# Policy Map Class Configuration Mode Commands

To configure a service policy in an AppNav or optimization policy map, use the **class** policy map configuration command. To unconfigure settings, use the **no** form of this command.

**class** *classmap-name* [**insert-before** [*existing_class*]]

**no class** *classmap-name* [**insert-before** [*existing_class*]]

## Syntax Description

| | |
|---|---|
| *classmap-name* | Class map name (up to 40 alpha-numeric characters and hyphen, beginning with a letter). |
| **insert-before** *existing_class* | Inserts a new class, or moves an existing class, before the specified class. If you do not specify an existing class name, the class is moved to the last position in the policy map. |

## Defaults

No default behavior or values.

## Command Modes

Policy map configuration

## Device Modes

application-accelerator

appnav-controller

## Usage Guidelines

Use the **class** command to add or modify a service policy (policy rule) for traffic identified by a class map. This command invokes the Policy Map Class configuration mode, which is indicated by a different prompt (config-pmap-c). To return to global configuration mode, enter the **exit** command.

You can delete a policy rule by using the **no** form of this command.

The WAAS software comes with many class maps and policy rules that help your WAAS system classify and optimize some of the most common traffic on your network. Before you create a new class map or policy rule, we recommend that you review the default class map and policy rules and modify them as appropriate. It is usually easier to modify an existing class map or policy rule than to create a new one. For a list of the default applications, class maps, and policy rules, see the *Cisco Wide Area Application Services Configuration Guide*.

**Note** We strongly recommend that you use the WAAS Central Manager GUI to centrally configure policy maps for your WAAS devices. For more information, see the *Cisco Wide Area Application Services Configuration Guide*.

## Examples

The following example shows how to configure an AppNav policy rule in a policy map:

```
wae(config)# policy-map type appnav mypolicy
wae(config-pmap)# class httpx
wae(config-pmap-c)# distribute service-node-group wng3
wae(config-pmap-c)# monitor-load http
```

The following example shows how to configure a policy rule in an optimization policy map:

```
wae(config)# policy-map waas WAAS-GLOBAL
wae(config-pmap)# class httpx
wae(config-pmap-c)# optimize full accelerate http application Web
```

## Related Commands

**(config-pmap-c) distribute**

**(config-pmap-c) monitor-load**

**(config-pmap-c) optimize**

**(config-pmap-c) pass-through**

**(config-pmap-c) service-policy**

**(config-pmap-c) set ip dscp**

# (config-pmap-c) distribute

To configure the WAAS node group to which to distribute traffic in an AppNav policy rule, use the **distribute** policy class map configuration command. To unconfigure the distribution policy, use the **no** form of this command.

**distribute service-node-group** *node-group-name* [**insert-before** [*existing-node-group*]]

**no distribute service-node-group** *node-group-name* [**insert-before** [*existing-node-group*]]

## Syntax Description

| | |
|---|---|
| **service-node-group** *node-group-name* | Name of a configured service node group (WAAS node group) to which you want to distribute traffic in the class. |
| **insert-before** *existing-node-group* | Inserts the new distribute action before an existing distribute action to the specified service node group. |

## Defaults

No default behavior or values.

## Command Modes

Policy map class configuration

## Device Modes

appnav-controller

## Usage Guidelines

Use this command to add a distribute action to an AppNav policy rule. One or two distribute actions are allowed in an AppNav policy rule, which specify primary and secondary WAAS node groups to which traffic in the class is to be distributed.

If one distribute action already exists, you can use the **insert-before** option to add the new distribute action before the existing one. If you do not specify the **insert-before** option, the new distribute action is placed after the existing one.

If you specify the distribute action in a policy, any pass-through action is removed.

## Examples

The following example shows how to configure traffic distribution in a policy rule:

```
wae(config)# policy-map type appnav mypolicy
wae(config-pmap)# class httpx
wae(config-pmap-c)# distribute service-node-group wng3
```

## Related Commands

**(config-pmap-c) monitor-load**

**(config-pmap-c) pass-through**

**(config-pmap-c) service-policy**

# (config-pmap-c) monitor-load

To configure the application accelerator to monitor in an AppNav policy rule, use the **monitor-load** policy class map configuration command. To unconfigure monitoring, use the **no** form of this command.

**monitor-load** {**cifs** | **http** | **ica** | **mapi** | **MS-port-mapper** | **nfs** | **ssl** | **video**}

**no monitor-load** {**cifs** | **http** | **ica** | **mapi** | **MS-port-mapper** | **nfs** | **ssl** | **video**}

## Syntax Description

| | |
|---|---|
| **monitor-load** {**cifs** \| **http** \| **ica** \| **mapi** \| **MS-port-mapper** \| **nfs** \| **ssl** \| **video**} | Monitors the load on the specified application accelerator, as follows:<br>• **cifs**—CIFS or SMB accelerator<br>• **http**—HTTP accelerator<br>• **ica**—ICA accelerator<br>• **mapi**—MAPI accelerator<br>• **MS-port-mapper**—EPM accelerator<br>• **nfs**—NFS accelerator<br>• **ssl**—SSL accelerator<br>• **video**—Video accelerator |

## Defaults

No default behavior or values.

## Command Modes

Policy map class configuration

## Device Modes

appnav-controller

## Usage Guidelines

Use this command to monitor the load on an application accelerator in an AppNav policy rule. When you monitor an application accelerator, the AppNav Controller checks for overload on that application accelerator and does not send new flows to a WAAS node that is overloaded.

If you specify the monitor-load action in a policy, any pass-through action is removed.

## Examples

The following example shows how to configure monitoring in a policy rule:

```
wae(config)# policy-map type appnav mypolicy
wae(config-pmap)# class httpx
wae(config-pmap-c)# monitor-load http
```

## Related Commands

**(config-pmap-c) distribute**

**(config-pmap-c) pass-through**

## (config-pmap-c) service-policy

# (config-pmap-c) optimize

To configure optimization actions in a WAAS optimization policy, use the **optimize** policy class map configuration command. To unconfigure optimization actions, use the **no** form of this command.

**optimize** {**tfo-only** | {[**DRE** {**bidirectional** | **adaptive** | **unidirectional**}] [**LZ**] | **full**} [**accelerate** {**cifs** | **http** | **ica** | **mapi** | **MS-port-mapper** | **nfs** | **ssl** | **video**}] [**application** *app-name*]

**no optimize** {**tfo-only** | {[**DRE** {**bidirectional** | **adaptive** | **unidirectional**}] [**LZ**] | **full**} [**accelerate** {**cifs** | **http** | **ica** | **mapi** | **MS-port-mapper** | **nfs** | **ssl** | **video**}] [**application** *app-name*]

## Syntax Description

| | |
|---|---|
| **tfo-only** | Optimize with transport flow optimizations (TFO) and not data redundancy elimination (DRE) or Lempel-Ziv (LZ) compression. |
| **DRE** | Optimize with DRE of the specified type. |
| **bidirectional** | Optimize with bidirectional caching DRE. |
| **adaptive** | Optimize with adaptive caching DRE. |
| **unidirectional** | Optimize with unidirectional caching DRE. |
| **LZ** | Apply LZ compression. |
| **full** | Apply full Layer 4 optimization; this keyword is equivalent to **DRE bidirectional LZ**. |
| **accelerate** {**cifs** \| **http** \| **ica** \| **mapi** \| **MS-port-mapper** \| **nfs** \| **ssl** \| **video**} | Accelerate the traffic using the specified application accelerator, as follows:<br>• **cifs**—CIFS or SMB accelerator<br>• **http**—HTTP accelerator<br>• **ica**—ICA accelerator<br>• **mapi**—MAPI accelerator<br>• **MS-port-mapper**—EPM accelerator<br>• **nfs**—NFS accelerator<br>• **ssl**—SSL accelerator<br>• **video**—Video accelerator |
| **application** *app-name* | Assign the specified application identifier to connections matching the class for statistics collection. |

## Defaults

No default behavior or values.

## Command Modes

Policy map class configuration

## Device Modes

application-accelerator

appnav-controller

## Usage Guidelines

This command configures the optimization actions in a WAAS optimization policy.

You may specify only a single **optimize** or **pass-through** action for a particular class. If one of these actions is already present and you specify a new action, the new action replaces the existing action. If neither of these actions is specified, the default is **pass-through**.

The following DRE caching modes are supported:

- Bidirectional—The peer WAEs maintain identical caches for inbound and outbound traffic. This caching mode is best suited where a significant portion of the traffic seen in one direction between the peers is also seen in the reverse direction.
- Unidirectional—The peer WAEs maintain different caches for inbound and outbound traffic. This caching mode is best suited where a significant portion of the traffic seen in one direction between the peers is not seen in the reverse direction.
- Adaptive—The peer WAEs negotiate either bidirectional or unidirectional caching based on the characteristics of the traffic seen between the peers.

## Examples

The following example shows how to configure the optimization action in a policy:

```
wae(config)# policy-map waas WAAS-GLOBAL
wae(config-pmap)# class httpx
wae(config-pmap-c)# optimize full accelerate http application Web
```

## Related Commands

**(config-pmap-c) pass-through**

**(config-pmap-c) set ip dscp**

# (config-pmap-c) pass-through

To configure the pass-through action in an AppNav or optimization policy rule, use the **pass-through** policy class map configuration command. To unconfigure the pass-through action, use the **no** form of this command.

**pass-through** [**application** *app-name*]

**no pass-through** [**application** *app-name*]

## Syntax Description

| | |
|---|---|
| **application** *app-name* | (Optional) Assign the specified application identifier to connections matching the class for statistics collection. Available only for WAAS optimization class maps. |

## Defaults

No default behavior or values.

## Command Modes

Policy map class configuration

## Device Modes

application-accelerator

appnav-controller

## Usage Guidelines

In an optimization policy, this command prevents the traffic in the class from being optimized and instead the traffic is passed through unoptimized. You can optionally specify an application name with which to associate the traffic in the class for statistics collection purposes.

You may specify only a single **optimize** or **pass-through** action for a particular class. If one of these actions is already present and you specify a new action, the new action replaces the existing action. If neither of these actions is specified, the default is **pass-through**.

In an AppNav policy rule, this command prevents the traffic in the class from being distributed to WAAS nodes and instead the traffic is passed through unoptimized.

This command is useful in a nested policy to override a distribute action specified in the parent policy.

If you specify the pass-through action in a policy, any distribute or monitor-load actions are removed.

## Examples

The following example shows how to configure the pass-through action in an optimization policy:

```
wae(config)# policy-map waas WAAS-GLOBAL
wae(config-pmap)# class httpx
wae(config-pmap-c)# pass-through
```

The following example shows how to configure the pass-through action in an AppNav policy rule:

```
wae(config)# policy-map type appnav mypolicy
wae(config-pmap)# class httpx
wae(config-pmap-c)# pass-through
```

**Related Commands**

(config-pmap-c) distribute

(config-pmap-c) monitor-load

(config-pmap-c) optimize

(config-pmap-c) service-policy

(config-pmap-c) set ip dscp

# (config-pmap-c) service-policy

To configure a nested policy map in an AppNav policy rule, use the **service-policy** policy class map configuration command. To unconfigure a nested policy map, use the **no** form of this command.

**service-policy** *policymap-name*

**no service-policy** *policymap-name*

## Syntax Description

| | |
|---|---|
| *policymap-name* | Name of an existing policy map to nest under this policy rule. |

## Defaults

No default behavior or values.

## Command Modes

Policy map class configuration

## Device Modes

appnav-controller

## Usage Guidelines

This command specifies another policy map that is applied as a nested policy to the traffic in the class. Policy rules in the nested policy map override policy rules in the parent policy map.

Only one nested policy is allowed in a class. If you specify this command when an existing nested policy exists in the class, the new nested policy replaces the existing one.

You can specify the same nested policy in multiple classes, which allows you to apply the same nested policy action, such as monitoring application accelerators, to many classes of traffic.

A policy map used as a nested policy may not contain classes that use match peer conditions.

## Examples

The following example shows how to configure nested policy map for a policy rule:

```
wae(config)# policy-map type appnav mypolicy
wae(config-pmap)# class httpx
wae(config-pmap-c)# distribute service-node-group wng3
wae(config-pmap-c)# service-policy npolicymonitor
```

## Related Commands

**(config-pmap-c) distribute**

**(config-pmap-c) monitor-load**

**(config-pmap-c) pass-through**

# (config-pmap-c) set ip dscp

To configure the DSCP marking in a WAAS optimization policy, use the **set ip dscp** policy class map configuration command. To unconfigure DSCP marking, use the **no** form of this command.

**set ip dscp** *dscp-marking*

**no set ip dscp** *dscp-marking*

## Syntax Description

| | |
|---|---|
| *dscp-marking* | Assign the specified DSCP marking value (Table 3-2) to the connections in the class. |

## Defaults

The default DSCP marking value is copy.

## Command Modes

Policy map class configuration

## Device Modes

application-accelerator

appnav-controller

## Usage Guidelines

This command overrides the global default DSCP marking value, which is set to copy by default.

If you do not specify the **set ip dscp** command, the class uses the global default DSCP marking, which is set by the **service-policy type waas set ip dscp** command.

You can specify the **set ip dscp** command only when the **optimize** action has been configured for a class.

## Examples

The following example shows how to configure the DSCP marking value for connections in the class:

```
wae(config)# policy-map waas WAAS-GLOBAL
wae(config-pmap)# class httpx
wae(config-pmap-c)# optimize full accelerate http application Web
wae(config-pmap-c)# set ip dscp 10
```

## Related Commands

**(config-pmap-c) optimize**

**(config-pmap-c) pass-through**

**(config) service-policy**