This chapter describes how to configure TCP/IP normalization and termination parameters to protect your Cisco Application Control Engine (ACE) module and the data center from attacks. It also describes IP fragmentation and reassembly parameters. The chapter contains the following major sections:

- TCP Normalization Overview
- IP Normalization Overview
- TCP/IP Normalization and Termination Configuration Quick Start
- Configuring a Connection Parameter Map for TCP/IP Normalization and Termination
- Configuring a Traffic Policy for TCP/IP Normalization and Termination
- Configuring Interface Normalization Parameters
- Configuring IP Fragment Reassembly Parameters
- Configuring the Switch Mode Feature
- Example of a TCP/IP Normalization and IP Reassembly Configuration
- Displaying Configurations and Statistics for TCP/IP and UDP Connections, IP Reassembly, and SYN Cookie
- Clearing TCP/IP and UDP Connections and Statistics
TCP Normalization Overview

This section describes how the ACE uses TCP normalization to protect itself and the data center from a variety of network-based attacks.

TCP normalization is a Layer 4 feature that consists of a series of checks that the ACE performs at various stages of a flow, from the initial connection setup to the closing of a connection. You can control many of the segment checks by configuring one or more advanced TCP connection settings. The ACE uses these TCP connection settings to decide which checks to perform and whether to discard a TCP segment based on the results of the checks. The ACE discards segments that appear to be abnormal or malformed.

With TCP normalization, the ACE checks for segments that have invalid or suspect conditions (for example, a SYN sent to the client from the server or a SYNACK sent to the server from the client) and takes actions based on the configured parameter settings. The ACE uses TCP normalization to block certain types of network attacks (for example, insertion attacks and evasion attacks). Insertion attacks occur when the inspection module accepts a packet that the end system rejects. Evasion attacks occur when the inspection module rejects a packet while the end system accepts it.

The ACE always discards segments when the following conditions exist:

- Bad segment checksum
- Bad TCP header or payload length
- Suspect TCP flags (for example, NULL, SYN/FIN, or FIN/URG)

TCP normalization is enabled by default. If you are migrating to, or replacing legacy products with, the ACE, disable normalization using the `no normalization` command in interface configuration mode until you are sure that everything is working properly. Then, reenable normalization using the `normalization` command in interface configuration mode.

To configure TCP normalization on the ACE, you assemble various TCP commands into a parameter map. After you create the connection parameter map, you associate it with a multi-match policy map, and activate the traffic policy globally across all interfaces in the context using a service policy. For details about configuring traffic policies, see the “Configuring a Traffic Policy for TCP/IP Normalization and Termination” section.
**IP Normalization Overview**

In addition to TCP normalization, the ACE uses a Layer 3 feature called IP normalization to protect itself and the data center from a variety of attacks. IP normalization performs the following series of checks on IP packets:

- General security checks
- ICMP security checks
- Fragmentation security checks
- IP fragment reassembly
- IP fragmentation if a packet exceeds the outbound maximum transmission unit (MTU)

If a packet fails one of these checks, the ACE takes action (including discarding a packet) depending on the IP parameters that you configure.

To configure the type of service (ToS) for IP traffic, use the `set ip tos` command in a connection parameter map.

To configure interface-related IP normalization parameters, see the “Configuring Interface Normalization Parameters” section.

**TCP/IP Normalization and Termination Configuration Quick Start**

Table 4-1 provides a quick overview of the steps required to configure TCP normalization. Each step includes the CLI command or a reference to the procedure required to complete the task. For a complete description of each feature and all the options associated with the CLI commands, see the sections following Table 4-1.
Chapter 4  Configuring TCP/IP Normalization and IP Reassembly Parameters

TCP/IP Normalization and Termination Configuration Quick Start

Table 4-1  TCP/IP Normalization and Termination Configuration Quick Start

<table>
<thead>
<tr>
<th>Task and Command Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. If you are operating in multiple contexts, observe the CLI prompt to verify that you are operating in the desired context. If necessary, change to the correct context.</td>
</tr>
<tr>
<td>host1/Admin# change to C1</td>
</tr>
<tr>
<td>host1/C1#</td>
</tr>
<tr>
<td>The rest of the examples in this table use the C1 user context, unless otherwise specified. For details on creating contexts, see the Cisco Application Control Engine Module Virtualization Configuration Guide.</td>
</tr>
<tr>
<td>2. Enter configuration mode.</td>
</tr>
<tr>
<td>host1/C1# config</td>
</tr>
<tr>
<td>host1/C1(config)#</td>
</tr>
<tr>
<td>3. Create a connection parameter map to group together TCP/IP normalization and termination parameters.</td>
</tr>
<tr>
<td>host1/C1(config)# parameter-map type connection TCPIP_PARAM_MAP</td>
</tr>
<tr>
<td>host1/C1(config-parammap-conn)#</td>
</tr>
<tr>
<td>4. Configure TCP/IP normalization parameters in the connection parameter map as required. For example, enter:</td>
</tr>
<tr>
<td>host1/C1(config-parammap-conn)# set timeout inactivity 2400</td>
</tr>
<tr>
<td>host1/C1(config-parammap-conn)# set ip tos 20</td>
</tr>
<tr>
<td>host1/C1(config-parammap-conn)# exit</td>
</tr>
<tr>
<td>host1/C1(config)#</td>
</tr>
<tr>
<td>5. Create a Layer 3 and Layer 4 TCP class map, and then configure match criteria as required.</td>
</tr>
<tr>
<td>host1/C1(config)# class-map match-any TCP_CLASS</td>
</tr>
<tr>
<td>host1/C1(config-cmap)# match destination-address 172.27.16.7</td>
</tr>
<tr>
<td>host1/C1(config-cmap)# match port tcp eq 21</td>
</tr>
<tr>
<td>host1/C1(config-cmap)# exit</td>
</tr>
<tr>
<td>6. Create a Layer 3 and Layer 4 policy map and associate the class map with it.</td>
</tr>
<tr>
<td>host1/C1(config)# policy-map multi-match TCPIP_POLICY</td>
</tr>
<tr>
<td>host1/C1(config-pmap)# class TCP_CLASS</td>
</tr>
<tr>
<td>host1/C1(config-pmap-c)# exit</td>
</tr>
<tr>
<td>host1/C1(config-pmap)# exit</td>
</tr>
</tbody>
</table>
Table 4-1  TCP/IP Normalization and Termination Configuration Quick Start (continued)

<table>
<thead>
<tr>
<th>Task and Command Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>7. Associate the connection parameter map as an action in the TCP/IP policy map.</td>
</tr>
<tr>
<td>host1/C1(config-pmap-c)# connection advanced-options TCPIP_PARAM_MAP</td>
</tr>
<tr>
<td>host1/C1(config-pmap-c)# exit</td>
</tr>
<tr>
<td>host1/C1(config-pmap)# exit</td>
</tr>
<tr>
<td>8. Apply the policy map globally across all interfaces in the context using a service policy.</td>
</tr>
<tr>
<td>host1/C1(config)# interface vlan 50</td>
</tr>
<tr>
<td>host1/C1(config-if)# service-policy input TCPIP_POLICY</td>
</tr>
<tr>
<td>host1/C1(config-if)# exit</td>
</tr>
<tr>
<td>9. Configure additional IP normalization parameters in interface configuration mode.</td>
</tr>
<tr>
<td>host1/C1(config-if)# ip ttl 15</td>
</tr>
<tr>
<td>host1/C1(config-if)# ip options clear</td>
</tr>
<tr>
<td>host1/C1(config-if)# ip df allow</td>
</tr>
<tr>
<td>host1/C1(config-if)# exit</td>
</tr>
<tr>
<td>host1/C1(config)# exit</td>
</tr>
<tr>
<td>10. (Optional) Save your configuration changes to flash memory.</td>
</tr>
<tr>
<td>host1/C1# copy running-config startup-config</td>
</tr>
<tr>
<td>11. Display the TCP/IP normalization configuration information.</td>
</tr>
<tr>
<td>host1/C1# show running-config policy-map</td>
</tr>
<tr>
<td>host1/C1# show running-config parameter-map</td>
</tr>
<tr>
<td>host1/C1# show running-config interface</td>
</tr>
<tr>
<td>host1/C1# show service-policy name</td>
</tr>
</tbody>
</table>
Configuring a Connection Parameter Map for TCP/IP Normalization and Termination

You can configure a parameter map to group TCP/IP connection-related commands that apply to normalization and termination. After you configure the parameter map, associate it with a specific action statement in a policy map using the `connection tcp advanced-options` command. For details about associating a parameter map with a policy map, see the “Associating a Connection Parameter Map with a Policy Map” section. This section contains the following topics:

- Creating a Connection Parameter Map for TCP/IP, UDP, and ICMP
- Configuring Rate Limits for a Policy Map
- Setting the Maximum Receive or Transmit Buffer Share
- Setting a Range for the Maximum Segment Size
- Configuring ACE Behavior for a Segment That Exceeds the Maximum Segment Size
- Setting the Maximum Number of TCP SYN Retries
- Enabling Nagle’s Algorithm
- Enabling Random TCP Sequence Numbers
- Configuring How the ACE Handles Reserved Bits
- Configuring the Timeout for an Embryonic Connection
- Configuring the Timeout for a Half-Closed Connection
- Configuring the Connection Inactivity Timeout
- Setting How the ACE Applies TCP Optimizations to Packets
- Setting the Window Scale Factor
- Enabling the TCP Slow Start Algorithm
- Setting the ACK Delay Timer
- Configuring How the ACE Handles TCP SYN Segments that Contain Data
- Configuring How the ACE Handles TCP Options
- Setting the Urgent Pointer Policy
- Setting the Type of Service
Creating a Connection Parameter Map for TCP/IP, UDP, and ICMP

You can create a connection parameter map for TCP/IP, UDP, and ICMP by using the parameter-map type connection command in configuration mode. The syntax of this command is as follows:

```
parameter-map type connection map_name
```

The `map_name` argument is a unique name as an unquoted text string with no spaces with a maximum of 64 alphanumeric characters.

For example, to create a connection parameter map, enter:

```
host1/C1(config)# parameter-map type connection TCPIP_PARAM_MAP
host1/C1(config-parammap-conn)#
```

To remove the connection parameter map from the configuration, enter:

```
host1/C1(config)# no parameter-map type connection TCPIP_PARAM_MAP
```

Use one or more of the commands in the sections that follow to define the connection parameter map.

To limit the maximum number of ACE connections, create a resource class and then use the following commands:

- Through-the-ACE connections—`limit-resource conc-connections`
- To-the-ACE connections—`limit-resource mgmt-connections`

Make sure that you assign the current context to the resource class. For details about resource classes, see the Cisco Application Control Engine Module Virtualization Configuration Guide.

Note

If you configure switch mode and you configure any connection parameter-map commands (for example, `set tcp buffer-share, rate-limit, exceed-mss, nagle, random-sequence-number, reserved-bits, set tcp wan-optimization, set timeout inactivity, slowstart`, and so on) either locally on a specific interface or globally on all interfaces, then switch mode will override these commands for certain types of traffic. This behavior applies only to non-load-balanced (VIP), non-inspection, non-NATed, and non-management traffic. The ACE continues to apply local, global, and VIP-specific connection parameter maps to load-balanced (VIP), inspected, NATed, and management traffic. For details about switch mode, see the “Configuring the Switch Mode Feature” section.
Configuring Rate Limits for a Policy Map

The ACE allows you to limit the connection rate and the bandwidth rate of a policy map. The connection rate is the number of connections per second that match the policy. The bandwidth rate is the number of bytes per second that match the policy. The ACE applies these rate limits to each class map that you associate with the policy at the virtual server level.

When the connection-rate limit or the bandwidth-rate limit is reached, the ACE blocks any further traffic that matches that policy until the connection rate or bandwidth rate drops below the configured limit. By default, the ACE does not limit the connection rate or the bandwidth rate of a policy.

You can also limit the connection rate and the bandwidth rate of a real server in a server farm. For details, see the Cisco Application Control Engine Module Server Load-Balancing Configuration Guide.

To limit the connection rate or the bandwidth rate of a policy, use the rate-limit command in parameter map connection configuration mode. The syntax of this command is as follows:

```
rate-limit {connection number1 | bandwidth number2}
```

The keywords and arguments are as follows:

- **connection number1**—Specifies the connection-rate limit for a policy in connections per second. Enter an integer from 0 to 350000. There is no default value.

- **bandwidth number2**—Specifies the bandwidth-rate limit for a policy in bytes per second. Enter an integer from 0 to 300000000. There is no default value.

For example, to limit the connection rate of a policy to 100000 connections per second, enter:

```
host1/Admin(config)# parameter-map type connection RATE-LIMIT
host1/Admin(config-parammap-conn)# rate-limit connection 100000
```

To return the behavior of the ACE to the default of not limiting the policy connection rate, enter:

```
host1/Admin(config-parammap-conn)# no rate-limit connection 100000
```

For example, to limit the policy bandwidth rate to 5000000 bytes per second, enter:

```
host1/Admin(config-parammap-conn)# rate-limit bandwidth 5000000
```

To return the behavior of the ACE to the default of not limiting the policy bandwidth rate, enter:

```
host1/Admin(config-parammap-conn)# no rate-limit bandwidth 5000000
```
host1/Admin(config)# parameter-map type connection RATE-LIMIT
host1/Admin(config-parammap-conn)# rate-limit bandwidth 50000000

To return the behavior of the ACE to the default of not limiting the policy bandwidth rate, enter:

host1/Admin(config-parammap-conn)# no rate-limit bandwidth 50000000

**Setting the Maximum Receive or Transmit Buffer Share**

To improve throughput and overall performance, the ACE checks the number of buffered bytes on each TCP and UDP connection against the configured buffer setting before accepting new receive or transmit data. By default, the maximum size of the receive or transmit buffer for each TCP or UDP connection is 32768 bytes. For large bandwidth and delay network connections, you may want to increase the default buffer size to improve your network performance. To set the maximum receive or transmit buffer size for each TCP and UDP connection, use the `set tcp buffer-share` command in parameter map connection configuration mode. The syntax of this command is as follows:

```
set tcp buffer-share number
```

The *number* argument is the maximum size of the receive or transmit buffer in bytes for each TCP and UDP connection. Enter an integer from 8192 to 262143 bytes. The default is 32768 bytes.

---

**Caution**

If you are using the ACE to terminate SSL traffic, do not decrease the buffer share value below the default value of 32 KB. With a buffer share value of less than 32 KB, SSL connections are significantly slower.

---

For example, enter:

host1/C1(config-parammap-conn)# set tcp buffer-share 16384

To reset the buffer limit to the default value of 32768 bytes, enter:

host1/C1(config-parammap-conn)# no set tcp buffer-share
Setting a Range for the Maximum Segment Size

The maximum segment size (MSS) is the largest amount of TCP data that the ACE accepts in one segment. To prevent the transmission of many smaller segments that waste bandwidth or very large segments that may require fragmentation, you can set the minimum and maximum acceptable sizes of the MSS. To set the MSS, use the `set tcp mss` command in parameter map connection configuration mode. The syntax of this command is as follows:

```
set tcp mss min number1 max number2
```

The keywords and arguments are as follows:

- **min number1**—Specifies the smallest segment size that the ACE will accept. Enter an integer from 0 to 65535 bytes. The default is 0 bytes. The `min number` value must be less than or equal to the `max number` value. A value of 0 instructs the ACE to not perform a minimum MSS check on the incoming segment.

- **max number2**—Specifies the largest segment size that the ACE will accept. Enter an integer from 0 to 65535 bytes. The default is 1460 bytes. The `max number` value must be greater than or equal to the `min number` value. A value of 0 instructs the ACE to not perform a maximum MSS check on the incoming segment.

⚠️ **Caution**

If you configure a Layer 7 policy map and set the maximum transmit unit (MTU) of the ACE server-side VLAN lower than the client maximum segment size (MSS), ensure that the maximum value of the MSS that you set for the ACE using the `set tcp mss max` command is at least 40 bytes (the size of the TCP header plus options) less than the MTU of the ACE server-side VLAN. Otherwise, the ACE may discard incoming packets from the server.
Both the host and the server can set the MSS when they first establish a connection. If either maximum exceeds the value that you set with the `set tcp mss max` command, then the ACE overrides the maximum value and inserts the value that you set. If either maximum is less than the value that you set with the `set tcp mss min` command, then the ACE overrides the maximum and inserts the minimum value that you set (the minimum value is actually the smallest maximum allowed). For example, if you set a maximum size of 1200 bytes and a minimum size of 400 bytes, when a host requests a maximum size of 1300 bytes, then the ACE alters the packet to request 1200 bytes (the maximum). If another host requests a maximum value of 300 bytes, then the ACE alters the packet to request 400 bytes (the minimum).

If the host or server does not request an MSS, the ACE assumes that the RFC 793 default value of 536 bytes is in effect.

For example, to set the minimum acceptable MSS size to 768 bytes, and the maximum acceptable MSS size to 1500, enter:

```
host1/C1(config-parammap-conn)# set tcp mss min 768 max 1500
```

To reset the minimum MSS to the default value of 0 bytes and the maximum MSS to the default value of 1460, enter:

```
host1/C1(config-parammap-conn)# no set tcp mss
```

### Configuring ACE Behavior for a Segment That Exceeds the Maximum Segment Size

You can configure the ACE behavior for a segment that exceeds the configured maximum segment size (MSS) by using the `exceed-mss` command in connection parameter map configuration mode. The syntax of this command is as follows:

```
exceed-mss {allow | drop}
```

The keywords are as follows:

- **allow**—Permits segments that exceed the configured MSS
- **drop**—(Default) Discards segments that exceed the configured MSS

For example, to configure the ACE to allow segments that exceed the MSS, enter:

```
host1/C1(config-parammap-conn)# exceed-mss allow
```
To reset the ACE behavior to the default of discarding segments that exceed the MSS set by a peer, enter:

```
host1/C1(config-parammap-conn)# no exceed-mss allow
```

### Setting the Maximum Number of TCP SYN Retries

You can set the maximum number of attempts that the ACE makes to transmit a TCP segment when initiating a Layer 7 connection by using the `set tcp syn-retry` command in connection parameter map configuration mode. The syntax of this command is as follows:

```
set tcp syn-retry number
```

The `number` argument is the number of SYN retries. Enter an integer from 1 to 15. The default is 4.

For example, to set the maximum TCP SYN retries to 3, enter:

```
host1/C1(config-parammap-conn)# set tcp syn-retry 3
```

To reset the TCP SYN retries to the default value of 4, enter:

```
host1/C1(config-parammap-conn)# no set tcp syn-retry
```

### Enabling Nagle’s Algorithm

Nagle’s algorithm instructs a sender to buffer any data to be sent until all outstanding data has been acknowledged or until there is a full segment of data to send. The algorithm automatically concatenates a number of small buffer messages transmitted over the TCP connection. This process increases the throughput by decreasing the number of segments that need to be sent over the network. However, the interaction between the Nagle algorithm and the TCP delay acknowledgment may increase latency in your TCP connection. You should disable the Nagle algorithm when you observe an unacceptable delay in a TCP connection.

You can enable Nagle’s algorithm by using the `nagle` command in parameter map connection configuration mode. By default, this command is disabled. The syntax of this command is as follows:

```
nagle
```
For example, enter:
host1/C1(config-parammap-conn)# nagle

To disable Nagle’s algorithm, enter:
host1/C1(config-parammap-conn)# no nagle

### Enabling Random TCP Sequence Numbers

Randomizing TCP sequence numbers adds a measure of security to TCP connections by making it more difficult for a hacker to guess or predict the next sequence number in a TCP connection. This feature is enabled by default. To enable TCP sequence number randomization after it has been disabled, use the `random-sequence-number` command in parameter map connection configuration mode.

The syntax of this command is as follows:

```
random-sequence-number
```

For example, to enable the use of random sequence numbers if you have disabled the feature, enter:
host1/C1(config-parammap-conn)# random-sequence-number

To disable sequence number randomization, enter:
host1/C1(config-parammap-conn)# no random-sequence-number

**Note**

You cannot disable sequence number randomization for Layer 7 traffic flows.

### Configuring How the ACE Handles Reserved Bits

You can configure how an ACE handles segments with the reserved bits set in the TCP header by using the `reserved-bits` command in parameter map connection configuration mode. The six reserved bits in the TCP header are for future use and usually have a value of 0. The syntax of this command is as follows:

```
reserved-bits {allow | clear | drop}
```
The keywords are as follows:

- **allow**—(Default) Permits segments with the reserved bits set in the TCP header
- **clear**—Clears the reserved bits in the TCP header and allows the segment
- **drop**—Discards segments with reserved bits set in the TCP header

For example, to configure the ACE to clear the reserved bits set in the TCP header of segments, enter:

```
host1/C1(config-parammap-conn)# reserved-bits clear
```

To reset the ACE behavior to the default of allowing reserved bits set in the TCP header of a segment, enter:

```
host1/C1(config-parammap-conn)# no reserved-bits clear
```

## Configuring the Timeout for an Embryonic Connection

Occasionally, the TCP three-way handshake for a connection may not complete for some reason. This type of connection is called an *embryonic* connection. To configure a timeout for embryonic connections, use the `set tcp timeout embryonic` command in parameter map connection configuration mode. The syntax of this command is as follows:

```
set tcp timeout embryonic seconds
```

The `seconds` argument is an integer from 0 to 4294967295 seconds. The default is 5 seconds. A value of 0 specifies that the ACE does not time out an embryonic connection.

**Note**

This command affects only Layer 4 flows and not Layer 7 flows.

For example, enter:

```
host1/C1(config-parammap-conn)# set tcp timeout embryonic 24
```

To reset the TCP embryonic connection timeout to the default value of 5 seconds, enter:

```
host1/C1(config-parammap-conn)# no set tcp timeout embryonic
```
Configuring the Timeout for a Half-Closed Connection

A half-closed connection is a connection in which the client (or server) sends a FIN and the server (or client) ACKs the FIN without sending a FIN itself. The timer starts once this condition has occurred. To configure a timeout for a half-closed connection, use the `set tcp timeout half-closed` command in parameter map connection configuration mode. The syntax of this command is as follows:

```
set tcp timeout half-closed seconds
```

The `seconds` argument is an integer from 0 to 4294967295 seconds. The default is 3600 seconds (1 hour). A value of 0 specifies that the ACE does not time out a half-closed TCP connection.

For example, enter:

```
host1/C1(config-parammap-conn)# set tcp timeout half-closed 2400
```

To reset the TCP half-closed connection timeout to the default value of 3600 seconds, enter:

```
host1/C1(config-parammap-conn)# no set tcp timeout half-closed
```

Configuring the Connection Inactivity Timeout

The ACE uses the connection inactivity timer to disconnect established TCP/IP, UDP, and ICMP connections that have remained idle for the duration of the specified timeout period. To configure the connection inactivity timer, use the `set timeout inactivity` command in parameter map connection configuration mode. The syntax of this command is as follows:

```
set timeout inactivity seconds
```

The `seconds` argument is the time period after which the ACE disconnects idle established connections. Enter an integer from 0 to 3217203 seconds. The defaults are as follows:

- ICMP—2 seconds
- TCP—3600 seconds (1 hour)
- UDP—10 seconds
A value of 0 specifies that the ACE does not time out a TCP connection. The ACE rounds up the value that you enter to the nearest 30-second interval.

For example, to set the connection inactivity timeout to 2400 seconds (40 minutes), enter:

```
host1/C1(config-parammap-conn)# set timeout inactivity 2400
```

To reset the connection inactivity timeout to the default values, enter:

```
host1/C1(config-parammap-conn)# no set timeout inactivity
```

### Setting How the ACE Applies TCP Optimizations to Packets

You can control how the ACE applies TCP optimizations to packets on a connection associated with a Layer 7 policy map using a round-trip time (RTT) value by using the `set tcp wan-optimization rtt` command in parameter map connection configuration mode.

TCP optimizations include the following connection parameter-map configuration mode operations:

- Nagle optimization algorithm (see the “Enabling Nagle’s Algorithm” section)
- Slow-start connection behavior (see the “Enabling the TCP Slow Start Algorithm” section)
- Acknowledgement (ACK) delay timer (see the “Setting the ACK Delay Timer” section)
- Window-scale factor (see the “Setting the Window Scale Factor” section)
- Retry settings (see the “Setting the Maximum Number of TCP SYN Retries” section)

The syntax of this command is as follows:

```
set tcp wan-optimization rtt number
```

The `number` argument is the round-trip time (RTT) in milliseconds and controls how the ACE applies TCP optimizations as follows:

- For a value of 0, the ACE applies TCP optimizations to packets for the life of a connection.
- For a value of 65535 (the default), the ACE performs normal operations (no optimizations) for the life of a connection.
For values from 1 to 65534, the ACE applies TCP optimizations to packets based on the client RTT to the ACE as follows:

- If the actual client RTT is less than the configured RTT, the ACE performs normal operations for the life of the connection.
- If the actual client RTT is greater than or equal to the configured RTT, the ACE performs TCP optimizations on the packets for the life of a connection.

For example, to specify the ACE applies TCP optimizations to packets for the life of a connection, enter:

```
host1/C1(config-parammap-conn)# set tcp wan-optimization rtt 0
```

To restore the ACE behavior to the default of not optimizing TCP connections, enter:

```
host1/C1(config-parammap-conn)# no set tcp wan-optimization rtt 0
```

### Setting the Window Scale Factor

The TCP window scaling feature adds support for the Window Scaling option in RFC 1323. We recommend that you increase the window size to improve TCP performance in network paths with large bandwidth, long-delay characteristics. This type of network is called a long fat network (LFN).

The window scaling extension expands the definition of the TCP window to 32 bits and then uses a scale factor to carry this 32-bit value in the 16-bit window field of the TCP header. You can increase the window size to a maximum scale factor of 14. Typical applications use a scale factor of 3 when deployed in LFNs.

For Layer 7 connections (where the ACE terminates the connection), the ACE forwards the original window scale factor from the client to the server. The ACE sends the window scale factor that you configure in the SYN-ACK to the client. The ACE window scale factor must match the server window scale factor or the `tcp-options` command must be set to clear the window scale option. Otherwise, unexpected results may occur. For more information about the `tcp-options` command, see the “Configuring How the ACE Handles TCP Options” section.

For Secure Sockets Layer (SSL) connections or for configurations where the WAN optimization RTT is set to zero (see the “Setting How the ACE Applies TCP Optimizations to Packets” section), window scale mismatches between the ACE and a server are allowed. For all other connections, the ACE window scale factor must match the server window scale factor.
To set the TCP window scale factor, use the `set tcp window-scale` command in parameter map connection configuration mode. The syntax of this command is as follows:

```
set tcp window-scale number
```

The `number` argument is an integer from 0 to 14. The default is 0.

For example, to set the TCP window scale factor to 3, enter:

```
host1/C1(config-parammap-conn)# set tcp window-scale factor 3
```

To reset to the default value of 0, enter:

```
host1/C1(config-parammap-conn)# no set tcp window-scale
```

### Enabling the TCP Slow Start Algorithm

The slow start algorithm is a congestion avoidance method in which TCP increases its window size as ACK handshakes arrive. It operates by observing that the rate at which new segments should be injected into the network is the rate at which the acknowledgments are returned by the host at the other end of the connection. This feature is disabled by default. For details about the TCP slow start algorithm, see RFC 2581 and 3782.

To enable the slow start algorithm, use the `slowstart` command in parameter map connection configuration mode. The syntax of this command is as follows:

```
slowstart
```

For example, to enable the slow start feature, enter:

```
host1/C1(config-parammap-conn)# slowstart
```

To disable the slow start algorithm, enter:

```
host1/C1(config-parammap-conn)# no slowstart
```
Setting the ACK Delay Timer

You can configure the ACE to delay sending the ACK from a client to a server. Some applications require delaying the ACK for best performance. Delaying the ACK can also help reduce congestion by sending one ACK for multiple segments rather than ACKing each segment individually. To configure an ACK delay, use the `set tcp ack-delay` command in parameter map connection configuration mode. The syntax of this command is as follows:

```
set ack-delay number
```

The `number` argument is an integer from 0 to 400 ms. The default is 200 ms.

For example, to delay sending an ACK from a client to a server for 400 ms, enter:

```
host1/C1(config-parammap-conn)# set ack-delay 400
```

To reset the ACK delay timer to the default value of 200 ms, enter:

```
host1/C1(config-parammap-conn)# no set ack-delay
```

Configuring How the ACE Handles TCP SYN Segments that Contain Data

Occasionally, the ACE may receive a TCP SYN segment that contains data. You can configure the ACE to either discard the segment or flag the segment for data processing. To set the ACE behavior for SYN segments with data, use the `syn-data` command in parameter map connection configuration mode. The syntax of this command is as follows:

```
syn-data { allow | drop }
```

The keywords are as follows:

- **allow**—(Default) Permits the SYN segments that contain data and marks them for data processing
- **drop**—Discards the SYN segments that contain data

For example, to discard SYN segments that contain data, enter:

```
host1/C1(config-parammap-conn)# syn-data drop
```
To reset the ACE behavior to the default of allowing SYN segments that contain data, enter:

```
host1/C1(config-parammap-conn)# no syn-data drop
```

### Configuring How the ACE Handles TCP Options

The ACE permits you to allow or clear the following explicitly supported TCP options specified in a SYN segment:

- Selective Acknowledgement (SACK)
- Time stamp
- Window Scale

You can also specify a range of TCP option numbers for those TCP options not explicitly supported by the ACE. To configure TCP options, use the `tcp-options` command in parameter map connection configuration mode. The syntax of this command is as follows:

```
tcp-options {range number1 number2 {allow | drop}} | {selective-ack | timestamp | window-scale} {allow | clear | drop}
```

The order of precedence for the actions in this command is as follows:

1. Drop
2. Clear
3. Allow

The keywords, options, and variables are as follows:

- **range**—Specifies the TCP options not explicitly supported by the ACE using a range of option numbers. This command enables you to allow or discard segments associated with the TCP options specified in the option range.
  
  - `number1`—Lower limit of the TCP option range. Enter either 6 or 7 or an integer from 9 to 255. See Table 4-2.
  
  - `number2`—Upper limit of the TCP option range. Enter either 6 or 7 or an integer from 9 to 255. See Table 4-2.

- **allow**—Allows any segment with the specified option set.

- **drop**—Used with the range or window-scale option only. Causes the ACE to discard any segment with the specified option set.
• **selective-ack**—Allows the ACE to inform the sender about all segments that it received. The sender needs to retransmit the lost segments only, rather than wait for a cumulative acknowledgement or retransmit segments unnecessarily. Selective ACK (SACK) can reduce the number of retransmitted segments and increase the throughput under some circumstances.

• **timestamp**—Measures the round-trip time (RTT) of a TCP segment between two nodes on a network. Time stamps are always sent and echoed in both directions.

• **window-scale**—Allows the ACE to use a window scale factor that essentially increases the size of the TCP send and receive buffers. The sender specifies a window scale factor in a SYN segment that determines the send and receive window size for the duration of the connection.

• **clear**—Default for the explicitly supported options. Clears the specified option from any segment that has it set and allows the segment.

Table 4-2 lists the TCP options available for the tcp-options range command.

<table>
<thead>
<tr>
<th>Kind</th>
<th>Length</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>6</td>
<td>Echo (obsoleted by option 8)</td>
<td>[RFC1072]</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>Echo reply (obsoleted by option 8)</td>
<td>[RFC1072]</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>Partial order connection permitted</td>
<td>[RFC1693]</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>Partial order service profile</td>
<td>[RFC1693]</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>CC</td>
<td>[RFC1644]</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>CC.NEW</td>
<td>[RFC1644]</td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>CC.ECHO</td>
<td>[RFC1644]</td>
</tr>
<tr>
<td>14</td>
<td>3</td>
<td>TCP alternate checksum request</td>
<td>[RFC1146]</td>
</tr>
<tr>
<td>15</td>
<td>N</td>
<td>TCP alternate checksum data</td>
<td>[RFC1146]</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>Skeeter</td>
<td>[Knowles]</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>Bubba</td>
<td>[Knowles]</td>
</tr>
</tbody>
</table>
Table 4-2  TCP Options for the tcp options range Command (continued)

<table>
<thead>
<tr>
<th>Kind</th>
<th>Length</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>3</td>
<td>Trailer checksum option</td>
<td>[Subbu &amp; Monroe]</td>
</tr>
<tr>
<td>19</td>
<td>18</td>
<td>MD5 signature option</td>
<td>[RFC2385]</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>SCPS capabilities</td>
<td>[Scott]</td>
</tr>
<tr>
<td>21</td>
<td></td>
<td>Selective negative acknowledgements (SNACK)</td>
<td>[Scott]</td>
</tr>
<tr>
<td>22</td>
<td></td>
<td>Record boundaries</td>
<td>[Scott]</td>
</tr>
<tr>
<td>23</td>
<td></td>
<td>Corruption experienced</td>
<td>[Scott]</td>
</tr>
<tr>
<td>24</td>
<td></td>
<td>SNAP</td>
<td>[Sukonnik]</td>
</tr>
<tr>
<td>25</td>
<td></td>
<td>Unassigned (released 12/18/00)</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
<td>TCP compression filter</td>
<td>[Bellovin]</td>
</tr>
</tbody>
</table>

Table 4-3 lists the TCP options explicitly supported by this command.

Table 4-3  TCP Options Explicitly Supported by the ACE

<table>
<thead>
<tr>
<th>Kind</th>
<th>Length</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>End of option list</td>
<td>[RFC793]</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>No operation</td>
<td>[RFC793]</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>WSOPT—Window Scale</td>
<td>[RFC1323]</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>Selective acknowledgement (SACK) permitted</td>
<td>[RFC2018]</td>
</tr>
<tr>
<td>5</td>
<td>N</td>
<td>SACK</td>
<td>[RFC2018]</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td>Time stamp option (TSOPT)</td>
<td>[RFC1323]</td>
</tr>
</tbody>
</table>

You can specify this command multiple times to configure different options and actions. If you specify the same option with different actions, the ACE uses the order of precedence described earlier in this section to decide which action to use.
For example, to allow a segment with the SACK option set, enter:

host1/C1(config-parammap-conn)# tcp-options selective-ack allow

To reset the ACE behavior to the default of clearing the SACK option and allowing the segment, enter:

host1/C1(config-parammap-conn)# no tcp-options selective-ack

You can specify a range of options for each action. If you specify overlapping option ranges with different actions, the ACE uses the order of precedence described earlier in this section to decide which action to perform for the specified options.

For example, enter:

host1/C1(config-parammap-conn)# tcp-options range 6 7 allow
host1/C1(config-parammap-conn)# tcp-options range 19 26 drop

To remove the TCP option ranges from the configuration, enter:

host1/C1(config-parammap-conn)# no tcp-options range 6 7 allow
host1/C1(config-parammap-conn)# no tcp-options range 19 26 drop

**Setting the Urgent Pointer Policy**

If the Urgent control bit (flag) is set in the TCP header, it indicates that the Urgent Pointer is valid. The Urgent Pointer contains an offset that indicates the location of the segment that follows the urgent data in the payload. Urgent data is data that should be processed as soon as possible, even before normal data is processed. The ACE permits you to allow or clear the Urgent flag. If you clear the Urgent flag, you invalidate the Urgent Pointer.

The ACE clears the Urgent flag for any traffic above Layer 4. If you have enabled TCP server connection reuse (see the *Cisco Application Control Engine Module Server Load-Balancing Configuration Guide*, Chapter 2, Configuring Traffic Policies for Server Load Balancing), the ACE does not pass the Urgent flag value to the server.

To set the Urgent Pointer policy, use the `urgent-flag` command in parameter map connection configuration mode. The syntax of this command is as follows:

```
urgent-flag {allow | clear}
```
The keywords are as follows:

- **allow**—(Default) Permits the status of the Urgent flag. If the Urgent flag is set, the offset in the Urgent Pointer that indicates the location of the urgent data is valid. If the Urgent flag is not set, the offset in the Urgent Pointer is invalid.

- **clear**—Sets the Urgent flag to 0, which invalidates the offset in the Urgent Pointer and allows the segment.

For example, to clear the Urgent flag and allow the segment, enter:

```plaintext
host1/C1(config-parammap-conn)# urgent-flag clear
```

To reset the ACE behavior to the default of allowing the Urgent flag, enter:

```plaintext
host1/C1(config-parammap-conn)# no urgent-flag
```

### Setting the Type of Service

The type of service (ToS) for an IP packet determines how the network handles the packet and balances its precedence, throughput, delay, reliability, and cost. This information resides in the IP header. To set the ToS for packets in a particular traffic class, use the `set ip tos` command in parameter map connection configuration mode. The syntax of this command is as follows:

```plaintext
set ip tos number
```

Use the `number` argument to replace a packet’s ToS byte value with the specified value. Enter an integer from 0 to 255. For details about the ToS byte, see RFCs 791, 1122, 1349, and 3168.

For example, to set a packet’s ToS byte value to 20, enter:

```plaintext
host1/C1(config-parammap)# set ip tos 20
```

To reset the ACE behavior to the default of not rewriting the ToS byte value of an incoming packet, enter:

```plaintext
host1/C1(config-parammap)# no set ip tos 20
```
Configuring a Traffic Policy for TCP/IP Normalization and Termination

This section describes how to configure a traffic policy for TCP/IP normalization and termination and contains the following topics:

- Configuring a Layer 4 Class Map
- Configuring a Layer 3 and Layer 4 Policy Map
- Associating a Connection Parameter Map with a Policy Map
- Associating a Layer 3 and Layer 4 Policy Map with a Service Policy

Configuring a Layer 4 Class Map

You can use a Layer 4 class map to classify network traffic for TCP/IP normalization and termination. To match the traffic class, the network traffic must satisfy the match criteria that you specify in the class map.

To configure a class map for TCP/IP normalization and termination, use the `class-map` command in configuration mode. For details about configuring a class map, see the *Cisco Application Control Engine Module Administration Guide*.

The syntax of this command is as follows:

```
class-map [match-all | match-any] name
```

The keywords, arguments, and options are as follows:

- **match-all | match-any**—(Optional) Determines how the ACE evaluates Layer 4 network traffic when multiple match criteria exist in a class map. The class map is considered a match if the `match` commands meet one of the following conditions.
  - **match-all**—(Default) To match the traffic class, network traffic must satisfy all the match criteria listed in the class map, typically, `match` commands of different types.
  - **match-any**—To match the traffic class, network traffic must match only one of the match criteria listed in the class map, typically, `match` commands of the same type.
• *name*—Identifier of the class map. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters. The class name is used for both the class map and to configure policy for the class in the policy map.

For example, enter:

```
host1/C1(config)# class-map match-any TCP_CLASS
host1/C1(config-cmap)#
```

To remove the class map from the configuration, enter:

```
host1/C1(config)# no class-map match-any TCP_CLASS
```

This section contains the following topics:

- Defining a Class Map Description
- Specifying IP Address Match Criteria
- Defining a TCP or UDP Port Number or Port Range Match Criteria

### Defining a Class Map Description

You can use the `description` command in class-map configuration mode to provide a brief description of the Layer 4 class map. The syntax of this command is as follows:

```
description text
```

The `text` argument is an unquoted text string with a maximum of 256 alphanumeric characters.

The following example specifies a description that the class map is to filter network traffic to the server:

```
host1/C1(config)# class-map TCP_CLASS
host1/C1(config-cmap)# description filter tcp connections
```

To remove the description from the class map, enter:

```
host1/C1(config-cmap)# no description filter tcp connections
```

Continue with the following section to enter match criteria as required using the `match` command in class-map configuration mode.
Specifying IP Address Match Criteria

You can specify a source address, destination address, or VIP address as the Layer 3 network traffic match criteria by using the `match` command in class-map configuration mode. The syntax of this command is as follows:

```
[line_number] match {source-address | destination-address | virtual-address} ip_address netmask
```

The keywords, arguments, and options are as follows:

- **line_number**—(Optional) Argument that assists you in editing or deleting individual `match` commands. For example, you can enter `no line_number` to delete long `match` commands instead of entering the entire line.
- **source-address**—Specifies the source IP address as the match criteria.
- **destination-address**—Specifies the destination IP address as the match criteria.
- **virtual-address**—Specifies the virtual IP (VIP) address as the match criteria.
- **ip_address**—IP address of the source, destination, or VIP. Enter an IP address in dotted-decimal notation (for example, 192.168.12.15). You can also specify 0.0.0.0 as a wildcard that will match any IP address.
- **netmask**—(Optional) Subnet mask for the IP address. Enter a subnet mask in dotted-decimal notation (for example, 255.255.255.0). The default subnet mask is 255.255.255.255. You can also specify 0.0.0.0 as a wildcard that will match any netmask.

There can be multiple `match address` commands within a single class map. Also, you can combine other `match` commands in the same class map.

The following example specifies that the network traffic must match destination IP address 172.27.16.7:

```
host1/C1(config)# class-map match-any IP_CLASS
host1/C1(config-cmap)# match destination-address 172.27.16.7
```

To remove the destination IP address match criteria from the class map, enter:

```
host1/C1(config-cmap)# no match destination-address 172.27.16.7
```
Defining a TCP or UDP Port Number or Port Range Match Criteria

You can specify a TCP or UDP port number or port range as the Layer 4 network traffic match criteria by using the `match port` command in class-map configuration mode. The syntax of this command is as follows:

```
[line_number] match port {tcp | udp [eq port1 | range port2 port3]}
```

The keywords, arguments, and options are as follows:

- **line_number**—(Optional) Argument that assists you in editing or deleting individual match commands. For example, you can enter `no line_number` to delete long match commands instead of entering the entire line.
- **tcp**—Specifies TCP.
- **udp**—Specifies UDP.
- **eq port1**—Specifies that the TCP or UDP port number of the network traffic must match the specified value. Enter an integer from 0 to 65535. A value of 0 instructs the ACE to match any port. Alternatively, you can enter a protocol keyword that corresponds to a TCP or UDP port number. See Table 4-4 for a list of supported well-known TCP port names and numbers. See Table 4-5 for a list of supported well-known UDP port names and numbers.

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Port Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>domain</td>
<td>53</td>
<td>Domain Name System</td>
</tr>
<tr>
<td>ftp</td>
<td>21</td>
<td>File Transfer Protocol</td>
</tr>
<tr>
<td>ftp-data</td>
<td>20</td>
<td>File Transfer Protocol Data</td>
</tr>
<tr>
<td>h323</td>
<td>1720</td>
<td>H.323 Call Signaling Protocol</td>
</tr>
<tr>
<td>http</td>
<td>80</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>https</td>
<td>443</td>
<td>HTTP over TLS/SSL</td>
</tr>
<tr>
<td>irc</td>
<td>194</td>
<td>Internet Relay Chat</td>
</tr>
<tr>
<td>matip-a</td>
<td>350</td>
<td>Mapping of Airline Traffic over Internet Protocol (MATIP) Type A</td>
</tr>
<tr>
<td>nntp</td>
<td>119</td>
<td>Network News Transport Protocol</td>
</tr>
<tr>
<td>pop2</td>
<td>109</td>
<td>Post Office Protocol v2</td>
</tr>
</tbody>
</table>
Table 4-4  Well-Known TCP Port Numbers and Keywords (continued)

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Port Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pop3</td>
<td>110</td>
<td>Post Office Protocol v3</td>
</tr>
<tr>
<td>rtsp</td>
<td>554</td>
<td>Real Time Streaming Protocol</td>
</tr>
<tr>
<td>sip</td>
<td>5060</td>
<td>Session Initiation Protocol</td>
</tr>
<tr>
<td>skinny</td>
<td>2000</td>
<td>Skinny Client Control Protocol</td>
</tr>
<tr>
<td>smtp</td>
<td>25</td>
<td>Simple Mail Transfer Protocol</td>
</tr>
<tr>
<td>telnet</td>
<td>23</td>
<td>Telnet</td>
</tr>
<tr>
<td>www</td>
<td>80</td>
<td>World Wide Web</td>
</tr>
</tbody>
</table>

Table 4-5  Well-Known UDP Port Numbers and Keywords

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Port Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>domain</td>
<td>53</td>
<td>Domain Name System</td>
</tr>
<tr>
<td>ras</td>
<td>1719</td>
<td>H.323 RAS protocol</td>
</tr>
<tr>
<td>sip</td>
<td>5060</td>
<td>Session Initiation Protocol (SIP)</td>
</tr>
<tr>
<td>wsp</td>
<td>9200</td>
<td>Connectionless Wireless Session Protocol (WSP)</td>
</tr>
<tr>
<td>wsp-wtls</td>
<td>9202</td>
<td>Secure Connectionless WSP</td>
</tr>
<tr>
<td>wsp-wtp</td>
<td>9201</td>
<td>Connection-based WSP</td>
</tr>
<tr>
<td>wsp-wtp-wtls</td>
<td>9203</td>
<td>Secure Connection-based WSP</td>
</tr>
</tbody>
</table>

- **range port2 port3**—Specifies a port range to use for the TCP or UDP port. Enter an integer from 0 to 65535. A value of 0 instructs the ACE to match any port.

You can have multiple `match port` commands within a single class map. Also, you can combine other `match` commands with the `match port` command in the same class map.

The following example specifies that the network traffic must match on TCP port number 23 (Telnet client):

```
host1/C1(config)# class-map TCP_CLASS
host1/C1(config-cmap)# match port tcp eq 23
```
To remove the TCP port number match criterion from the class map, enter:

```
host1/C1(config-cmap)# no match port tcp eq 23
```

## Configuring a Layer 3 and Layer 4 Policy Map

You can configure a Layer 4 traffic policy for TCP normalization, termination, and reuse by using the `policy-map` command in configuration mode. The ACE attempts to match multiple classes within a Layer 4 policy map, but can match only one class within each feature. If a classification matches more than one class map, then the ACE executes all the corresponding actions. However, for a specific feature, the ACE executes only the first matching classification action. For more information about policy maps, see the *Cisco Application Control Engine Module Administration Guide*.

The syntax of this command is as follows:

```
policy-map multi-match name
```

The `name` argument is the identifier of the policy map. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, enter:

```
host1/C1(config)# policy-map multi-match TCP_POLICY
host1/C1(config-pmap)#
```

To remove a policy map from the configuration, enter:

```
host1/C1(config)# no policy-map multi-match TCP_POLICY
```

## Associating a Layer 3 and Layer 4 Class Map with a Policy Map

You can associate a Layer 4 class map with a Layer 4 policy map by using the `class` command in policy-map configuration mode. The syntax of this command is as follows:

```
class {name1 | class-default} [insert-before name2]
```
The keywords, arguments, and options are as follows:

- **name1**—Name of a previously defined traffic class configured with the `class-map` command. Enter an unquoted text string with no spaces and a maximum of 32 alphanumeric characters.

- **class-default**—Specifies a reserved, well-known class map created by the ACE. You cannot delete or modify this class. All traffic that fails to meet the other matching criteria in the named class map belongs to the default traffic class. If none of the specified classifications match the traffic, then the ACE performs the action specified under the `class class-default` command. The `class-default` class map has an implicit `match any` statement in it that enables it to match all traffic.

  **Note** When you configure a connection parameter map with a class-default class map in a Layer 3 policy map, it is applied to all other Layer 3 class maps in the same Layer 3 policy map.

- **insert-before name2**—(Optional) Places the current class map ahead of an existing class map specified by the `name2` argument in the policy-map configuration. The ACE does not save the sequence reordering as part of the configuration.

The following example shows how to use the `insert-before` command to define the sequential order of two class maps in the policy map:

```
(config-pmap)# 10 class TCP_CLASS insert-before IP_CLASS
(config-pmap-c)#
```

To remove a class map from a Layer 4 policy map, enter:

```
(config-pmap)# no 10 class TCP_CLASS
```

**Associating a Connection Parameter Map with a Policy Map**

You can associate a connection parameter map with a policy map by using the `connection advanced-options` command in policy-map class configuration mode. For details about configuring a connection parameter map, see the “Configuring a Connection Parameter Map for TCP/IP Normalization and Termination” section. The syntax of this command is as follows:

```
connection advanced-options name
```
The *name* argument is a unique identifier of an existing parameter map, specified as an unquoted text string with a maximum of 64 alphanumeric characters.

For example, enter:

```
host1/C1(config-pmap-c)# connection advanced-options TCP_PARAM_MAP
```

To dissociate the TCP parameter map from a policy map, enter:

```
host1/C1(config-pmap-c)# no connection advanced-options TCP_PARAM_MAP
```

### Associating a Layer 3 and Layer 4 Policy Map with a Service Policy

After you configure a Layer 4 policy map with a class map, a connection parameter map, and connection parameters, you must associate the policy map with a service policy to activate it. To associate a policy map with a service policy, use the `service-policy` command in configuration mode. The syntax of this command is as follows:

```
service-policy input name
```

The keywords and arguments are as follows:

- **input**—Specifies that the service policy is to be applied to the incoming traffic
- **name**—Identifier of the policy map that you want to associate with the service policy

For example, enter:

```
host1/C1(config)# service-policy input TCP_POLICY
```

To dissociate a policy map from a service policy, enter:

```
host1/C1(config)# no service-policy input TCP_POLICY
```
Configuring Interface Normalization Parameters

This section describes how to configure TCP/IP normalization parameters in interface configuration mode. It contains the following topics:

- Disabling TCP Normalization on an Interface
- Disabling the ICMP Security Checks on an Interface
- Configuring SYN-Cookie Denial-of-Service Protection
- Configuring How the ACE Handles the Don’t Fragment Bit
- Configuring How the ACE Handles IP Options
- Setting the IP Packet TTL
- Configuring Unicast Reverse-Path Forwarding

Disabling TCP Normalization on an Interface

By default, TCP normalization is enabled. To disable TCP normalization on an interface, use the `no normalization` command in interface configuration mode. Disabling TCP normalization affects only Layer 4 traffic. TCP normalization is always enabled for Layer 7 traffic.

Use this command when you encounter the following two types of asymmetric flows, which would otherwise be blocked by the normalization checks that the ACE performs:

- ACE only sees the client-to-server traffic. For example, for a TCP connection, the ACE sees the SYN from the client, but not the SYN-ACK from the server. In this case, apply the `no normalization` command to the client-side VLAN.

- ACE only sees the server-to-client traffic. For example, for a TCP connection, the ACE receives a SYN-ACK from the server without having received the SYN from the client. In this case, apply the `no normalization` command to the server-side VLAN.

Note

With TCP normalization disabled, the ACE still sets up flows for the asymmetric traffic described above and makes entries in the connection table.
Disabling TCP normalization may expose your ACE and your data center to potential security risks. TCP normalization helps protect the ACE and the data center from attackers by enforcing strict security policies that are designed to examine traffic for malformed or malicious segments.

The syntax of this command is as follows:

```
no normalization
```

For example, to disable TCP normalization on interface VLAN 100, enter:

```
host1/C1(config)# interface vlan 100
host1/C1(config-if)# no normalization
```

To reenable TCP normalization, enter:

```
host1/C1(config-if)# normalization
```

Disabling the ICMP Security Checks on an Interface

The ACE provides several ICMP security checks by matching ICMP reply packets with request packets and using mismatched packets to detect attacks. Also, the ACE forwards ICMP error packets only if a connection record exists pertaining to the flow for which the error packet was received. By default, the ACE ICMP security checks are enabled.

To disable the ICMP security checks, use the `no icmp-guard` command in interface mode. Use this command as part of an overall strategy to operate the ACE as a pure server load balancer. For details, see Chapter 1, Overview, in the *Cisco Application Control Engine Module Server Load-Balancing Configuration Guide*.

The syntax of this command is as follows:

```
no icmp-guard
```

Disabling the ACE ICMP security checks may expose your ACE and your data center to potential security risks. After you enter the `no icmp-guard` command, the ACE no longer performs NAT translations on the ICMP header and payload in error packets, which potentially can reveal real host IP addresses to attackers.
For example, to disable ICMP security checks on interface VLAN 100, enter:

```
host1/C1(config)# interface vlan 100
host1/C1(config-if)# no icmp-guard
```

To reenable ICMP security checks, enter:

```
host1/C1(config-if)# icmp-guard
```

## Configuring SYN-Cookie Denial-of-Service Protection

This section describes the SYN cookie feature that the ACE uses to protect itself and devices in the data center from Denial of Service (DoS) attacks. It covers the following topics:

- **Overview of SYN Cookie DoS Protection**
- **Configuration and Operational Considerations**
- **Configuring SYN Cookie DoS Protection on an Interface**

### Overview of SYN Cookie DoS Protection

Occasionally, a TCP three-way handshake (SYN, SYN-ACK, ACK) may not complete for some reason. These incomplete or half-open connections are known as *embryonic* connections. Such occurrences are normal if the frequency is low. However, a large number of embryonic connections could indicate a DoS attack (SYN flood attack) by a hacker.

A SYN flood attack is characterized by a large number of SYNs sent to a server or other host from one or more hosts with source IP addresses that are invalid and unreachable. Such an attack causes half-open connections on the target host that must time out before the host can service other connection requests. When multiple hosts in different networks are used to attack a server or other host, the attack is known as a Distributed Denial of Service (DDoS). The goal of the attacker is to overwhelm the target host, consume its resources, and cause it to deny service to legitimate connection requests.

The ACE allows you to protect it and the servers and other hosts in the data center from SYN flood attacks by configuring SYN-cookie-based DoS protection for TCP connections. You configure an embryonic connection threshold, beyond which the ACE applies SYN cookie protection.
When the configured embryonic connection threshold is reached, the ACE intercepts the next SYN packet from a client. The ACE responds to the SYN with a SYN-ACK using a sequence number that is the actual SYN cookie value. The SYN cookie consists of the following:

- A 32-bit timer that increases every 64 seconds.
- An encoding of the client MSS, which the ACE forwards to the server.
- An ACE-selected secret that is calculated from the 4-tuple (source IP address, source port, destination IP address, and destination port) and the timer value.

Normally, if the SYN queue fills up, the ACE drops additional connection requests. If the SYN queue fills up on the ACE with SYN cookies enabled, the ACE continues to service a client request normally by sending a SYN-ACK to the requesting client as if the SYN queue was actually larger. The ACE uses the calculated SYN cookie value as the sequence number \( n \) and discards the SYN queue entry.

When it receives an ACK (sequence number = \( n + 1 \)) from the client, the ACE verifies the validity of the secret and the SYN cookie value for a recent value of the SYN cookie timer. If the secret or the sequence number is not valid, the ACE drops the packet. If the secret and the sequence number are valid, the ACE rebuilds the SYN queue entry based on the encoded MSS and the ACK from the client. At this point, the connection process proceeds normally; the ACE sends the newly built SYN to the server and establishes the back-end TCP connection.

When you configure SYN cookie protection, the ACE calculates the internal embryonic connection threshold value for each network processor (NP) as \( \frac{\text{configured\_threshold}}{2} \) (fractions are disregarded). Therefore, you may occasionally observe that SYN cookie protection is applied before the embryonic connection count reaches the configured threshold value. For example, suppose that you configure a threshold value of 4. Because the threshold value is divided by two internally for each NP, the internally calculated threshold is 2. After two incomplete connection attempts (SYNs) are sent to the same NP, the ACE activates SYN cookie protection and intercepts the third SYN going to that NP.

**Configuration and Operational Considerations**

When you use the SYN cookie feature, be aware of the following considerations:

- If the server drops the SYN that is sent by the ACE, the ACE resets the connection using the embryonic timeout. It does not retry the SYN packet.
A SYN cookie supports only the MSS TCP option. The ACE ignores all other TCP options, even if there are problems with those other options.

The ACE returns an MSS of 536 to the client, which is the RFC-specified default.

If you use a parameter map to specify the minimum and maximum MSS values, the ACE ignores those values.

Disabling normalization and using a SYN cookie concurrently may result in unpredictable behavior.

The ACE does not generate any syslogs for a SYN cookie, even if the number of embryonic connections exceeds the configured threshold, which may indicate a SYN-flood attack.

If you are configuring the SYN cookie feature on a bridged VLAN with non-loadbalanced flows, you must configure static routes for non-loadbalanced destinations that do not reside in the same subnet as the bridge-group virtual interface (BVI).

For example, assuming the following configuration:

- BVI IP address is 192.168.1.1
- Gateway1 IP address 192.168.1.2 to reach external network 172.16.1.0
- Gateway2 IP address 192.168.1.3 to reach external network 172.31.1.0

Configure the following static routes:

- ip route 172.16.1.0 255.255.255.0 192.168.1.2
- ip route 172.31.1.0 255.255.255.0 192.168.1.3

**Configuring SYN Cookie DoS Protection on an Interface**

To configure SYN-cookie-based DoS protection, use the `syn-cookie` command in interface configuration mode. The syntax of this command is as follows:

```
syn-cookie number
```

The `number` is the embryonic connection threshold above which the ACE applies SYN-cookie DoS protection. Enter an integer from 2 to 65535.
For example, to configure SYN-cookie DoS protection for servers in a data center connected to VLAN 100, enter:

```bash
host1/C1(config)# interface vlan 100
host1/C1(config-if)# syn-cookie 4096
```

To remove SYN-cookie DoS protection from the interface, enter:

```bash
host1/C1(config-if)# no syn-cookie
```

### Configuring How the ACE Handles the Don’t Fragment Bit

Occasionally, an ACE may receive a packet that has its Don’t Fragment (DF) bit set in the IP header. This flag tells network routers and the ACE not to fragment the packet and to forward it in its entirety. To configure how the ACE handles the DF bit, use the `ip df` command in interface configuration mode. The syntax of this command is as follows:

```
ip df {clear | allow}
```

The keywords are as follows:

- **clear**—Clears the DF bit and permits the packet. If the packet is larger than the next-hop MTU, the ACE fragments the packet.

- **allow**—Permits the packet with the DF bit set. If the packet is larger than the next-hop MTU, the ACE discards the packet and sends an ICMP unreachable message to the source host.

For example, to clear the DF bit and permit the packet, enter:

```bash
host1/C1(config-if)# ip df clear
```

To instruct the ACE to ignore the DF bit, enter:

```bash
host1/C1(config-if)# no ip df
```
Configuring How the ACE Handles IP Options

The ACE can process IP options and perform specific actions when an IP option is set in a packet. To configure how the ACE handles IP options, use the `ip options` command in interface configuration mode. The syntax of this command is as follows:

```
ip options {allow | clear | clear-invalid | drop}
```

The keywords are as follows:

- **allow**—Allows the packet with IP options set
- **clear**—Clears all IP options from the packet and allows the packet
- **clear-invalid**—(Default) Clears all IP options from the packet if the ACE encounters one or more invalid or unsupported IP options and allows the packet
- **drop**—Instructs the ACE to discard the packet regardless of any IP options that are set

For example, enter:

```
host1/C1(config-if)# ip options allow
```

To reset the ACE behavior to the default of clearing all IP options if the module encounters one or more invalid or unsupported IP options, enter:

```
host1/C1(config-if)# no ip options
```

Setting the IP Packet TTL

The packet time to live (TTL) specifies the number of hops that a packet is allowed to reach its destination. Each router along the packet’s path decrements the TTL by one. If the packet’s TTL reaches zero before the packet reaches its destination, the packet is discarded.

To specify the minimum TTL value that the ACE accepts in the IP header of an incoming packet, use the `ip ttl` command in interface configuration mode. The default behavior of the ACE is to not rewrite the TTL value of a packet. The syntax of this command is as follows:

```
ip ttl minimum number
```
The *number* argument is the minimum number of hops that a packet is allowed to reach its destination. Enter an integer from 1 to 255 hops.

**Note**

If the TTL value of the incoming packet is lower than the configured minimum value, the ACE rewrites the TTL with the configured value. Otherwise, the ACE transmits the packet with its TTL unchanged or discards the packet if the TTL equals zero.

For example, to set the TTL to 15, enter:

```
host1/C1(config-if)# ip ttl minimum 15
```

To reset the behavior of the ACE to the default of not overwriting the TTL of an incoming IP packet, enter:

```
host1/C1(config-if)# no ip ttl minimum
```

### Configuring Unicast Reverse-Path Forwarding

Unicast reverse-path forwarding (URPF) helps to mitigate problems caused by the introduction of malformed or forged (spoofed) IP source addresses into a network by allowing the ACE to discard IP packets that lack a verifiable source IP address. This feature enables the ACE to filter both ingress and egress packets to verify addressing and route integrity. It is called RPF because the route lookup is typically based on the destination address, not the source address.

When you enable this feature, the ACE discards packets if there is no route found or if the route does not match the interface on which the packet arrived.

**Note**

If you configure the `mac-sticky` command on the interface, you cannot configure the `ip verify reverse-path` command. For details about the `mac-sticky` command, see the *Cisco Application Control Engine Module Routing and Bridging Configuration Guide*.

To enable this feature, use the `ip verify reverse-path` command in interface configuration mode. The syntax of this command is as follows:

```
ip verify reverse-path
```
For example, to enable reverse-path forwarding, enter:

```
host/C1(config-if)# ip verify reverse-path
```

To disable reverse-path forwarding, enter:

```
host/C1(config-if)# no ip verify reverse-path
```

## Configuring IP Fragment Reassembly Parameters

You can configure several parameters that control how the ACE performs IP fragment reassembly. This section contains the following topics:

- IP Fragment Reassembly Configuration Quick Start
- Configuring the MTU for an Interface
- Configuring the Maximum Number of Fragments in a Packet
- Configuring the Minimum Fragment Size for Reassembly
- Configuring an IP Reassembly Timeout

### IP Fragment Reassembly Configuration Quick Start

Table 4-6 provides a quick overview of the steps required to configure IP fragment reassembly. Each step includes the CLI command or a reference to the procedure required to complete the task. For a complete description of each feature and all the options associated with the CLI commands, see the sections following Table 4-6.
### Table 4-6  **IP Fragment Reassembly Configuration Quick Start**

**Task and Command Example**

1. If you are operating in multiple contexts, observe the CLI prompt to verify that you are operating in the desired context. If necessary, change to the correct context.

   ```
   host1/Admin# ch Gent C1
   host1/C1#
   ```

   The rest of the examples in this table use the C1 context, unless otherwise specified. For details on creating contexts, see the *Cisco Application Control Engine Module Virtualization Configuration Guide*.

2. Enter configuration mode.

   ```
   host1/C1# config
   host1/C1(config)#
   ```

3. Enter interface configuration mode for the interface on which you want to configure fragment reassembly parameters.

   ```
   host1/C1(config)# interface vlan 100
   host1/C1(config-if)#
   ```

4. Configure the maximum number of fragments belonging to the same packet that the ACE accepts for reassembly.

   ```
   host1/C1(config-if)# fragment chain 126
   ```

5. Configure the minimum fragment size that the ACE will accept for reassembly.

   ```
   host1/C1(config-if)# fragment min-mtu 1024
   ```

6. Configure a fragment reassembly timeout to specify the period of time after which the ACE abandons the fragment reassembly process if it does not receive any outstanding fragments for the current fragment chain (fragments that belong to the same packet).

   ```
   host1/C1(config-if)# fragment timeout 15
   ```

7. (Optional) Save your configuration changes to flash memory.

   ```
   host1/C1# copy running-config startup-config
   ```

8. Display the IP fragment reassembly configuration information.

   ```
   host1/C1# show interface vlan 100
   ```
Configuring the MTU for an Interface

The default maximum transmission unit (MTU) is 1500 bytes in a block for Ethernet interfaces. This value is sufficient for most applications, but you can pick a lower number if network conditions require it. Data that is larger than the MTU value is fragmented before being sent to the next hop router.

⚠️ Caution ⚠️

If you configure a Layer 7 policy map and set the maximum transmit unit (MTU) of the ACE server-side VLAN lower than the client maximum segment size (MSS), ensure that the maximum value of the MSS that you set for the ACE using the `set tcp mss max` command is at least 40 bytes (size of the TCP header plus options) less than the MTU of the ACE server-side VLAN. Otherwise, the ACE may discard incoming packets from the server.

To specify the MTU for an interface, use the `mtu` command in interface configuration mode. This command allows you to set the data size that is sent on a connection. The syntax of this command is as follows:

```
mtu bytes
```

The `bytes` argument is the number of bytes in the MTU. Enter a number from 68 to 9216 bytes. The default is 1500 bytes.

To specify the MTU data size of 1000 bytes for an interface, enter:

```
host1/admin(config-if)# mtu 1000
```

To reset the MTU block size to 1500 bytes, use the `no mtu` command. For example, enter:

```
host1/admin(config-if)# no mtu
```

Configuring the Maximum Number of Fragments in a Packet

You can configure the maximum number of fragments belonging to the same packet that the ACE accepts for reassembly by using the `fragment chain` command in interface configuration mode. The syntax of this command is as follows:

```
fragment chain number
```
The number argument, is fragment chain limit as an integer from 1 to 256 fragments. The default is 24 fragments.

For example, enter:

```
host1/C1(config-if)# fragment chain 126
```

To reset the maximum number of fragments in a packet to the default of 24, enter:

```
host1/C1(config-if)# no fragment chain
```

### Configuring the Minimum Fragment Size for Reassembly

You can configure the minimum fragment size that the ACE accepts for reassembly by using the `fragment min-mtu` command in interface configuration mode. The syntax of this command is as follows:

```
fragment min-mtu number
```

The number argument is the minimum fragment size as an integer from 28 to 9216 bytes. The default is 576 bytes.

For example, enter:

```
host1/C1(config-if)# fragment min-mtu 1024
```

To reset the minimum fragment size to the default value of 576 bytes, enter:

```
host1/C1(config-if)# no fragment min-mtu
```

### Configuring an IP Reassembly Timeout

The IP reassembly timeout specifies the period of time after which the ACE abandons the fragment reassembly process if it does not receive any outstanding fragments for the current fragment chain (fragments that belong to the same packet). To configure a reassembly timeout, use the `fragment timeout` command in interface configuration mode. The syntax of this command is as follows:

```
fragment timeout seconds
```

The seconds argument is an integer from 1 to 30 seconds. The default is 5 seconds.
For example, enter:

```
host1/C1(config-if)# fragment timeout 15
```

To reset the fragment timeout to the default value of 5 seconds, enter:

```
host1/C1(config-if)# no fragment timeout
```

### Configuring the Switch Mode Feature

Use the switch mode feature to change the way that the ACE handles TCP connections that are not destined to a particular VIP and those connections that do not have any policies associated with their traffic. When you enable this feature, the ACE still creates connection objects for those TCP sessions that are not destined to the VIP. The ACE processes these connections as stateless connections, which means that they do not undergo any TCP normalization checks (for example, TCP window, TCP state, TCP sequence number, and other normalization checks).

The ACE also creates stateless connections for non-SYN TCP packets if they satisfy all other configured requirements, for example, ACLs and other policies. This process ensures that a long-lived persistent connection passes through the ACE successfully (even if it times out) by being reestablished by any incoming packet related to the connection.

By default, these stateless connections time out after 2 hours and 15 minutes unless you configure the timeout otherwise. When a stateless connection times out, the ACE does not send a TCP RST packet but instead closes the connection silently. Even though these connections are stateless, the TCP RST and FIN-ACK flags are honored and the connections are closed when the ACE sees these flags in the received packets.

To change the default timeout for these stateless connections, use the `set timeout inactivity` command in parameter map connection configuration mode. For details about this command, see the *Cisco Application Control Engine Module Security Configuration Guide*.

The SYN cookie feature still operates normally for these stateless connections that are not destined to any VIP.

The default timeout value of 2 hours and 15 minutes is also applicable to the UDP connections that are not destined to any VIP.
Chapter 4  Configuring TCP/IP Normalization and IP Reassembly Parameters

Example of a TCP/IP Normalization and IP Reassembly Configuration

Note

If you configure switch mode and you configure any connection parameter-map commands (for example, set tcp buffer-share, rate-limit, exceed-mss, nagle, random-sequence-number, reserved-bits, set tcp wan-optimization, timeout inactivity, slowstart, and so on) either locally on a specific interface or globally on all interfaces, then switch mode will override these commands for certain types of traffic. This behavior applies only to non-VIP, non-inspection, non-NATed, and non-management traffic. The ACE continues to apply local, global, and VIP-specific connection parameter maps to load-balanced (VIP), inspected, NATed, and management traffic. For details about switch mode, see the “Configuring the Switch Mode Feature” section.

To enable the switch mode feature, use the switch-mode command in configuration mode. The syntax of this command is as follows:

```
switch-mode
```

For example, to enable the switch mode feature, enter the following command:

```
host1/Admin(config)# switch-mode
```

To disable the switch mode feature, enter the following command:

```
host1/Admin(config)# no switch-mode
```

Example of a TCP/IP Normalization and IP Reassembly Configuration

The following example illustrates a running-configuration in which the ACE uses TCP normalization to perform checks for Layer 4 packets that have invalid or suspect conditions and to take the appropriate actions based on the configured TCP connection parameter map settings. The ACE uses TCP normalization to block certain types of network attacks. This configuration also includes IP fragment reassembly parameters. The TCP/IP normalization and IP fragment reassembly configuration appears in bold in the example.
In the following configuration, the ACE does the following:

- Includes a connection parameter map that groups together TCP/IP normalization and termination parameters, such as a connection inactivity timer, ToS for an IP packet, and discarding the SYN segments that contain data. The connection parameter map is associated as an action in the TCP/IP policy map.

- Configures additional IP normalization parameters for a specific VLAN interface, such as clearing all IP options from the packet, define the number of hops that a packet is allowed to reach its destination, and permit the packet with the DF bit set.

- Configures IP fragment reassembly parameters for a specific VLAN interface, such as the minimum fragment size that the ACE accepts for reassembly, the maximum number of fragments that belong to the same packet that the ACE accepts for reassembly, and the minimum fragment size that the ACE accepts for reassembly.

```plaintext
access-list ACL1 line 10 extended permit ip any any

parameter-map type connection TCPIP_PARAM_MAP
    set timeout inactivity 30
    set ip tos 20
    tcp-options timestamp allow
    syn-data drop
    urgent-flag clear

class-map match-all L4_TCP_CLASS
    description Filter TCP Connections
    2 match destination-address 172.27.16.7
    3 match port tcp eq 21

policy-map multi-match L4_TCPIP_POLICY
    class L4_TCP_CLASS
        connection advanced-options TCP_PARAM_MAP

interface vlan 50
    access-group input ACL1
    ip address 192.168.1.100 255.255.255.0
    service-policy input L4_TCPIP_POLICY
    ip ttl1 minimum 15
    ip options clear
    ip df allow
    fragment size 400
    fragment chain 126
    fragment min-mtu 1024
    fragment timeout 15
    no shutdown
```
Displaying Configurations and Statistics for TCP/IP and UDP Connections, IP Reassembly, and SYN Cookie

This section describes the `show` commands that you can use to display configurations and statistics for the following:

- TCP connections parameters
- IP connections parameters
- UDP connection parameters
- IP fragment reassembly

This section contains the following topics:

- Displaying TCP/IP and UDP Connection Configurations
- Displaying a Connection Parameter Map
- Displaying TCP/IP and UDP Connection Statistics
- Displaying Global Context Connection Statistics
- Displaying IP Statistics
- Displaying TCP Statistics
- Displaying UDP Statistics
- Displaying Service Policy Statistics
- Displaying SYN Cookie Statistics

Displaying TCP/IP and UDP Connection Configurations

You can display TCP, IP, and UDP connection configurations by using the following `show` commands in Exec mode:

- **show running-config class-map**—Displays all traffic classifications configured in the current context, including match statements for IP addresses and TCP or UDP ports
- **show running-config policy-map**—Displays all policy maps configured in the current context, including the associated class maps
• **show running-config interface**—Displays all interface VLAN configurations in the current context

For example, to display all policy maps in the current context, enter:

```
host1/C1# show running-config policy-map
```

### Displaying a Connection Parameter Map

You can display a connection parameter map configuration by using the **show parameter-map** command in Exec mode. The syntax of this command is as follows:

```
show parameter-map name
```

The *name* argument is the name of an existing connection parameter map. Enter the name as an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to display a connection parameter map configuration, enter:

```
host1/C1# show parameter-map CONN_PMAP
```

*Table 4-7* describes the fields in the **show parameter-map** command output.
### Table 4-7  Field Descriptions for the show parameter-map Command Output

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter map</td>
<td>Name of the connection parameter map.</td>
</tr>
<tr>
<td>Type</td>
<td>Connection.</td>
</tr>
<tr>
<td>Nagle</td>
<td>Status of the <code>nagle</code> command: enabled or disabled.</td>
</tr>
<tr>
<td>Slow start</td>
<td>Status of the <code>slow start</code> command: enabled or disabled.</td>
</tr>
</tbody>
</table>
| Inactivity timeout (seconds) | Configured number of seconds after which an inactive connection times out. Possible values are from 0 to 3217203. If the `set timeout inactivity` command is not configured, the default values in seconds appear, as follows:  
  - ICMP—2  
  - TCP—3600  
  - UDP—10 |
| Embryonic timeout (seconds)  | Configured number of seconds after which an incomplete TCP handshake times out. Possible values are from 0 to 4294967295. |
| Ack-delay                    | Configured number of seconds that the ACE delays sending an ACK from a client to a server. |
| WAN optimization RTT (milliseconds) | Configured number of milliseconds that determines how the ACE applies TCP optimizations to packets on a connection that is associated with a Layer 7 policy. |
| Half-closed timeout (seconds) | Number of seconds after which a half-closed connection times out. Possible values are from 0 to 4294967295. |
| TOS rewrite                  | State of the `set ip tos` command: enabled or disabled.                     |
| SYN retry count              | State of the `set tcp syn-retry` command: enabled or disabled.              |
| TCP MSS min                  | Minimum value of the TCP maximum segment size that the ACE accepts. Possible values are from 0 to 65535. |
| TCP MSS max                  | Maximum value of the TCP maximum segment size that the ACE accepts. Possible values are from 0 to 65535. |
### Table 4-7  
**Field Descriptions for the show parameter-map Command Output (continued)**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tcp-options drop range</td>
<td>Range of numbers representing the TCP options that the ACE drops.</td>
</tr>
<tr>
<td>Tcp-options allow range</td>
<td>Range of numbers representing the TCP options that the ACE allows. Possible values are 6 or 7 and from 9 to 255.</td>
</tr>
<tr>
<td>Tcp-options clear range</td>
<td>Range of numbers representing the TCP options that the ACE clears. Possible values are 6 or 7 and from 9 to 255.</td>
</tr>
<tr>
<td>Selective-ack</td>
<td>Configured action that the ACE performs for the selective acknowledgement TCP option. Possible actions are allow or clear.</td>
</tr>
<tr>
<td>Timestamp</td>
<td>Configured action that the ACE performs for the time-stamp TCP option. Possible actions are allow or clear.</td>
</tr>
<tr>
<td>Window-scale</td>
<td>Configured action that the ACE performs for the window scale TCP option. Possible actions are allow, clear, or drop.</td>
</tr>
<tr>
<td>Window-scale factor</td>
<td>Value of the <code>set tcp window-scale</code> command. Possible values are from 0 to 14.</td>
</tr>
<tr>
<td>Reserved-bits</td>
<td>Configured action for the <code>reserved-bits</code> command. Possible actions are allow, clear, or drop.</td>
</tr>
<tr>
<td>Random-seq-num</td>
<td>Configured state of the <code>random-sequence-number</code> command. Possible states are enabled or disabled.</td>
</tr>
<tr>
<td>SYN data</td>
<td>Configured action for the <code>syn-data</code> command. Possible actions are allow or drop.</td>
</tr>
<tr>
<td>Exceed-mss</td>
<td>Configured action for the <code>exceed-mss</code> command. Possible actions are allow or drop.</td>
</tr>
<tr>
<td>urgent-flag</td>
<td>Configured action for the <code>urgent-flag</code> command. Possible values are allow or clear.</td>
</tr>
<tr>
<td>conn-rate-limit</td>
<td>Configured maximum number of connections per second that the ACE allows.</td>
</tr>
<tr>
<td>bandwidth-rate-limit</td>
<td>Configured maximum number of bytes per second that the ACE allows.</td>
</tr>
</tbody>
</table>
Displaying TCP/IP and UDP Connection Statistics

This section describes the `show` commands that you can use to display TCP/IP and UDP connection statistics. To display connection statistics, use the `show conn` command in Exec mode. The syntax of this command is as follows:

```
show conn {address ip_address1 [ip_address2] netmask mask [detail]} | count | detail | {port number1 [number2] [detail]} | {protocol {tcp | udp} [detail]}
```

The keywords, arguments, and options are as follows:

- **address ip_address1 [ip_address2]**—Displays connection statistics for a single source or destination IP address or, optionally, for a range of source or destination IP addresses. To specify a range of IP addresses, enter an IP address for the lower limit of the range and a second IP address for the upper limit of the range. Enter one or two IP addresses in dotted-decimal notation (for example, 192.168.12.15).

- **netmask mask**—Displays the network mask for the IP address or range of IP addresses that you specify. Enter a network mask in dotted-decimal notation (for example, 255.255.255.0).

- **count**—Displays the total current connections to the ACE.

- **detail**—Displays detailed connection information.

- **port number1 [number2]**—Displays connection statistics for a single source or destination port or, optionally, for a range of source or destination ports.

- **protocol {tcp | udp}**—Displays connection statistics for TCP or UDP.

For example, to display connection statistics for a range of IP addresses, enter:

```
host1/C1# show conn address 192.168.12.15 192.168.12.35 netmask 255.255.255.0
```
Table 4-8 describes the fields in the `show conn detail` command output.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Current</td>
<td>Total number of current connections to the ACE.</td>
</tr>
<tr>
<td>Conn-ID</td>
<td>Identifier of the inbound or outbound connection.</td>
</tr>
<tr>
<td>NP</td>
<td>Number of the network processor (NP) responsible for the connection.</td>
</tr>
<tr>
<td>Dir</td>
<td>Direction of the connection: in(bound) or out(bound).</td>
</tr>
<tr>
<td>Prot</td>
<td>Protocol used for the connection: TCP or UDP.</td>
</tr>
<tr>
<td>VLAN</td>
<td>Identifier of the interface used for the connection.</td>
</tr>
<tr>
<td>Source</td>
<td>Source IP address and port.</td>
</tr>
<tr>
<td>Destination</td>
<td>Destination IP address and port.</td>
</tr>
<tr>
<td>State</td>
<td>For TCP connections, the current state of the connection (for example, ESTAB).</td>
</tr>
<tr>
<td>Idle Time</td>
<td>Length of time that this connection has been idle.</td>
</tr>
<tr>
<td>Byte Count</td>
<td>Number of bytes that have traversed the connection.</td>
</tr>
<tr>
<td>Elapsed Time</td>
<td>Length of time that has elapsed since the connection was established.</td>
</tr>
<tr>
<td>Packet Count</td>
<td>Number of packets that have traversed the connection.</td>
</tr>
<tr>
<td>Conn in Reuse Pool</td>
<td>Indication of whether the ACE has placed the connection in the pool for possible reuse. Valid values are TRUE or FALSE.</td>
</tr>
</tbody>
</table>

Note: The total current connections is the number of connection objects. There are two connection objects for each flow and complete connection.
Displaying Global Context Connection Statistics

You can display global connection statistics for the current context by using the `show stats connection` command in Exec mode. The syntax of this command is as follows:

```
show stats connection
```

For example, to display global connection statistics for the Admin context, enter the following command:

```
host1/Admin# show stats connection
```

Table 4-9 describes the fields in the `show stats connection` command output.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Connections Created</td>
<td>Total number of connections that were created in the current context. This number is the sum of the Total Connections Current, Total Connections Destroyed, Total Connections Timed-out, and Total Connections Failed.</td>
</tr>
<tr>
<td>Total Connections Current</td>
<td>Total number of existing connections to the current context.</td>
</tr>
<tr>
<td>Total Connections Destroyed</td>
<td>Total number of connections that were torn down in the current context.</td>
</tr>
<tr>
<td>Total Connections Timed-out</td>
<td>Total number of connections that exceeded the configured timeout in the current context.</td>
</tr>
<tr>
<td>Total Connections Failed</td>
<td>Total number of connection attempts that failed to complete.</td>
</tr>
</tbody>
</table>
Displaying IP Statistics

This section describes the show commands that you can use to display IP statistics, including fragmentation, ICMP, TCP, and UDP, and ARP statistics. It contains the following topics:

- Displaying IP Traffic Information
- Displaying IP Fragmentation and Reassembly Statistics

Displaying IP Traffic Information

You can display IP traffic information by using the show ip traffic command in Exec mode. Aside from fragmentation, reassembly and ARP statistics, this command displays statistics for traffic destined to the ACE, rather than through the ACE. The syntax of this command is as follows:

```
show ip traffic
```

For example, enter:

```
host1/C1# show ip traffic
```

Table 4-10 describes the fields in the show ip traffic command output.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP Statistics</td>
<td></td>
</tr>
<tr>
<td>Rcvd</td>
<td>Total number of packets received by the ACE, number of bytes received by the ACE, number of input errors, number of packets received by the ACE with no route, and the number of packets received by the ACE that had an unknown protocol.</td>
</tr>
<tr>
<td>Frags</td>
<td>Number of fragments that the ACE reassembled, number of fragments that the ACE could not reassemble, number of packets that the ACE fragmented, and the number of packets that the ACE could not fragment.</td>
</tr>
<tr>
<td>Bcast</td>
<td>Number of broadcast packets received and sent.</td>
</tr>
<tr>
<td>Mcast</td>
<td>Number of multicast packets received and sent.</td>
</tr>
</tbody>
</table>
### Table 4-10  Field Descriptions for the show ip traffic Command Output

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sent</td>
<td>Total packets sent, number of bytes sent, and the number of packets sent with no route.</td>
</tr>
<tr>
<td>Drop</td>
<td>Number of packets discarded because they had no route, and number of packets discarded.</td>
</tr>
</tbody>
</table>

**ICMP Statistics**

| Rcvd  | Reports statistics for the following ICMP messages received by the ACE:  
|       | • Redirects  
|       | • ICMP Unreachable  
|       | • ICMP Echo  
|       | • ICMP Echo Reply  
|       | • Mask Requests  
|       | • Mask Replies  
|       | • Quench  
|       | • Parameter  
<p>|       | • Timestamp |</p>
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| Sent  | Reports statistics for the following ICMP messages sent by the ACE:  
  - Redirects  
  - ICMP Unreachable  
  - ICMP Echo  
  - ICMP Echo Reply  
  - Mask Requests  
  - Mask Replies  
  - Quench  
  - Timestamp  
  - Parameter  
  - Time Exceeded |
Table 4-10  Field Descriptions for the show ip traffic Command Output

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP Statistics</td>
<td></td>
</tr>
<tr>
<td>Rcvd</td>
<td>Total number of TCP segments and errors received by the ACE.</td>
</tr>
<tr>
<td>Sent</td>
<td>Total number of TCP segments sent by the ACE.</td>
</tr>
<tr>
<td>UDP Statistics</td>
<td></td>
</tr>
<tr>
<td>Rcvd</td>
<td>Total number of UDP segments, UDP errors, and segments with no port number received by the ACE.</td>
</tr>
<tr>
<td>Sent</td>
<td>Total number of UDP segments sent by the ACE.</td>
</tr>
<tr>
<td>ARP Statistics</td>
<td></td>
</tr>
<tr>
<td>Rcvd</td>
<td>Number of ARP packets, errors, requests, and responses received by the ACE.</td>
</tr>
<tr>
<td>Sent</td>
<td>Number of ARP packets, errors, requests, and responses sent by the ACE.</td>
</tr>
</tbody>
</table>

Displaying IP Fragmentation and Reassembly Statistics

You can display IP fragmentation and reassembly statistics for all interfaces in the ACE or the specified interface by using the `show fragment` command in Exec mode. The syntax of this command is as follows:

```
show fragment [vlan vlan_id]
```

For the optional `vlan_id` argument, enter the unique identifier of an existing interface as an integer from 2 to 4094. If you omit the `vlan` keyword and `vlan_id` argument, you can display statistics for all interfaces in the ACE.

For example, to display IP fragmentation and reassembly statistics for all interfaces in the ACE, enter:

```
host1/C1# show fragment
```
Table 4-11 describes the fields in the `show fragment` command output.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface</td>
<td>VLAN ID of the interface.</td>
</tr>
<tr>
<td>Fragment Stats</td>
<td></td>
</tr>
<tr>
<td>Required</td>
<td>Number of packets that were sent to the ACE for</td>
</tr>
<tr>
<td></td>
<td>fragmentation.</td>
</tr>
<tr>
<td>OK</td>
<td>Number of fragments that the ACE successfully</td>
</tr>
<tr>
<td></td>
<td>created.</td>
</tr>
<tr>
<td>Failed</td>
<td>Number of fragmentation attempts that were</td>
</tr>
<tr>
<td></td>
<td>unsuccessful.</td>
</tr>
<tr>
<td>Created</td>
<td>Total number of fragments that the ACE created.</td>
</tr>
<tr>
<td>IP Reassembly Stats</td>
<td></td>
</tr>
<tr>
<td>Required</td>
<td>Number of packets that were sent to the ACE for</td>
</tr>
<tr>
<td></td>
<td>reassembly.</td>
</tr>
<tr>
<td>OK</td>
<td>Number of packets that the ACE successfully</td>
</tr>
<tr>
<td></td>
<td>reassembled.</td>
</tr>
<tr>
<td>Failed</td>
<td>Number of packet reassembly attempts that were</td>
</tr>
<tr>
<td></td>
<td>unsuccessful.</td>
</tr>
</tbody>
</table>

### Displaying TCP Statistics

You can display TCP statistics by using the `show tcp statistics` command in Exec mode. This command display statistics for traffic destined to the ACE, rather than through the ACE. The syntax of this command is as follows:

```
show tcp statistics
```

For example, to display TCP statistics for the current context, enter:

```
host1/C1# show tcp statistics
```
Table 4-12 describes the fields in the `show tcp statistics` command output.

**Table 4-12 Field Descriptions for the show tcp statistics Command Output**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rcvd</td>
<td>Total number of TCP segments and errors received by the ACE.</td>
</tr>
<tr>
<td>Sent</td>
<td>Total number of TCP segments, reset flag segments, active opens, and passive opens sent by the ACE.</td>
</tr>
<tr>
<td>Connections</td>
<td>Number of failed connection attempts, established connections that were reset, and currently established connections.</td>
</tr>
</tbody>
</table>

**Displaying UDP Statistics**

You can display UDP statistics by using the `show udp statistics` command in Exec mode. This command displays statistics for traffic destined to the ACE, rather than through the ACE. The syntax of this command is as follows:

`show udp statistics`

For example, to display UDP statistics for the current context, enter:

`host1/C1# show udp statistics`

Table 4-13 describes the fields in the `show udp statistics` command output.

**Table 4-13 Field Descriptions for the show udp statistics Command Output**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rcvd</td>
<td>Total number of UDP segments, errors, and segments with no port specified that the ACE received.</td>
</tr>
<tr>
<td>Sent</td>
<td>Total number of UDP segments sent by the ACE.</td>
</tr>
</tbody>
</table>
Displaying Service Policy Statistics

You can display service-policy statistics by using the `show service-policy` command in Exec mode. The syntax of this command is as follows:

```
show service-policy name
```

The `name` argument is a unique identifier for an existing service policy, specified as an unquoted text string with a maximum of 64 alphanumeric characters.

For example, to display service-policy statistics for the current context, enter:

```
host1/C1# show service-policy POLICY1
```

Table 4-14 describes the fields in the `show service-policy` command output.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface</td>
<td>VLAN identifier of the interface associated with the service policy.</td>
</tr>
<tr>
<td>Service-Policy</td>
<td>Identifier of the service policy.</td>
</tr>
<tr>
<td>Class</td>
<td>Identifier of the class map associated with the service policy.</td>
</tr>
<tr>
<td>Loadbalance</td>
<td></td>
</tr>
<tr>
<td>L7 Loadbalance Policy</td>
<td>Identifier of the Layer 7 load-balancing policy map associated with the service policy.</td>
</tr>
<tr>
<td>VIP Route Metric</td>
<td>Configured distance metric for the route that needs to be entered in the routing table.</td>
</tr>
<tr>
<td>VIP Route Advertise</td>
<td>State of route health injection (RHI) using the <code>loadbalance vip advertise</code> command. Possible values are ENABLED or DISABLED.</td>
</tr>
<tr>
<td>VIP ICMP Reply</td>
<td>State of the VIP’s ability to reply to ICMP requests. Possible values are ENABLED or DISABLED.</td>
</tr>
<tr>
<td>VIP State</td>
<td>Current status of the virtual IP address. Possible values are INSERVICE or OUTOFSERVICE.</td>
</tr>
<tr>
<td>Curr Conns</td>
<td>Number of active connections.</td>
</tr>
</tbody>
</table>
Displaying SYN Cookie Statistics

To display SYN cookie statistics, use the `show syn-cookie` command in Exec mode. To display SYN cookie statistics for all VLANs that are configured in the current context, enter the command with no arguments. The syntax of this command is as follows:

```
show syn-cookie [vlan number]
```

The optional `vlan number` keyword and argument instruct the ACE to display SYN cookie statistics for the specified interface. Enter an integer from 2 to 2024.

### Table 4-14 Field Descriptions for the show service-policy Command Output (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hit Count</td>
<td>Number of connections that the ACE.</td>
</tr>
<tr>
<td>Dropped Conns</td>
<td>Number of connections that the ACE discarded.</td>
</tr>
<tr>
<td>Client Pkt Count</td>
<td>Number of packets received from clients.</td>
</tr>
<tr>
<td>Client Byte Count</td>
<td>Number of bytes received from clients.</td>
</tr>
<tr>
<td>Server Pkt Count</td>
<td>Number of packets received from servers.</td>
</tr>
<tr>
<td>Server Byte Count</td>
<td>Number of bytes received from servers.</td>
</tr>
<tr>
<td>Max Conn Limit</td>
<td>Configured maximum number of connections limit.</td>
</tr>
<tr>
<td>Conn Rate Limit</td>
<td>Configured connection rate limit.</td>
</tr>
<tr>
<td>Bandwidth Rate Limit</td>
<td>Configured bandwidth rate limit.</td>
</tr>
<tr>
<td>Drop Count</td>
<td>Number of times that a connection was dropped because the connection rate limit or the bandwidth rate limit was reached.</td>
</tr>
</tbody>
</table>
For example, to display SYN cookie statistics for VLAN 100, enter:

```
host1/C1# show syn-cookie vlan 100
```

Table 4-15 describes the fields in the `show syn-cookie` command output.

**Table 4-15  Field Descriptions for the show syn-cookie Command Output**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface</td>
<td>Name of the VLAN interface configured on the ACE.</td>
</tr>
<tr>
<td>Configured TCP Embryonic Connection Limit</td>
<td>Configured embryonic connection threshold above which the ACE applies SYN-cookie DoS protection.</td>
</tr>
<tr>
<td>Current TCP Embryonic Connection Limit</td>
<td>Number of embryonic connections that the ACE is currently tracking.</td>
</tr>
<tr>
<td>Number of TCP SYNs Intercepted by SYN COOKIE</td>
<td>Number of client SYN packets that the ACE intercepted because the SYN-cookie embryonic connection threshold was exceeded.</td>
</tr>
<tr>
<td>Number of TCP ACKs Successfully Processed by SYN COOKIE</td>
<td>Number of client ACK packets that the ACE saw and that matched a given SYN cookie. Each client ACK that matches a cookie creates a valid embryonic connection on the ACE.</td>
</tr>
<tr>
<td>Failed Number of TCP ACKs Processed by SYN COOKIE</td>
<td>Number of client ACK packets that did not match a SYN cookie.</td>
</tr>
</tbody>
</table>
Clearing TCP/IP and UDP Connections and Statistics

The following sections describe the commands that you can use to clear connections and statistics related to TCP/IP and UDP connections, IP reassembly, and SYN Cookie.

- Clearing Connections
- Clearing Connection Statistics
- Clearing IP, TCP, and UDP Statistics
- Clearing IP Fragmentation and Reassembly Statistics
- Clearing SYN Cookie Statistics

Clearing Connections

You can clear ICMP, TCP, and UDP connections by using the `clear conn` command in Exec mode. The syntax of this command is as follows:

```
clear conn [all | flow {icmp | tcp | udp} | rserver]
```

The keywords are as follows:

- **all**—(Optional) Clears all connections to and through the ACE in the current context.
- **flow {icmp | tcp | udp}**—(Optional) Clears all connections of the specified flow type: ICMP, TCP, or UDP.
- **rserver**—(Optional) Clears all connections for the specified real server.

For example, to clear all TCP connections in the current context, enter:

```
host1/C1# clear conn flow tcp
```
Clearing Connection Statistics

You can clear all connection statistics in the current context by using the `clear stats conn` command in Exec mode. The syntax of this command is as follows:

```
clear stats conn
```

For example, to clear all connection statistics in the Admin context, enter:

```
host1/Admin# clear stats conn
```

Clearing IP, TCP, and UDP Statistics

Use the commands in this section to clear IP, TCP, and UDP statistics. This section contains the following topics:

- Clearing IP Statistics
- Clearing TCP Statistics
- Clearing UDP Statistics

Clearing IP Statistics

You can clear IP statistics by using the `clear ip statistics` command in Exec mode. This command clears all statistics associated with IP normalization, fragmentation, and reassembly in the current context. The syntax of this command is as follows:

```
clear ip statistics
```

For example, to clear IP statistics in the current context, enter:

```
host1/C1# clear ip statistics
```

Note

If you configured redundancy, then you need to explicitly clear IP statistics on both the active and the standby ACEs. Clearing statistics on the active module alone will leave the standby module’s statistics at the old values.
Clearing TCP Statistics

You can clear TCP statistics by using the `clear tcp statistics` command in Exec mode. This command clears all statistics associated with TCP connections and normalization in the current context. The syntax of this command is as follows:

```
clear tcp statistics
```

For example, to clear TCP statistics in the current context, enter:

```
host1/C1# clear tcp statistics
```

**Note**

If you configured redundancy, then you need to explicitly clear TCP statistics on both the active and the standby ACEs. Clearing statistics on the active module alone will leave the standby module’s statistics at the old values.

Clearing UDP Statistics

You can clear UDP statistics by using the `clear udp statistics` command in Exec mode. This command clears all statistics associated with UDP connections in the current context. The syntax of this command is as follows:

```
clear udp statistics
```

For example, to clear UDP statistics in the current context, enter:

```
host1/C1# clear udp statistics
```

**Note**

If you configured redundancy, then you need to explicitly clear UDP statistics on both the active and the standby ACEs. Clearing statistics on the active module alone will leave the standby module’s statistics at the old values.
Clearing IP Fragmentation and Reassembly Statistics

You can clear IP fragmentation and reassembly statistics by using the `clear interface` command in Exec mode. The syntax of this command is as follows:

```
clear interface [vlan vlan_id]
```

The optional `vlan_id` argument is a unique identifier of an existing interface as an integer from 2 to 4094. If you omit the `vlan` keyword and `vlan_id` argument, you can clear fragmentation and reassembly statistics for all interfaces in the context.

For example, to clear IP fragmentation and reassembly statistics for all interfaces in the C1 context, enter:

```
host1/C1# clear interface
```

---

**Note**

If you configured redundancy, then you need to explicitly clear IP fragmentation and reassembly statistics on both the active and the standby ACEs. Clearing statistics on the active module alone will leave the standby module’s statistics at the old values.

---

Clearing SYN Cookie Statistics

To clear the SYN cookie statistics, use the `clear syn-cookie` command. To clear SYN cookie statistics for all VLANs that are configured in the current context, enter the command with no arguments. The syntax of this command is as follows:

```
clear syn-cookie [vlan number]
```

The optional number argument instructs the ACE to clear SYN cookie statistics for the specified interface. Enter an integer from 2 to 2024.

For example, to clear SYN cookie statistics for VLAN 100, enter:

```
host1/C1# clear syn-cookie vlan 100
```