# Service Policy

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 actions or rules applied to an interface or applied globally.



# About Service Policies

The following topics describe how service policies work.

## The Components of a Service Policy

The point of service policies is to apply advanced services to the traffic you are allowing. Any traffic permitted by access rules can have service policies applied, and thus receive special processing, such as being redirected to a service module or having application inspection applied.

You can have these types of service policy:

- One global policy that gets applied to all interfaces.
- One service policy applied per interface. The policy can be a mix of classes for traffic going through the device and management traffic directed at the ASA interface rather than going through it,

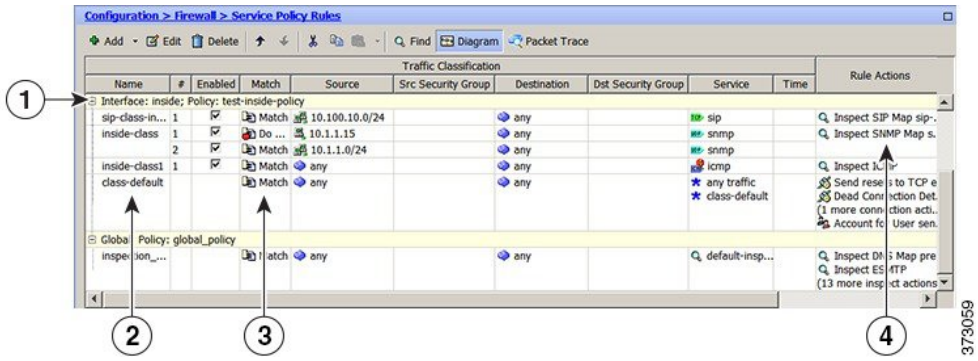
Each service policy is composed of the following elements:

1. Service policy map, which is the ordered set of rules, and is named on the **service-policy** command. In ASDM, the policy map is represented as a folder on the Service Policy Rules page.
2. Rules, each rule being a **class** command within the service policy map and the commands associated with the **class** command. In ASDM, each rule is shown on a separate row, and the name of the rule is the class name.

The **class** command defines the traffic matching criteria for the rule.

The commands associated with class, such as **inspect**, **set connection timeout**, and so forth, define the services and constraints to apply to matching traffic. Note that inspect commands can point to inspection policy maps, which define actions to apply to inspected traffic. Keep in mind that inspection policy maps are not the same as service policy maps.

The following example compares how service policies appear in the CLI with how they appear in ASDM. Note that there is not a one-to-one mapping between the figure call-outs and lines in the CLI.




The following CLI is generated by the rules shown in the figure above.

```
: Access lists used in class maps.
: In ASDM, these map to call-out 3, from the Match to the Time fields.
access-list inside_mpc line 1 extended permit tcp 10.100.10.0 255.255.255.0 any eq sip
access-list inside_mpc_1 line 1 extended deny udp host 10.1.1.15 any eq snmp
access-list inside_mpc_1 line 2 extended permit udp 10.1.1.0 255.255.255.0 any eq snmp
access-list inside_mpc_2 line 1 extended permit icmp any any
: SNMP map for SNMP inspection. Denies all but v3.
: In ASDM, this maps to call-out 4, rule actions, for the class-inside policy.
snmp-map snmp-v3only
  deny version 1
  deny version 2
  deny version 2c
: Inspection policy map to define SIP behavior.
: The sip-high inspection policy map must be referred to by an inspect sip command
: in the service policy map.
: In ASDM, this maps to call-out 4, rule actions, for the sip-class-inside policy.
policy-map type inspect sip sip-high
  parameters
    rtp-conformance enforce-payloadtype
    no traffic-non-sip
    software-version action mask log
    uri-non-sip action mask log
    state-checking action drop-connection log
    max-forwards-validation action drop log
    strict-header-validation action drop log
: Class map to define traffic matching for the inside-class rule.
: In ASDM, this maps to call-out 3, from the Match to the Time fields.
class-map inside-class
  match access-list inside_mpc_1
: Class map to define traffic matching for the sip-class-inside rule.
: In ASDM, this maps to call-out 3, from the Match to the Time fields.
class-map sip-class-inside
  match access-list inside_mpc
: Class map to define traffic matching for the inside-class1 rule.
: In ASDM, this maps to call-out 3, from the Match to the Time fields.
```

```
class-map inside-class1
  match access-list inside_mpc_2
: Policy map that actually defines the service policy rule set named test-inside-policy.
: In ASDM, this corresponds to the folder at call-out 1.
policy-map test-inside-policy
: First rule in test-inside-policy, named sip-class-inside. Inspects SIP traffic.
: The sip-class-inside rule applies the sip-high inspection policy map to SIP inspection.
: In ASDM, each rule corresponds to call-out 2.
  class sip-class-inside
    inspect sip sip-high
: Second rule, inside-class. Applies SNMP inspection using an SNMP map.
  class inside-class
    inspect snmp snmp-v3only
: Third rule, inside-class1. Applies ICMP inspection.
  class inside-class1
    inspect icmp
: Fourth rule, class-default. Applies connection settings and enables user statistics.
  class class-default
    set connection timeout embryonic 0:00:30 half-closed 0:10:00 idle 1:00:00
reset dcd 0:15:00 5
    user-statistics accounting
: The service-policy command applies the policy map rule set to the inside interface.
: This command activates the policies.
service-policy test-inside-policy interface inside
```

# Features Configured with Service Policies

The following table lists the features you configure using service policies.

***Table 1: Features Configured with Service Policies***

| Feature | For Through Traffic? | For Management Traffic? | See: |
|---|---|---|---|
| Application inspection (multiple types) | **All** except RADIUS accounting | **RADIUS accounting** only | • Getting Started with Application Layer Protocol Inspection.<br>• Inspection of Basic Internet Protocols.<br>• Inspection for Voice and Video Protocols.<br>• Inspection for Mobile Networks. |
| ASA FirePOWER (ASA SFR) | **Yes** | No | ASA FirePOWER Module. |
| NetFlow Secure Event Logging filtering | **Yes** | **Yes** | See the NetFlow implementation guide. |
| QoS input and output policing | **Yes** | No | Quality of Service. |
| QoS standard priority queue | **Yes** | No | Quality of Service. |
| TCP and UDP connection limits and timeouts, and TCP sequence number randomization | **Yes** | **Yes** | Connection Settings. |
| TCP normalization | **Yes** | No | Connection Settings. |

| Feature | For Through Traffic? | For Management Traffic? | See: |
|---|---|---|---|
| TCP state bypass | **Yes** | No | 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 the following table for the directionality of each feature.

***Table 2: Feature Directionality***

| Feature | Single Interface Direction | Global Direction |
|---|---|---|
| Application inspection (multiple types) | Bidirectional | Ingress |
| ASA FirePOWER (ASA SFR) | 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 |
| 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

A packet matches class maps in a policy map for a given interface according to the following rules:

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, on page 6 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.

### Examples of Packet Matching

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 cannot 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.

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 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. Application inspections that can be combined with other inspections:
   a. IPv6
   b. IP options
   c. WAAS
4. Application inspections that cannot be combined with other inspections. See Incompatibility of Certain Feature Actions, on page 6 for more information.
5. ASA FirePOWER (ASA SFR)
6. QoS output policing
7. QoS standard priority queue

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

# Incompatibility of Certain Feature Actions

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

- You cannot configure QoS priority queuing 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. Exceptions are listed in Order in Which Multiple Feature Actions are Applied, on page 5.
- You cannot configure traffic to be sent to multiple modules.
- HTTP inspection is not compatible with ASA FirePOWER.

**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.

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 the first example, traffic destined to port 21 is mistakenly configured for both FTP and HTTP inspection. In the second example, 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: 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 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 ftp
     inspect ftp
   class http
     inspect http
```

## 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.

# Guidelines for Service Policies

### Inspection Guidelines

There is a separate topic that provides detailed guidelines for application inspection service policies. See Guidelines for Application Inspection.

### IPv6 Guidelines

Supports IPv6 for the following features:

- Application inspection for several, but not all, protocols. For details, see Guidelines for Application Inspection.

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

### Class Map (Traffic Class) Guidelines

The maximum number of class maps (traffic classes) of 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.

### 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, on page 6.

### Service Policy Guidelines

- Interface service policies on ingress interfaces 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 ingress interface policy with FTP inspection, then only the ingress interface policy FTP inspection is applied to that interface. If no ingress or global policy implements a feature, then an interface service policy on the egress interface that specifies the feature is applied.
- 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. Output for the **show** command will not include data about the old connections.

  For example, if you remove a QoS service policy from an interface, then 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. Use the **clear conn** or **clear local-host** commands.

# Defaults for Service Policies

The following topics describe the default settings for service policies and the Modular Policy Framework.

## Default Service Policy 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
- 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
  service-policy global_policy global
```

# Default Class Maps (Traffic Classes)

The configuration includes a default Layer 3/4 class map (traffic class) 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.

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

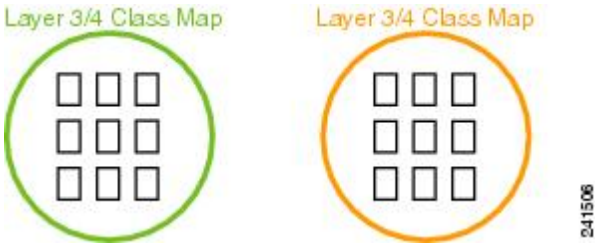
# Configure Service Policies

To configure service policies using the Modular Policy Framework, perform the following steps:

**Procedure**

**Step 1** Identify the traffic on which you want to act by creating Layer 3/4 class maps, as described in Identify Traffic (Layer 3/4 Class Maps), on page 12.

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.




**Step 2** Optionally, perform additional actions on some inspection traffic.

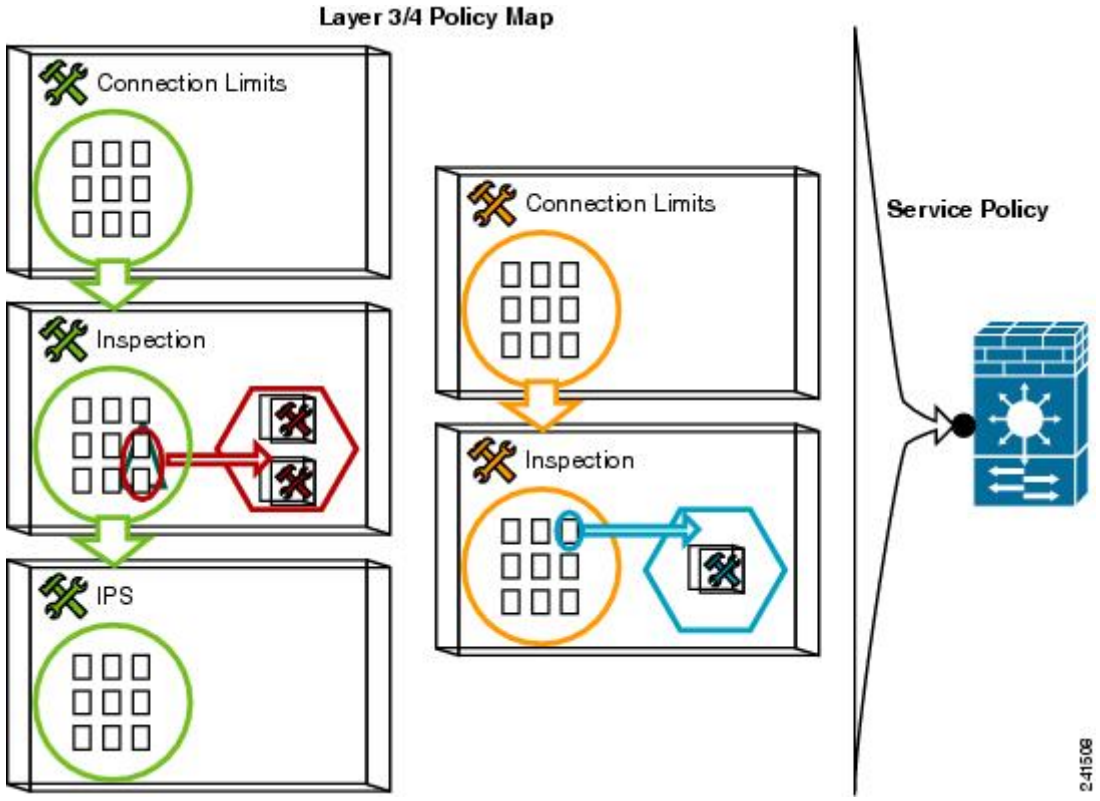
If one of the actions you want to perform is application inspection, and you want to perform additional actions on some inspection traffic, then create an inspection policy map. The inspection policy map identifies the traffic and specifies what to do with it.

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. For example, you could match text within a inspected packets using a regular expression or a group of regular expressions (a regular expression class map), and target actions based on narrower criteria. For example, you might want to drop all HTTP requests with a URL including the text “example.com.”

See Configure Application Layer Protocol Inspection.

**Step 3** Define the actions you want to perform on each Layer 3/4 class map by creating a Layer 3/4 policy map, as described in Define Actions (Layer 3/4 Policy Map), on page 16.




**Step 4** Determine on which interfaces you want to apply the policy map, or apply it globally, as described in Apply Actions to an Interface (Service Policy), on page 17.

# Identify 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.

## Create 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.

**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.

### Procedure

**Step 1** Create a Layer 3/4 class map: **class-map** *class_map_name*

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.

**Example:**

```
hostname(config)# class-map all_udp
```

**Step 2** (Optional) Add a description to the class map.

**description** *string*

**Example:**

```
hostname(config-cmap)# description All UDP traffic
```

**Step 3** Match traffic using one of the following commands. Unless otherwise specified, you can include only one **match** command in the class map.

- **match any**—Matches all traffic.

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

- **match access-list** *access_list_name*—Matches traffic specified by an extended ACL.

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

- **match port** {**tcp** | **udp** | **sctp**} {**eq** *port_num* | **range** *port_num port_num*}—Matches destination ports, either a single port or a contiguous range of ports, for the indicated protocol. For applications that use multiple, non-contiguous ports, use the **match access-list** command and define an ACE to match each port.

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

- **match default-inspection-traffic**—Matches default traffic for inspection: the default TCP and UDP ports used by all applications that the ASA 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 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, on page 6 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 Inspections and NAT Limitations 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.

- **match dscp** *value1* [*value2*] [...] [*value8*]—Matches the DSCP value in an IP header, up to eight DSCP values.

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

- **match precedence** *value1* [*value2*] [*value3*] [*value4*]—Matches up to four precedence values, represented by the TOS byte in the IP header, where the precedence values can be 0 to 7.

```
hostname(config-cmap)# match precedence 1 4
```

- **match rtp** *starting_port range*—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.

```
hostname(config-cmap)# match rtp 4004 100
```

- **match tunnel-group** *name*—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.

```
hostname(config-cmap)# match tunnel-group group1
hostname(config-cmap)# match flow ip destination-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
```

## Create 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.

### Procedure

**Step 1** Create a management class map: **class-map type management** *class_map_name*

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.

**Example:**

```
hostname(config)# class-map management all_udp
```

**Step 2** (Optional) Add a description to the class map.

**description** *string*

**Example:**

```
hostname(config-cmap)# description All UDP traffic
```

**Step 3** Match traffic using one of the following commands.

- **match access-list** *access_list_name*—Matches traffic specified by an extended ACL.

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

- **match port** {**tcp** | **udp** | **sctp**} {**eq** *port_num* | **range** *port_num port_num*}—Matches destination ports, either a single port or a contiguous range of ports, for the indicated protocol. For applications that use multiple, non-contiguous ports, use the **match access-list** command and define an ACE to match each port.

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

# Define Actions (Layer 3/4 Policy Map)

After you configure Layer 3/4 class maps to identify traffic, use a Layer 3/4 policy map to associate actions to those classes.

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

### Procedure

**Step 1** Add the policy map: **policy-map** *policy_map_name*

Where *policy_map_name* 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.

**Example:**

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

**Step 2** Specify a previously configured Layer 3/4 class map: **class** *class_map_name*

Where the *class_map_name* is the name of the class map.

See Identify Traffic (Layer 3/4 Class Maps), on page 12 to add a class map.

**Example:**

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

**Step 3** Specify one or more actions for this class map.

See Features Configured with Service Policies, on page 3.

**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.

**Step 4** Repeat the process for each class map you want to include in this policy map.

### Examples

The following is an example of a **policy-map** command for a 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.

# Apply 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. Use the following command:

**service-policy** *policy_map_name* {**global** | **interface** *interface_name*} [**fail-close**]

Where:

- *policy_map_name* is the name of the policy map.
- **global** creates a service policy that applies to all interfaces that do not have a specific policy.

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**.

- **interface** *interface_name* creates a service policy by associating a policy map with an interface.
- **fail-close** generates a syslog (767001) for IPv6 traffic that is dropped by application inspections that do not support IPv6 traffic. By default, syslogs are not generated.

### 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.

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

# Monitoring Service Policies

To monitor service policies, enter the following command:

- **show service-policy**

  Displays the service policy statistics.

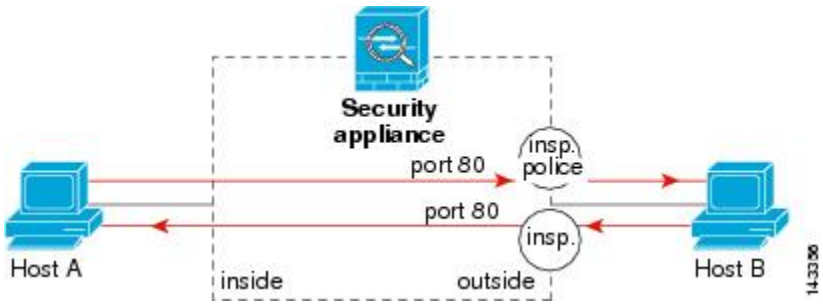
# Examples for Service Policies (Modular Policy Framework)

This section includes several Modular Policy Framework examples.

## Applying Inspection and QoS Policing to HTTP Traffic

In this example, 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: HTTP Inspection and QoS Policing*



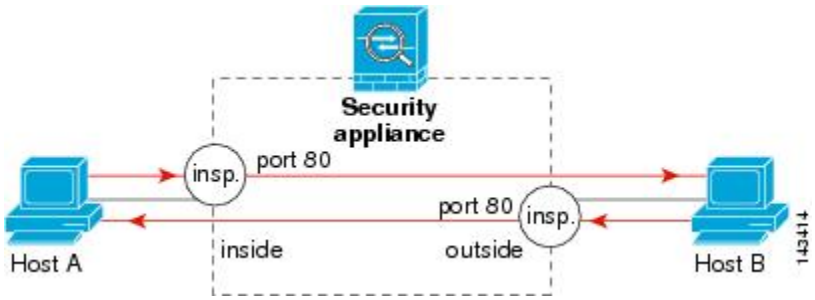

See the following commands for this example:

```
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, 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 2: Global HTTP Inspection*




See the following commands for this example:

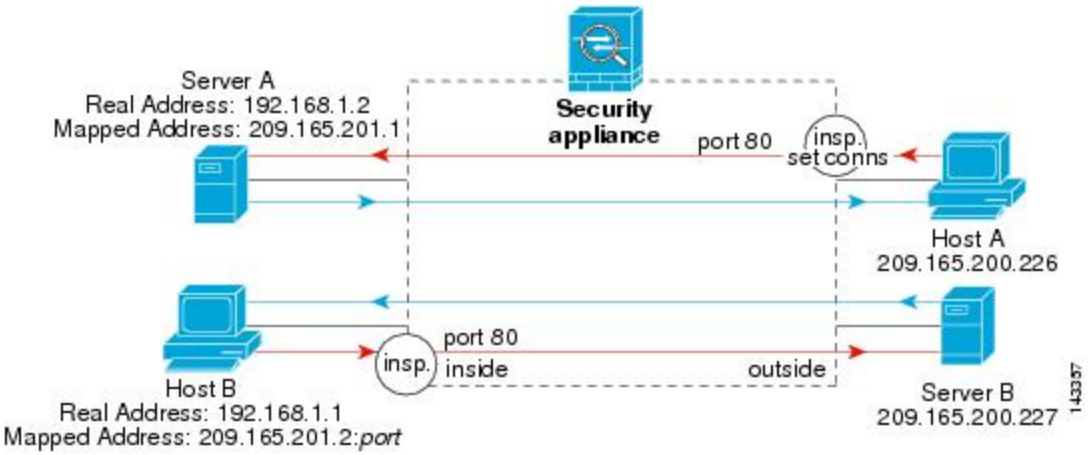
```
hostname(config)# class-map http_traffic
hostname(config-cmap)# match port tcp eq 80

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, 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 do not match the ACL in the class map, so they are 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 do not match the ACL in the class map, so they are not affected.

***Figure 3: HTTP Inspection and Connection Limits to Specific Servers***



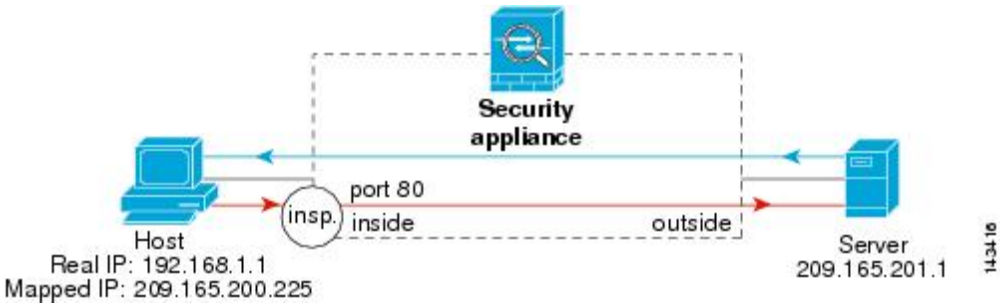

See the following commands for this example:

```
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 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
```

# History for Service Policies

| Feature Name | Releases | Description |
|---|---|---|
| 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**. |

| Feature Name | Releases | Description |
|---|---|---|
| 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. |