and Video Protocols."
CSC	Chapter 56, "Configuring the Content Security and Control Application on the CSC SSM."
IPS	Chapter 55, "Configuring the IPS Module."
NetFlow Secure Event Logging filtering	Chapter 73, "Configuring NetFlow Secure Event Logging (NSEL)."
QoS input and output policing	Chapter 50, "Configuring QoS."
QoS standard priority queue	Chapter 50, "Configuring QoS."
QoS traffic shaping, hierarchical priority queue	Chapter 50, "Configuring QoS."
TCP and UDP connection limits and timeouts, and TCP sequence number randomization	Chapter 49, "Configuring Connection Settings."
TCP normalization	Chapter 49, "Configuring Connection Settings."
TCP state bypass	Chapter 49, "Configuring Connection Settings."

Supported Features for Management Traffic

Modular Policy Framework supports the following features for management traffic:

- Application inspection for RADIUS accounting traffic—See Chapter 42, "Configuring Inspection for Management Application Protocols."
- Connection limits—See Chapter 49, "Configuring Connection Settings."

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 30-2 for the directionality of each feature.

Table 30-2 Feature Directionality

Feature	Single Interface Direction	Global Direction
Application inspection (multiple types)	Bidirectional	Ingress
CSC	Bidirectional	Ingress
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

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 adaptive security appliance 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 adaptive security appliance also applies the actions for the subsequent class map, if supported. See the "Incompatibility of Certain Feature Actions" section on page 30-5 for more information about unsupported combinations.

For example, if a packet matches a class map for connection limits, and also matches a class map for 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.

Note Application inspection includes multiple inspection types, and each inspection type is a separate feature when you consider the matching guidelines above.

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 is 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 adaptive security appliance 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. CSC
4. Application inspection (multiple types)

   The order of application inspections applied when a class of traffic is classified for multiple inspections is as follows. Only one inspection type can be applied to the same traffic. WAAS inspection is an exception, because it can be applied along with other inspections for the same traffic. See the "Incompatibility of Certain Feature Actions" section on page 30-5 for more information.

   a. CTIQBE
   b. DNS
   c. FTP
   d. GTP
   e. H323
   f. HTTP
   g. ICMP
   h. ICMP error
   i. ILS
   j. MGCP
   k. NetBIOS
   l. PPTP
   m. Sun RPC
   n. RSH
   o. RTSP
   p. SIP
   q. Skinny
   r. SMTP

s. SNMP
t. SQL*Net
u. TFTP
v. XDMCP
w. DCERPC
x. Instant Messaging

Note RADIUS accounting is not listed because it is the only inspection allowed on management traffic. WAAS is not listed because it can be configured along with other inspections for the same traffic.

5. IPS
6. QoS output policing
7. QoS standard priority queue
8. QoS traffic shaping, hierarchical priority queue

Incompatibility of Certain Feature Actions

Some features are not compatible with each other for the same traffic. For example, you cannot configure QoS priority queueing and QoS policing for the same set of traffic. Also, most inspections should not be combined with another inspection, so the adaptive security appliance only applies one inspection if you configure multiple inspections for the same traffic. In this case, the feature that is applied is the higher priority feature in the list in the "Order in Which Multiple Feature Actions are Applied" section on page 30-4.

For information about compatibility of each feature, see the chapter or section for your feature.

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 adaptive security appliance, then the adaptive security appliance applies the TFTP inspection; when TCP traffic for port 21 arrives, then the adaptive security appliance applies the FTP inspection. So in this case only, you can configure multiple inspections for the same class map. Normally, the adaptive security appliance 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.

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 30-1, traffic destined to port 21 is mistakenly configured for both FTP and HTTP inspection. In Example 30-2, 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 30-1 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 30-2 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
All 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 FTP, HTTP, ICMP, SIP, SMTP and IPSec-pass-thru
- IPS
- NetFlow Secure Event Logging filtering
- TCP and UDP connection limits and timeouts, TCP sequence number randomization
- TCP normalization
- TCP state bypass

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 maps to approximately 235. See the "Default Class Maps" section on page 30-9.

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 the "Incompatibility of Certain Feature Actions" section on page 30-5.

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.

Default Settings

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

- Default Configuration, page 30-8
- Default Class Maps, page 30-9

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 inspection for the maximum message length of 512 bytes
- FTP
- H323 (H225)
- H323 (RAS)
- RSH
- RTSP
- ESMTP
- SQLnet
- Skinny (SCCP)
- SunRPC
- XDMCP
- SIP
- NetBios
- TFTP

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 512
policy-map global_policy
 class inspection_default
  inspect dns preset_dns_map
  inspect ftp
  inspect h323 h225
  inspect h323 ras
  inspect rsh
  inspect rtsp
  inspect esmtp
  inspect sqlnet
  inspect skinny
  inspect sunrpc
  inspect xdmcp
  inspect sip
  inspect netbios
  inspect tftp
service-policy global_policy global
```

Note See the "Incompatibility of Certain Feature Actions" section on page 30-5 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 adaptive security appliance 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 adaptive security appliance, then the adaptive security appliance applies the TFTP inspection; when TCP traffic for port 21 arrives, then the adaptive security appliance applies the FTP inspection. So in this case only, you can configure multiple inspections for the same class map. Normally, the adaptive security appliance 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 adaptive security appliance 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:

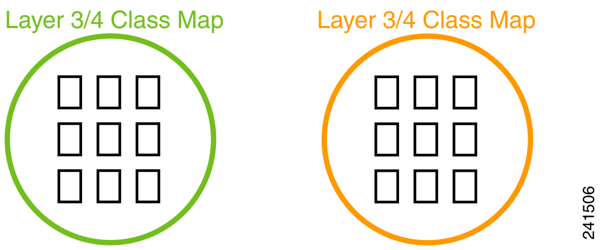
- Task Flow for Using the Modular Policy Framework, page 30-9
- Task Flow for Configuring Hierarchical Policy Maps for QoS Traffic Shaping, page 30-11

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.

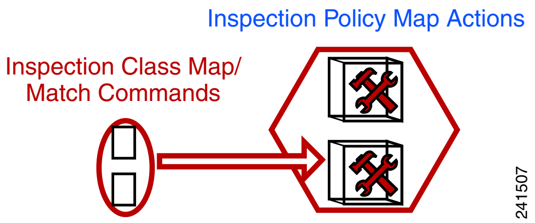
For example, you might want to perform actions on all traffic that passes through the adaptive security appliance; or you might only want to perform certain actions on traffic from 10.1.1.0/24 to any destination address.



See the "Identifying Traffic (Layer 3/4 Class Maps)" section on page 30-12.

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.

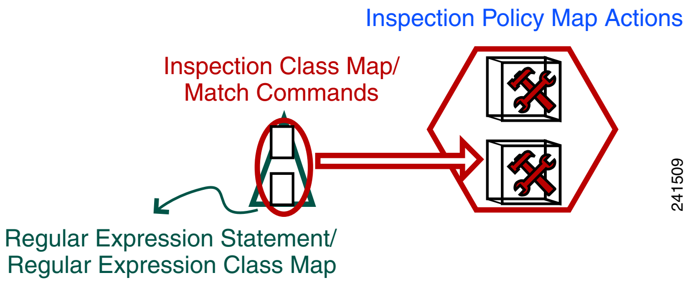
For example, you might want to drop all HTTP requests with a body length greater than 1000 bytes.



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 the "Defining Actions in an Inspection Policy Map" section on page 31-2 and the "Identifying Traffic in an Inspection Class Map" section on page 31-5.

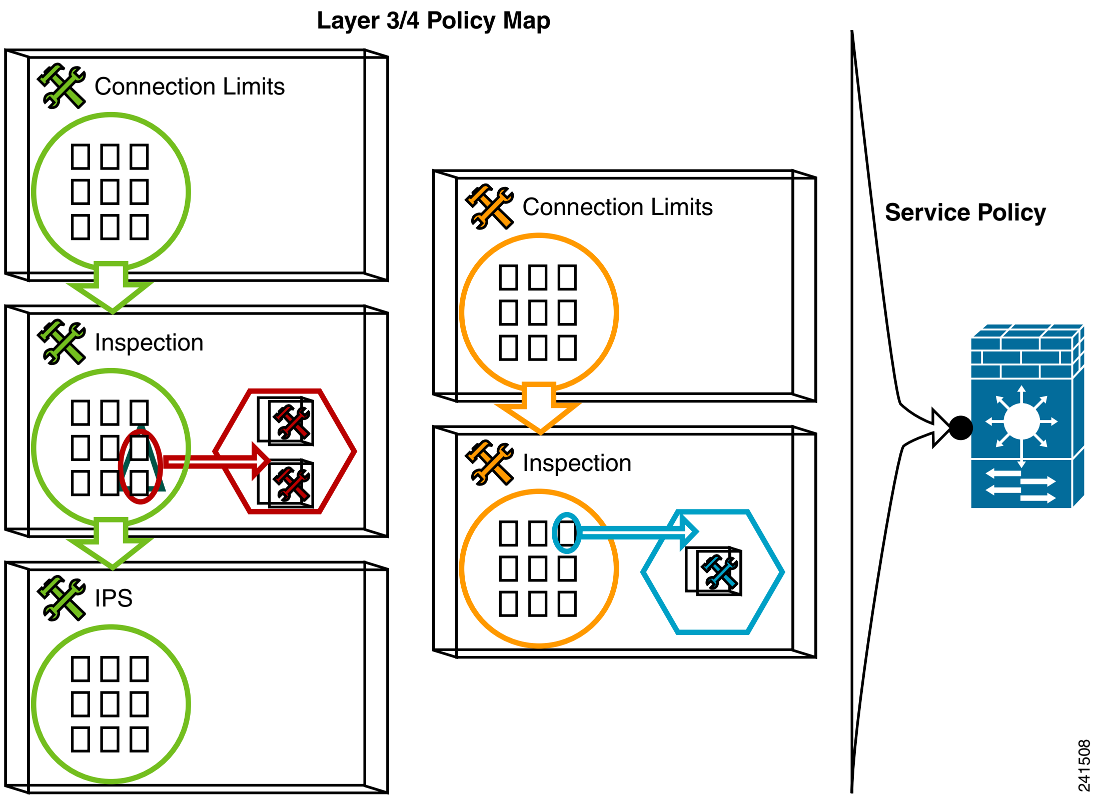
Step 3 Create a regular expression—If you want to match text with a regular expression within inspected packets, you can create a regular expression or a group of regular expressions (a regular expression class map). Then, when you define the traffic to match for the inspection policy map, you can call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

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


Inspection Policy Map Actions
Inspection Class Map/
Match Commands
Regular Expression Statement/
Regular Expression Class Map
241509

See the "Creating a Regular Expression" section on page 11-12 and the "Creating a Regular Expression Class Map" section on page 11-15.

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 the "Defining Actions (Layer 3/4 Policy Map)" section on page 30-15 and the "Applying Actions to an Interface (Service Policy)" section on page 30-17.

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 queueing for a subset of shaped traffic. To do so, you need to create a policy map for the priority queueing, 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 50, "Information About QoS," for more information about this feature.

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

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)" section on page 30-12.

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)" section on page 30-15, 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)" section on page 30-15, 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 accrding to "Applying Actions to an Interface (Service Policy)" section on page 30-17.

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 30-12
- Creating a Layer 3/4 Class Map for Management Traffic, page 30-15

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

Step 1 Create a Layer 3/4 class map by entering the following command:

```
hostname(config)# class-map class_map_name
hostname(config-cmap)#
```

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) Add a description to the class map by entering the following command:

```
hostname(config-cmap)# description string
```

Step 3 Define the traffic to include in the class by matching one of the following characteristics. Unless otherwise specified, you can include only one **match** command in the class map.

- Any traffic—The class map matches all traffic.

```
hostname(config-cmap)# match any
```

Note For features that support IPv6 (see the "Guidelines and Limitations" section on page 30-6), then the **match any** and **match default-inspection-traffic** commands are the only commands that match IPv6 traffic. For example, you cannot match an IPv6 access list.

- Access list—The class map matches traffic specified by an extended access list. If the adaptive security appliance is operating in transparent firewall mode, you can use an EtherType access list.

```
hostname(config-cmap)# match access-list access_list_name
```

For more information about creating access lists, see Chapter 13, "Adding an Extended Access List," or Chapter 14, "Adding an EtherType Access List.".

For information about creating access lists with NAT, see the "IP Addresses Used for Access Lists When You Use NAT" section on page 12-3.

- TCP or UDP destination ports—The class map matches a single port or a contiguous range of ports.

```
hostname(config-cmap)# match port {tcp | udp} {eq port_num | range port_num port_num}
```

Tip For applications that use multiple, non-contiguous ports, use the **match access-list** command and define an ACE to match each port.

For a list of ports you can specify, see the "TCP and UDP Ports" section on page B-11.

For example, enter the following command to match TCP packets on port 80 (HTTP):

```
hostname(config-cmap)# match tcp eq 80
```

- Default traffic for inspection—The class map matches the default TCP and UDP ports used by all applications that the adaptive security appliance can inspect.

```
hostname(config-cmap)# match default-inspection-traffic
```

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 adaptive security appliance, then the adaptive security appliance applies the TFTP inspection; when TCP traffic for port 21 arrives, then the adaptive security appliance 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 the "Incompatibility of Certain Feature Actions" section on page 30-5 for more information about combining actions). Normally, the adaptive security appliance 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 the "Default Settings" section on page 38-4 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 access list 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 adaptive security appliance performance can be impacted.

Note For features that support IPv6 (see the "Guidelines and Limitations" section on page 30-6), then the **match any** and **match default-inspection-traffic** commands are the only commands that match IPv6 traffic. For example, you cannot match an IPv6 access list.

- DSCP value in an IP header—The class map matches up to eight DSCP values.

```
hostname(config-cmap)# match dscp value1 [value2] [...] [value8]
```

For example, enter the following:

```
hostname(config-cmap)# match dscp af43 cs1 ef
```

- Precedence—The class map matches up to four precedence values, represented by the TOS byte in the IP header.

  ```
  hostname(config-cmap)# match precedence value1 [value2] [value3] [value4]
  ```

  where *value1* through *value4* can be 0 to 7, corresponding to the possible precedences.
- RTP traffic—The class map matches RTP traffic.

  ```
  hostname(config-cmap)# match rtp starting_port range
  ```

  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.
- Tunnel group traffic—The class map matches traffic for a tunnel group to which you want to apply QoS.

  ```
  hostname(config-cmap)# match tunnel-group name
  ```

  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 enter the following command to police each flow:

  ```
  hostname(config-cmap)# match flow ip destination address
  ```

  All traffic going to a unique IP destination address is considered a flow.

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 adaptive security appliance, you might want to perform actions specific to this kind of traffic. You can specify a management class map that can match an access list 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 the "Supported Features for Management Traffic" section on page 30-2.

Detailed Steps

Step 1 Create a class map by entering the following command:

```
hostname(config)# class-map type management class_map_name
hostname(config-cmap)#
```

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) Add a description to the class map by entering the following command:

```
hostname(config-cmap)# description string
```

Step 3 Define the traffic to include in the class by matching one of the following characteristics. You can include only one **match** command in the class map.

- Access list—The class map matches traffic specified by an extended access list. If the adaptive security appliance is operating in transparent firewall mode, you can use an EtherType access list.

  ```
  hostname(config-cmap)# match access-list access_list_name
  ```

  For more information about creating access lists, see Chapter 13, "Adding an Extended Access List," or Chapter 14, "Adding an EtherType Access List."

  For information about creating access lists with NAT, see the "IP Addresses Used for Access Lists When You Use NAT" section on page 12-3.

- TCP or UDP destination ports—The class map matches a single port or a contiguous range of ports.

  ```
  hostname(config-cmap)# match port {tcp | udp} {eq port_num | range port_num port_num}
  ```

Tip For applications that use multiple, non-contiguous ports, use the **match access-list** command and define an ACE to match each port.

For a list of ports you can specify, see the "TCP and UDP Ports" section on page B-11.

For example, enter the following command to match TCP packets on port 80 (HTTP):

```
hostname(config-cmap)# match tcp eq 80
```

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

Step 1 Add the policy map by entering the following command:

```
hostname(config)# policy-map policy_map_name
```

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) Specify a description for the policy map:

```
hostname(config-pmap)# description text
```

Step 3 Specify a previously configured Layer 3/4 class map using the following command:

```
hostname(config-pmap)# class class_map_name
```

where the *class_map_name* is the name of the class map you created earlier. See the "Identifying Traffic (Layer 3/4 Class Maps)" section on page 30-12 to add a class map.

Step 4 Specify one or more actions for this class map. See the "Supported Features for Through Traffic" section on page 30-2.

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 the "Task Flow for Configuring Hierarchical Policy Maps for QoS Traffic Shaping" section on page 30-11 for more information.

Step 5 Repeat Step 3 and Step 4 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 tcp 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 tcp 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 tcp 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 tcp 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 adaptive security appliance 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.

Detailed Steps

- To create a service policy by associating a policy map with an interface, enter the following command:

  ```
  hostname(config)# service-policy policy_map_name interface interface_name
  ```

- To create a service policy that applies to all interfaces that do not have a specific policy, enter the following command:

  ```
  hostname(config)# service-policy policy_map_name global
  ```

By default, the configuration includes a global policy that matches all default application inspection traffic and applies inspection to the traffic globally. 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.

The default service policy includes the following command:

```
service-policy global_policy global
```

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 adaptive security appliance 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

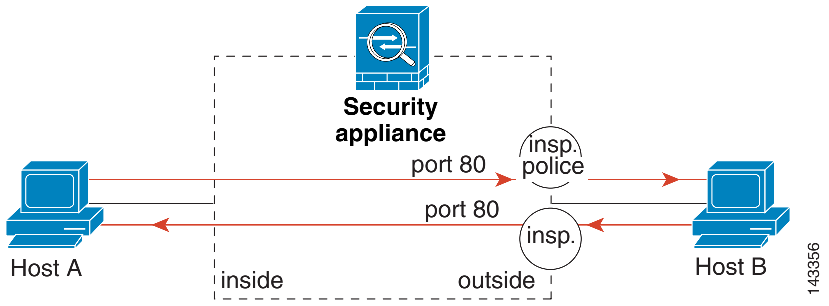
This section includes several Modular Policy Framework examples and includes the following topics:

- Applying Inspection and QoS Policing to HTTP Traffic, page 30-19
- Applying Inspection to HTTP Traffic Globally, page 30-19
- Applying Inspection and Connection Limits to HTTP Traffic to Specific Servers, page 30-20
- Applying Inspection to HTTP Traffic with NAT, page 30-21

Applying Inspection and QoS Policing to HTTP Traffic

In this example (see Figure 30-1), any HTTP connection (TCP traffic on port 80) that enters or exits the adaptive security appliance through the outside interface is classified for HTTP inspection. Any HTTP traffic that exits the outside interface is classified for policing.

Figure 30-1 HTTP Inspection and QoS Policing



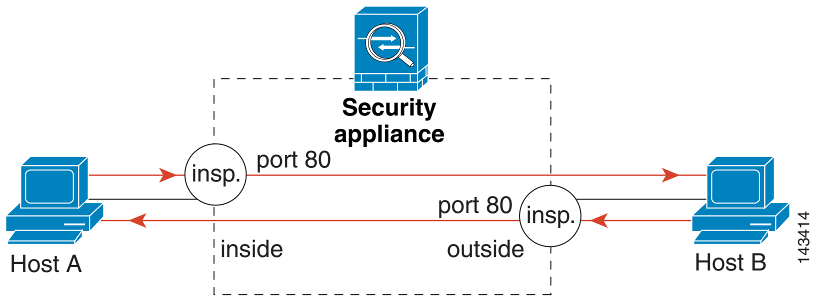
See the following commands for this example:

```
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 30-2), any HTTP connection (TCP traffic on port 80) that enters the adaptive security appliance 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 30-2 Global HTTP Inspection



See the following commands for this example:

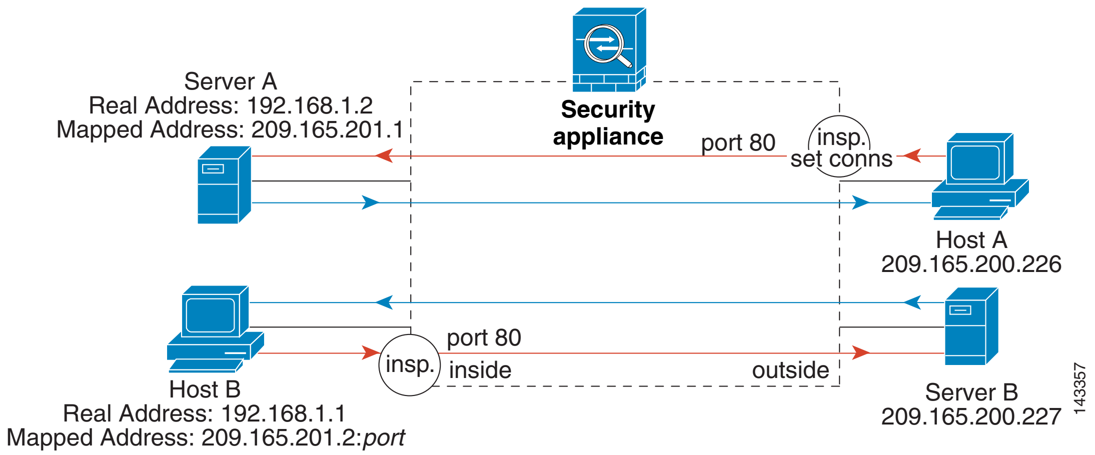
```
hostname(config)# class-map http_traffic
hostname(config-cmap)# match port tcp eq 80
hostname(config)# policy-map http_traffic_policy
hostname(config-pmap)# class http_traffic
hostname(config-pmap-c)# inspect http
hostname(config)# service-policy http_traffic_policy global
```

Applying Inspection and Connection Limits to HTTP Traffic to Specific Servers

In this example (see Figure 30-3), any HTTP connection destined for Server A (TCP traffic on port 80) that enters the adaptive security appliance 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 access list in the class map, so it is not affected.

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

Figure 30-3 HTTP Inspection and Connection Limits to Specific Servers



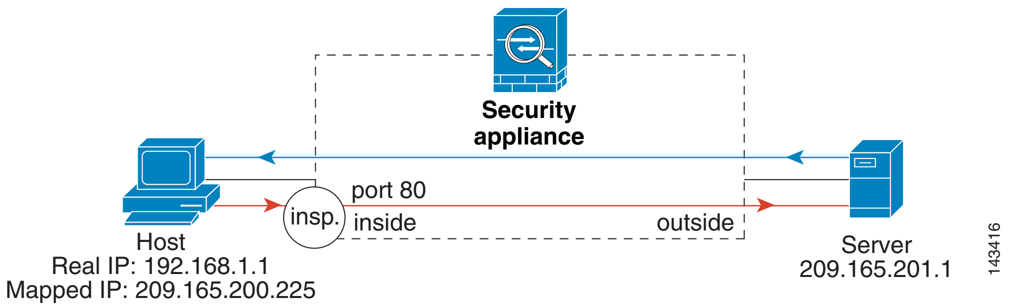
See the following commands for this example:

```
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. Because the policy is applied to the inside interface, where the real address is used, then you must use the real IP address in the access list in the class map. If you applied it to the outside interface, you would use the mapped address.

Figure 30-4 HTTP Inspection with NAT



See the following commands for this example:

```
hostname(config)# static (inside,outside) 209.165.200.225 192.168.1.1
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 30-3 lists the release history for this feature.

Table 30-3 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**.

Table 30-3 Feature History for Service Policies (continued)

Feature Name	Releases	Feature Information
Inspection policy maps	7.2(1)	The inspection policy map was introduced. The following commands were introduced: **policy-map type inspect**, **class-map type inspect**.
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.
Maximum connections and embryonic connections for management traffic	8.0(2)	The **set connection** command is now available for a Layer 3/4 management class map, for to-the-security appliance management traffic. Only the **conn-max** and **embryonic-conn-max** keywords are available.