



Using Modular Policy Framework

This chapter describes how to use Modular Policy Framework to create security policies for TCP and general connection settings and inspections.

This chapter includes the following sections:

- [Modular Policy Framework Overview, page 18-1](#)
- [Identifying Traffic Using a Class Map, page 18-2](#)
- [Defining Actions Using a Policy Map, page 18-3](#)
- [Applying a Policy to an Interface Using a Service Policy, page 18-6](#)
- [Modular Policy Framework Examples, page 18-7](#)

Modular Policy Framework Overview

Modular Policy Framework provides a consistent and flexible way to configure FWSM features in a manner similar to Cisco IOS software QoS CLI. For example, you can use Modular Policy Framework to create a timeout configuration that is specific to a particular TCP application, as opposed to one that applies to all TCP applications.

Modular Policy Framework is supported with these features:

- TCP connection limits and timeouts
- Application inspection

Configuring Modular Policy Framework consists of three tasks:

1. Identify the traffic to which you want to apply actions. See [“Identifying Traffic Using a Class Map” section on page 18-2](#).
2. Apply actions to the traffic. See [“Defining Actions Using a Policy Map” section on page 18-3](#).
3. Activate the actions on an interface. See [“Applying a Policy to an Interface Using a Service Policy” section on page 18-6](#).

Default Global Policy

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

The default policy configuration includes the following commands:

```
class-map inspection_default
 match default-inspection-traffic
policy-map global_policy
 class inspection_default
  inspect dns maximum-length 512
  inspect ftp
  inspect h323 h225
  inspect h323 ras
  inspect rsh
  inspect smtp
  inspect sqlnet
  inspect skinny
  inspect sunrpc
  inspect xdmcp
  inspect sip
  inspect netbios
  inspect tftp
service-policy global_policy global
```

Identifying Traffic Using a Class Map

A class map identifies traffic to which you want to apply actions. The maximum number of class maps is 255 in single mode or per context in multiple mode. The configuration includes a default class map that the FWSM uses in the default global policy. It is called **inspection_default** and matches the default inspection traffic:

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

To define a class map, perform the following steps:

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

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

where *class_map_name* is a string up to 40 characters in length.

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—You match the class to all traffic.
hostname(config-cmap)# **match any**
- Access list—You can match the class to traffic specified by an extended access list. If the FWSM is operating in transparent firewall mode, you can use an EtherType access list.

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

For more information about creating access lists, see the [“Adding an Extended Access List”](#) section on page 10-5 or the [“Adding an EtherType Access List”](#) section on page 10-8.

For information about creating access lists with NAT, see the [“IP Addresses Used for Access Lists When You Use NAT”](#) section on page 10-3.

- TCP or UDP destination ports—You can match the class to 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 D-12.

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—You can match the class to the traffic that the FWSM inspects by default.

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

See the “[Application Engine Defaults](#)” section on page 20-5 for a list of default ports. The FWSM includes a default global policy that matches the default inspection traffic, and applies common inspections to the traffic on all interfaces. 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 or protocols in the access list are ignored.

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)# exit
hostname(config)# 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)# exit
hostname(config)# 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)# exit
hostname(config)# 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
hostname(config-cmap)# exit
```

Defining Actions Using a Policy Map

This section describes how to associate actions with class maps by creating a policy map. This section includes the following topics:

- [Policy Map Overview, page 18-4](#)
- [Default Policy Map, page 18-4](#)
- [Adding a Policy Map, page 18-5](#)

Policy Map Overview

You can identify multiple class maps in a policy map, and you can assign multiple actions from one or more feature types to each class map. Feature types include the following:

- TCP connection limits and timeouts
- Application inspection

A packet can match only one class map in the policy map for each feature type. When the packet matches a class map for a feature type, the FWSM does not attempt to match it to any subsequent class maps for that feature type. If the packet matches a subsequent class map for a different feature type, however, then the FWSM also applies the actions for the subsequent class map.

For example, if a packet matches a class map for connection limits, and also matches a class map for application inspection, then both class map actions are applied. If a packet matches a class map for application inspection, but also matches another class map for application inspection, then the second class map actions are not applied.

Actions are applied only to traffic that enters the interface to which you apply the policy map.

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:

- TCP connection limits and timeouts
- Application inspection

You can only assign one policy map per interface, but you can apply the same policy map to multiple interfaces.

Default Policy Map

The configuration includes a default policy map that the FWSM uses in the default global policy. It is called **global_policy** and performs inspection on the default inspection traffic. 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 policy map configuration includes the following commands:

```
policy-map global_policy
class inspection_default
  inspect dns maximum-length 512
  inspect ftp
  inspect h323 h225
  inspect h323 ras
  inspect rsh
  inspect smtp
  inspect sqlnet
  inspect skinny
  inspect sunrpc
  inspect xdmcp
  inspect sip
  inspect netbios
  inspect tftp
```

Adding a Policy Map

To create a policy map, perform the following steps:

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

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

Step 2 (Optional) Specify a description for the policy map:

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

Step 3 Specify a previously configured class maps using the following command:

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

See the “[Identifying Traffic Using a Class Map](#)” section on page 18-2 to add a class map.

Step 4 Specify one or more actions for this class map.

- Connection limits. See the “[Configuring Connection Limits and Timeouts](#)” section on page 19-3.
- Application inspection. See [Chapter 20, “Applying Application Layer Protocol Inspection.”](#)



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 5 Repeat [Step 4](#) for each class map you want to include in this policy map.

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 FWSM does not make this match because they previously matched other classes.

Applying a Policy to an Interface Using a Service Policy

To activate the policy map, create a service policy that applies it to one or more interfaces or that applies it globally to all interfaces. Interface service policies take precedence over the global service policy.

- 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
```

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 FWSM interfaces:

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

Modular Policy Framework Examples

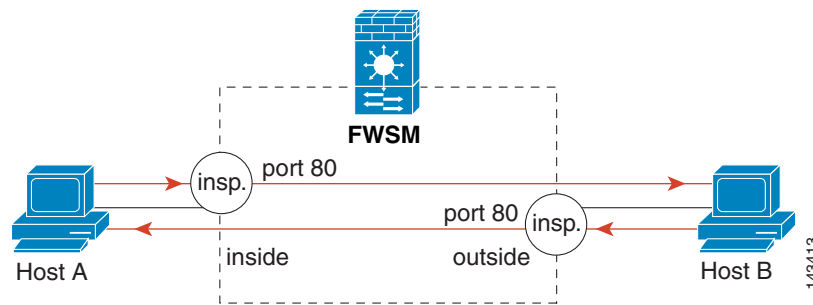
This section includes several Modular Policy Framework examples, and includes the following topics:

- [Applying Inspection to HTTP Traffic Globally, page 18-7](#)
- [Applying Inspection and Connection Limits to HTTP Traffic to Specific Servers, page 18-8](#)
- [Applying Inspection to HTTP Traffic with NAT, page 18-9](#)

Applying Inspection to HTTP Traffic Globally

In this example (see [Figure 18-1](#)), any HTTP connection (TCP traffic on port 80) that enters the FWSM through any interface is classified for HTTP inspection.

Figure 18-1 Global HTTP Inspection



See the following commands for this example:

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

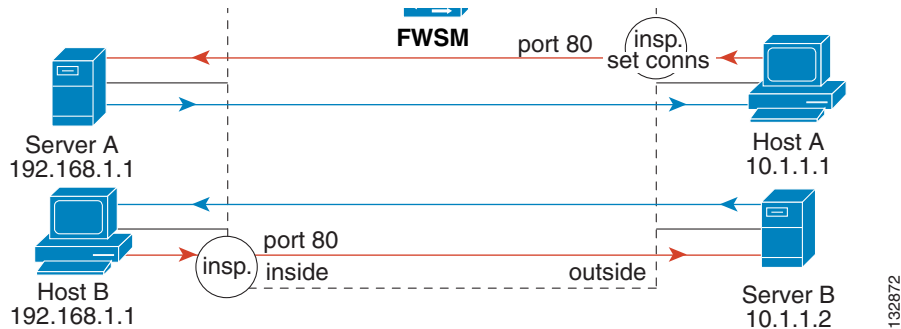
hostname(config)# policy-map http_traffic_policy
hostname(config-pmap)# class http_traffic
hostname(config-pmap-c)# inspect http
hostname(config)# service-policy http_traffic_policy global
```

Applying Inspection and Connection Limits to HTTP Traffic to Specific Servers

In this example (see Figure 18-2), any HTTP connection destined for Server A (TCP traffic on port 80) that enters the FWSM through the outside interface is classified for HTTP inspection and maximum connection limits.

Any HTTP connection destined for Server B that enters the FWSM through the inside interface is classified for HTTP inspection.

Figure 18-2 HTTP Inspection and Connection Limits to Specific Servers



See the following commands for this example:

```
hostname(config)# access-list serverA extended permit tcp any host 192.168.1.1 eq 80
hostname(config)# access-list ServerB extended permit tcp any host 10.1.1.2 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 http_map_serverA
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 http_map_serverB

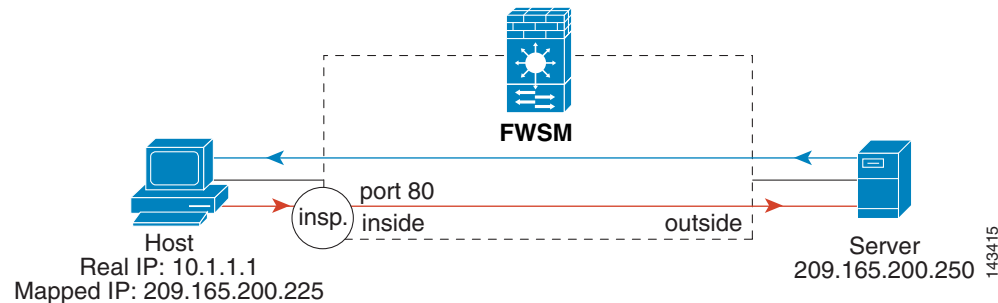
hostname(config)# service-policy policy_serverB interface inside
hostname(config)# service-policy policy_serverA interface outside
```

132872

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 10.1.1.1, 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 addresses.

Figure 18-3 HTTP Inspection with NAT



See the following commands for this example:

```
hostname(config)# static (inside,outside) 209.165.200.225 10.1.1.1
hostname(config)# access-list http_client extended permit tcp host 10.1.1.1 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
```

