CHAPTER 3

Configuring Application Protocol Inspection

This chapter describes how to configure application protocol inspection for the Cisco Application Control Engine (ACE) module. Application protocol inspection provides functionality for several protocols that carry Layer 3 and Layer 4 information in the application payload, require some form of deep packet inspection of the HTTP protocol, or require File Transfer Protocol (FTP) request command filtering.

This chapter contains the following major sections:

Application Protocol Inspection Overview

Certain applications require special handling of the data portion of a packet as the packets pass through the ACE. Application protocol inspection helps to verify the protocol behavior and identify unwanted or malicious traffic that passes through the ACE. Based on the specifications of the traffic policy, the ACE accepts or rejects the packets to ensure the secure use of applications and services.

This section contains the following topics on application protocol inspection:

Performing Application Protocol Inspection

You can configure the ACE to perform application protocol inspection, sometimes referred to as an application protocol “fixup” for applications that do the following:

- Embed IP addressing information in the data packet including the data payload.
- Open secondary channels on dynamically assigned ports.

You may require the ACE to perform application inspection of Domain Name System (DNS), FTP (File Transfer Protocol), HTTP, Internet Control Message Protocol (ICMP), Internet Locator Service (ILS), Real-Time Streaming Protocol (RTSP), Skinny Client Control Protocol (SCCP), and Session Initiation Protocol (SIP) as a first step before passing the packets to the destination server. For HTTP, the ACE performs deep packet inspection to statefully monitor the HTTP protocol and permit or deny traffic based on user-defined traffic policies. HTTP deep packet inspection focuses mainly on HTTP attributes such as the HTTP header, the URL, and the payload. For FTP, the ACE performs FTP command inspection for FTP sessions, allowing you to restrict specific commands by the ACE.

Application inspection helps you to identify the location of the embedded IP addressing information in the TCP or UDP flow. This inspection allows the ACE to translate embedded IP addresses and to update any checksum or other fields that are affected by the translation.

Translating IP addresses embedded in the payload of protocols is especially important for NAT (explicitly configured by the user) and server load balancing (an implicit NAT).

Application inspection also monitors TCP or UDP sessions to determine the port numbers for secondary channels. Some protocols open secondary TCP or UDP ports to improve performance. The initial session on a well-known port is used to negotiate dynamically assigned port numbers. The application protocol inspection function monitors these sessions, identifies the dynamic port assignments, and permits data exchange on these ports for the duration of the session.

Table 3-1 describes the application inspection protocols supported by the ACE, the default TCP or UDP protocol and port, and whether the protocol is compatible with Network Address Translation (NAT) and Port Address Translation (PAT).

Table 3-1 Application Inspection Support

Application Protocol	Transport Protocol	Port	NAT/PAT Support	Enabled by Default	Standards[1]	Comments/Limitations
DNS	UDP	Src—Any Dest—53	NAT	No	RFC 1123	Inspects DNS packets destined to port 53. You can specify the maximum length of the DNS packet to be inspected. See the "DNS Inspection" section for more information.
FTP	TCP	Src—Any Dest—21	Both	No	RFC 959	Inspects FTP packets, translates address and port embedded in the payload, and opens up a secondary channel for data. See the "FTP Inspection" section for more information.
FTP strict	TCP	Src—Any Dest—21	Both	No	RFC 959	The **inspect ftp strict** command allows the ACE to track each FTP command and response sequence and also prevents an FTP client from determining valid usernames that are supported on an FTP server. See the "FTP Inspection" section for more information.
HTTP	TCP	Src—Any Dest—80	Both	No	RFC 2616	Inspects HTTP packets. See the "HTTP Deep Packet Inspection" section for more information.
ICMP	ICMP	Src—N/A Dest—N/A	Both	No	—	See the "ICMP Inspection" section for more information.

Table 3-1 Application Inspection Support (continued)

Application Protocol	Transport Protocol	Port	NAT/PAT Support	Enabled by Default	Standards[1]	Comments/Limitations
ICMP error	ICMP	Src—N/A Dest—N/A	NAT	No	—	The **error** keyword supports NAT of ICMP error messages. When you enable ICMP error inspection, the ACE creates translation sessions for intermediate hops that send ICMP error messages, based on the NAT configuration. The ACE overwrites the packet with the translated IP addresses. See the "ICMP Inspection" section for more information.
ILS	TCP	Src—Any Dest—389	NAT	No	RFC 2251 (LDAPv3) Includes support for RFC 1777 (LDAPv2)	Referral requests and responses are not supported. Users in multiple directories are not unified. Single users having multiple identities in multiple directories cannot be recognized by NAT.

Table 3-1 Application Inspection Support (continued)

Application Protocol	Transport Protocol	Port	NAT/PAT Support	Enabled by Default	Standards[1]	Comments/Limitations
RTSP	TCP	Src—Any Dest—554	NAT	No	RFC 2326, RFC 2327, RFC 1889	Inspects RTSP packets and translates the payload according to NAT rules. The ACE opens up the secondary channels for audio and video. Not all the RTSP methods (packet types) specified in the RFC are supported. See the "RTSP Inspection" section for more information.
SCCP	TCP	Src—Any Dest—2000	NAT	No	—	The ACE does not support PAT with SCCP.
SIP	TCP and UDP	Src—Any Dest—5060	NAT	No	RFC 2543, RFC 3261, RFC 3265, RFC 3428	The ACE does not support PAT with SIP.

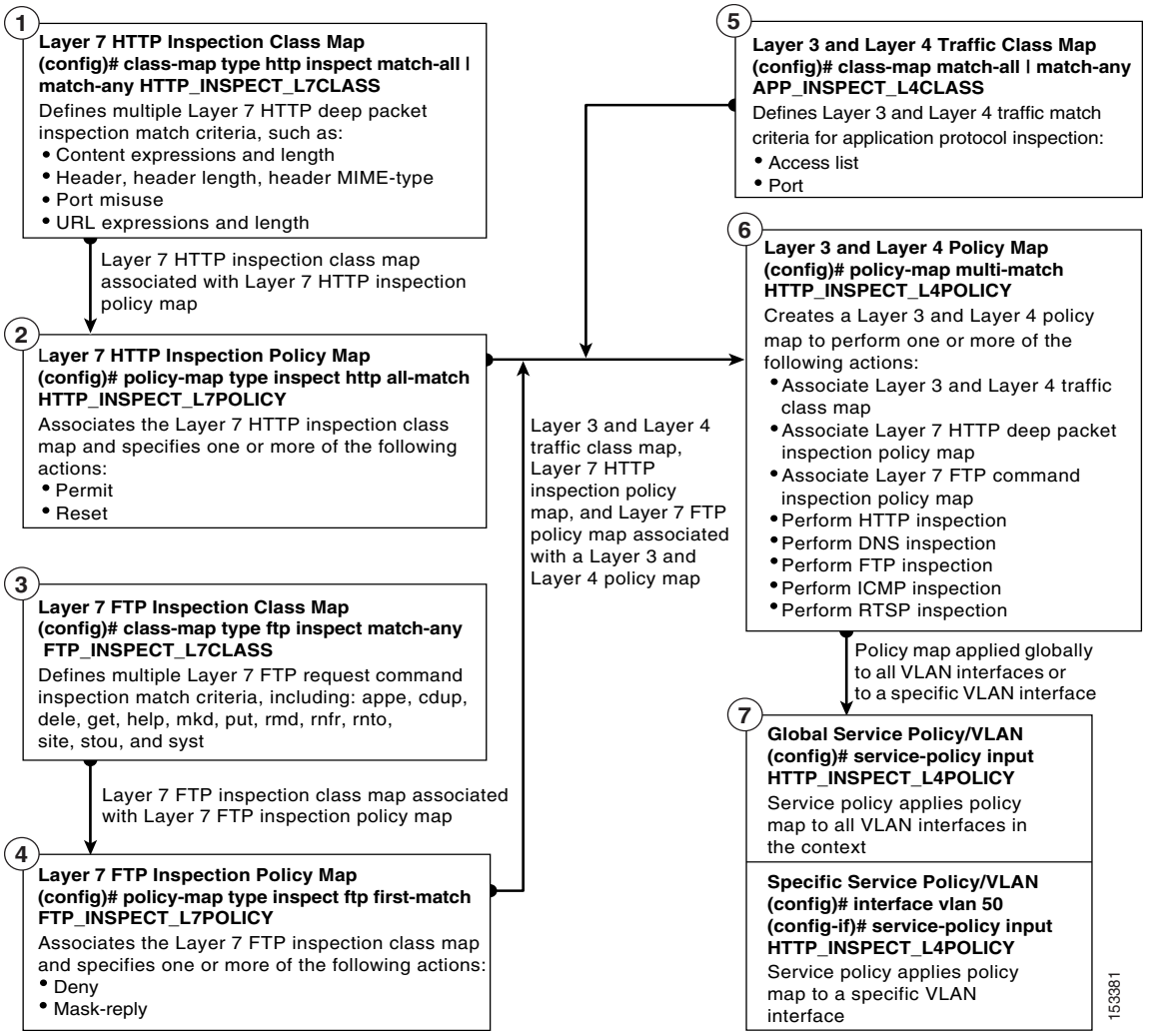
1. The ACE is in compliance with these standards, but it does not enforce compliance on packets being inspected. For example, FTP commands are supposed to be in a particular order, but the ACE does not enforce the order.

You configure rules for application protocol inspection through class maps, policy maps, and service policies. The following items summarize the role of each function in configuring application protocol inspection:

- Layer 7 class map—Provides the Layer 7 network traffic classification to identify protocol inspection attributes (such as the HTTP header and the URL) and FTP request commands.
- Layer 7 policy map—Configures the applicable match statements and actions that the ACE executes on the network traffic that matches the classifications defined in the Layer 7 class map.
- Layer 3 and Layer 4 class map—Classifies the network traffic that passes through the ACE for application inspection and matches the traffic associated with the specified **inspect** commands in a policy map.
- Layer 3 and Layer 4 policy map—Enables DNS, FTP, HTTP, ICMP, ILS, RTSP, SCCP, and SIP protocol inspection and FTP command inspection for a traffic classification that matches the criteria listed in the class map.
- Service policy—Activates the policy map and attaches the traffic policy to a VLAN interface or globally on all VLAN interfaces.

Figure 3-1 provides an overview of the process required to configure class maps and policy maps to perform application protocol inspection. The flow chart also shows how the ACE associates the various components of the class map and policy map configuration with each other.

Figure 3-1 Application Protocol Inspection Configuration Flow Diagram



Application Inspection Protocol Overview

This section provides an overview of the application inspection protocols supported by the ACE and contains the following topics:

DNS Inspection

Domain Name System (DNS) inspection performs the following tasks:

- Monitors the message exchange to ensure that the ID of the DNS response matches the ID of the DNS query.
- Allows one DNS response for each DNS query in a UDP connection. The ACE removes the DNS session associated with the DNS query as soon as the DNS reply is forwarded.
- Translates the DNS A-record based on the NAT configuration. Only forward lookups are translated using NAT; the ACE does not handle pointer (PTR) records.

Note The DNS rewrite function is not applicable for PAT because multiple PAT rules apply to each A-record. Using multiple PAT rules makes it difficult for the ACE to properly choose the correct PAT rule.

- Performs a maximum DNS packet length check to verify that the maximum length of a DNS reply is no greater than the value specified in the **inspect dns** command.

Note If you enter the **inspect dns** command without specifying the **maximum-length** optional keyword, the ACE does not check the DNS packet size.

- Performs a number of security checks as follows:
  - Verifies that the maximum label length is no greater than 63 bytes
  - Verifies that the maximum domain name length is no greater than 255 bytes
  - Checks for the existence of compression loops

A single connection is created for multiple DNS sessions if the DNS sessions are between the same two hosts and the sessions have the same 5-tuple (source and destination IP address, source and destination port, and protocol). DNS identification is tracked by *app_id*, and the idle timer for each *app_id* runs independently.

Because the *app_id* expires independently, a legitimate DNS response can only pass through the ACE within a limited period of time and there is no resource build-up. However, if you enter the **show conn** command, you will see the idle timer of a DNS connection being reset by a new DNS session. This reset action is due to the shared DNS connection.

FTP Inspection

FTP inspection inspects FTP sessions for address translation in a message, dynamic opening of ports, and stateful tracking of request and response messages. Each specified FTP command must be acknowledged before the ACE allows a new command. Command filtering allows you to restrict specific commands by the ACE. When the ACE denies a command, it closes the connection.

The ACE performs the FTP command inspection process as follows:

- Prepares a dynamic secondary data connection. The channels are allocated in response to a file upload, a file download, or a directory listing event and must be prenegotiated. The port is negotiated through the PORT or PASV commands.

- Tracks the FTP command-response sequence. The ACE performs the following FTP command checks listed below.
  - Truncated command—Checks the number of commas in the PORT and PASV reply command against a fixed value of five. If the value is not five, the ACE assumes that the PORT command is truncated, issues a warning message, and closes the TCP connection.
  - Incorrect command—Checks the FTP command to verify if it ends with <CR><LF> characters, as required by RFC 959. If the FTP command does not end with those characters, the ACE closes the connection.
  - Invalid port negotiation—Checks the negotiated dynamic port value to verify that it is greater than 1024 (port numbers from 2 to 1024 are reserved for well-known connections). If the negotiated port falls in this range, the ACE closes the TCP connection.
  - Command pipelining—Checks the number of characters present after the port numbers in the PORT and PASV reply command against a constant value of 8. If the number of characters is greater than 8, the ACE closes the TCP connection.

  In addition to these FTP command checks, if you specify the **strict** keyword with the **inspect ftp** command in a Layer 3 and Layer 4 policy map, the ACE tracks each FTP command and response sequence for the anomalous activity outlined below. The **strict** keyword can be used with a Layer 7 FTP policy map (nested within the Layer 3 and Layer 4 policy map) to deny certain FTP commands or to mask the server reply for the SYST command.

Note Using the **strict** keyword may affect FTP clients that do not comply with the RFC standards.

  - Size of RETR and STOR commands—Checks the size of the RETR and STOR commands against a fixed constant of 256. If the size is greater, the ACE logs an error message and closes the connection.
  - Command spoofing—Verifies that the PORT command is always sent from the client. If a PORT command is sent from the server, the ACE denies the TCP connection.
  - Reply spoofing—Verifies that the PASV reply command (227) is always sent from the server. If a PASV reply command is sent from the client, the ACE denies the TCP connection. This denial prevents a security hole when the user executes “227 xxxxx a1, a2, a3, a4, p1, p2.”

- Translates embedded IP addresses with NAT. FTP command inspection translates the IP address within the application payload. See RFC 959 for more details.

HTTP Deep Packet Inspection

The ACE performs a stateful deep packet inspection of the HTTP protocol. Deep packet inspection is a special case of application inspection where the ACE examines the application payload of a packet or a traffic stream and makes decisions based on the content of the data. During HTTP deep inspection, the main focus of the application inspection process is on HTTP attributes such as the HTTP header, the URL, and to a limited extent, the payload. User-defined regular expressions can also be used to detect "signatures" in the payload.

You define policies to permit or deny the traffic, or to send a TCP reset message to the client or server to close the connection.

The security features covered by HTTP application inspection are as follows:

- RFC compliance monitoring and RFC method filtering
- Content, URL, and HTTP header length checks
- Transfer-encoding methods
- Content type verification and filtering
- Port 80 misuse

ICMP Inspection

Internet Control Message Protocol (ICMP) inspection allows ICMP traffic to have a "session" so that it can be inspected similarly to TCP and UDP traffic. If you do not use ICMP inspection, we recommend that you do not create an ACL that allows ICMP traffic to pass through the ACE. Without stateful inspection, ICMP can be used to attack your network. ICMP inspection ensures that there is only one response for each request, and that the sequence number is correct.

For stateful ICMP, state information, as maintained for TCP or UDP flows, is maintained for ICMP instead of performing only the ACL and NAT functions. The maintenance of ICMP state information is required to resolve the following problems:

- ICMP reply messages without request messages
- Unsolicited ICMP error messages
- Unknown ICMP types

ICMP error messages are generated by intermediate nodes situated on the network path to a destination whenever a packet sent to that destination cannot be forwarded. ICMP error messages may also be generated by endpoint nodes, as in the case of port unreachable errors. ICMP error messages carry the original packet for which the error is generated in the data part of the message. The error message also contains the addresses of the intermediate node or endpoint node in the outer header and the destination node in the inner header.

ICMP error fixup handles address translation of node address and destination address to global addresses using the NAT configuration. ICMP error fixup is user configurable. If you do not enable this feature, intermediate node or endpoint node addresses are translated in the same way as the destination address of the embedded packet. As a result, error messages appear as if they are originating from the destination and the node addresses or the route to the destination are not included.

ICMP inspection performs the following tasks for ICMP request or reply messages:

- Creates a bidirectional session or connection record. The lookup key in the forward direction is the source IP address, destination IP address, protocol, ICMP type, ICMP identifier, and VLAN.
- Verifies that the connection record contains a sequence number window that specifies the list of sequence numbers of outstanding requests for which replies are pending.
- Verifies that the connection record has a timeout, so that the inactive connection record can be reused for other flows and can protect the inside network against fraudulent ICMP reply packets.
- Allows reply packets only if a valid connection record exists and prevents the reply packets from passing through an ACL again if the connection record (or the state information) exists.
- Creates a connection record for the transit ICMP request or reply packets and also for those packets addressed to or from the ACE.

ICMP error message inspection performs the following tasks:

- Extracts the embedded IP header in the ICMP error message and checks for the presence of a connection record that corresponds to the embedded packet for which the error message has been generated.

- Performs an ACL of the ICMP error message regardless of the existence of a session for the embedded packet. The ICMP error message is itself stateless and requires access control.
- Allocates NAT translation entries (xlate) for intermediate nodes or endpoint nodes to perform NAT of a local IP address to a global IP address in any ICMP error message.
- Updates the checksum in the outer and inner headers.

ILS Inspection

Internet Locator Service (ILS) is used by Microsoft NetMeeting to help users find other users. ILS interfaces with the Lightweight Directory Access Protocol (LDAP) to provide directory services. The ACE ILS inspection feature provides NAT support for NetMeeting, Site Server, and Active Directory products that use LDAP to exchange directory information with an ILS server. The ACE does not support PAT for ILS because the LDAP database stores only IP addresses and not ports.

ILS/LDAP follows the client/server model and uses a single TCP connection for each session. Depending on the client actions, several sessions may be required. During the connection setup, the client sends a BIND protocol data unit (PDU) to the server. After the client receives the BIND RESPONSE from the server, other messages (for example, ADD, DEL, SEARCH, or MODIFY) can be exchanged to perform operations on the ILS Directory.

The ADD REQUEST and SEARCH REQUEST PDUs may contain addresses of NetMeeting peers. NetMeeting version 2.x and 3.x provide ILS support.

Because ILS traffic occurs only on the secondary UDP channel, the ACE disconnects the TCP connection after the TCP inactivity interval has elapsed. By default, the TCP inactivity is 60 minutes, but you can adjust it using a connection parameter map. For information about configuring a connection parameter map, see Chapter 1, Configuring TCP/IP Normalization and IP Reassembly Parameters.

The ACE performs the following ILS inspection operations:

- Decodes the LDAP REQUEST/RESPONSE PDUs using the Basic Encoding Rules (BER) decoder functions
- Parses the LDAP packet
- Extracts IP addresses

- Translates IP addresses as necessary
- Encodes the PDU with translated addresses using BER encode functions
- Copies the newly encoded PDU back to the TCP packet
- Performs an incremental TCP checksum and sequence number adjustment

The following restrictions apply to the ACE ILS inspection feature:

- Referral requests and responses are not supported.
- Users in multiple directories are not unified.
- Single users having multiple identities in multiple directories cannot be recognized by NAT.

RTSP Inspection

The Real-Time Streaming Protocol (RTSP) is used by RealAudio, RealNetworks, Apple QuickTime 4, RealPlayer, and Cisco IP/TV connections. RTSP applications use the well-known port 554 with TCP and UDP as the control channel. The ACE supports TCP only in conformity with RFC 2326.

The TCP control channel negotiates the data channels used to transmit audio and video traffic, depending on the transport mode that is configured on the client. The supported data transport modes are rtp/avp, rtp/avp/udp, x-real-rdt, x-real-rdt/udp, and x-pn-tng/udp. The data transport types rtp/avp/tcp and x-real-rdt/tcp use the control channel to stream data. RTSP inspection is not required in this case to open a secure port (pinhole) for the data channel.

The ACE parses SETUP response messages with a status code of 200.

Because RFC 2326 does not require that the client and server ports are contained in the SETUP response message, the ACE must track the state and remember the client ports in the SETUP message. QuickTime places the client ports in the SETUP message; the server responds with only the server ports.

During RTSP inspection, the ACE does not do the following:

- Inspect RTSP messages that pass through UDP ports.
- Support RealNetworks multicast mode (x-real-rdt/mcast).
- Support the ability to recognize HTTP cloaking where RTSP messages are hidden in HTTP messages.

- Perform NAT on RTSP messages because the embedded IP addresses are contained in the Session Description Protocol (SDP) files as part of HTTP or RTSP messages.

The following additional restrictions apply to RTSP inspection as performed by the ACE:

- With Cisco IP/TV, the number of translations that the ACE performs on the SDP part of the message is proportional to the number of program listings in the Content Manager (each program listing can have at least six embedded IP addresses).
- When using RealPlayer, you must properly configure the transport mode. For the ACE, add an ACL classification from the server to the client. For RealPlayer, change the transport mode by clicking **Tools>Preferences> Connection>Network Transport>RTSP Settings**.
  - If you use TCP mode on the RealPlayer, check the **Attempt to use TCP for all content** check box. It is not necessary to configure RTSP application inspection on the ACE.
  - If you use UDP mode on the RealPlayer, check the **Attempt to use UDP for all content** check box. Configure RTSP application inspection on the ACE.

SCCP Inspection

Skinny Client Control Protocol (SCCP) is used in VoIP networks, for example, with Cisco IP phones and Cisco CallManager. The ACE supports all versions of the SCCP protocol through version 3.3.2.

SCCP inspection provides the following operations:

- Supports NAT for embedded IP addresses and ports.
- Dynamically opens secure ports.
- Drops messages with an SCCPPrefix length that is less than the message ID length (configurable).
- Supports video.
- Validates message ID length (configurable maximum).
- Ensures that only registered clients can make calls. This feature is configurable and disabled by default.
- Allows you to configure timeouts.

SIP Inspection

Session Initiation Protocol (SIP) is used for call handling sessions, especially two-party conferences. SIP works with SDP for call signaling.

SIP inspection provides the following operations:

- Translates the SIP text-based messages, recalculates the content length for the SDP portion of the message, and recalculates the packet length and checksum.
- Dynamically opens media connections for ports specified in the SDP portion of the SIP message as addresses and ports on which the endpoint should listen.
- Opens RTP and RTCP connections between the two endpoints using media addresses and ports that are maintained in a SIP inspection database with CALL_ID, FROM, and TO indices from the SIP header. These indices identify the call, the source, and the destination.
- Performs RFC 3261 compliance checks, including checking the Request Message to ensure it is one of the predefined methods: OPTIONS, INVITE, REGISTER, ACK, CANCEL, BYE and validates their syntax.
- Checks whether a SIP message is compliant with the following RFC extensions:
  - RFC 2976 (INFO)
  - RFC 3262 (PRACK)
  - RFC 3265 (SUBSCRIBE/NOTIFY)
  - RFC 3311 (UPDATE)
  - RFC 3515 and RFC 3892 (REFER)
  - RFC 3428 (MESSAGE)
- Enforces the mandatory header fields (From, To, Call-Id, CSeq, Via, Max-Forwards) presence and validity.
- Enforces forbidden header fields.
- Checks URI in Header fields against a permit or deny list of callers or callees. If the user is not entitled to talk to any host on the protected network, the SIP ACE module will generate a SIP message (Response 603 Decline).
- Checks the Via field to deny messages from specific SIP proxy servers.

- Checks the validity of each header parameter in the context of each message following the syntax rules specified in RFC 3261.
- Removes the optional User-Agent and Server header fields to hide the endpoint software version.
- Checks the Max-Forwards header field. If the Max-Forwards value reaches 0 before the request reaches its destination, the ACE rejects the request with a 483 (Too Many Hops) error response.
- Validates SIP URIs and URIs present in the SIP header fields.
- Handles unknown SIP methods. Because SIP is an evolving protocol, which includes many extensions, some of the new methods may not be recognized by the ACE (only the methods defined by RFC 3261 and the extensions listed above are supported). You can configure how the ACE handles “unknown” SIP methods.
- Permits or denies third-party registrations or deregistrations and specifies which users are allowed to perform these functions. If this policy is enabled, REGISTER messages, with mismatched To and From headers and with From values that do not match any of the privileged user IDs, are dropped.
- Protects against buffer overflows as follows:
  - Enforces the Content-Length and the Content-Type (user configurable) values:
  - Allows you to configure the maximum size of a SIP message body. When a request or response SIP message passes through the ACE module, the message is checked to ensure that it meets the size constraints. If it does not, the action configured for this policy by the user will be executed.
  - Cross checks the Content-Length header field value with the actual message size.
  - Allows you to select whether a subset of Content-types are permitted through the ACE module. You can specify the Content-type string in the form of a regular expression, for example, Application/SDP, text/html. The default behavior is to allow all types.
  - Enforces SIP or SIPS URI length (user configurable).

- Enables or disables Instant Messenger (IM):
  - Allows you to disable IM over SIP, which causes the ACE to drop all messages belonging to IM as specified by SIMPLE RFC extensions. An appropriate warning message is displayed to call out the exact methods that this feature drops.
  - You can specify a list of users (in the form of a regex) that are not allowed to use IM through the ACE module.
- Allows you to configure which SIP methods that the ACE supports. You can also specify if additional SIP methods (that are not part of the RFCs or RFC extensions that the ACE is compliant with) should be denied. The ACE maintains the list of invalid methods as a regex table.
- Enables you to hide or remove risky header fields (for example, Alert-Info and Call-Info) that, if provided by a malicious caller, may cause the callee to display inappropriate, offensive, dangerous, or illegal content.
- Allows you to enable IP address privacy. If both the caller and the callee are on the inside network and on the same subnet, and the proxy is on the outside network, there is a possibility that the two parties may try to contact each other by bypassing the proxy. If enabled, this feature prevents such direct contact because the embedded addresses in the message from the proxy to the callee are not fixed. Therefore, the callee cannot learn the real IP address of the caller.

Application Protocol Inspection Configuration Quick Start Procedures

Table 3-2, Table 3-3, and Table 3-4 provide a quick overview of the steps required to configure application protocol inspection on the ACE:

- See Table 3-2 for a quick overview on configuring Layer 7 FTP request command inspection.
- See Table 3-3 for a quick overview on configuring Layer 7 HTTP deep inspection.
- See Table 3-4 for a quick overview on configuring Layer 3 and Layer 4 DNS, FTP, HTTP, ICMP, and RTSP application protocol inspection.

Table 3-2 Layer 7 FTP Request Command Inspection Quick Start

Task and Command Example

1. If you are operating in multiple context mode, observe the CLI prompt to verify that you are operating in the desired context. Change to the correct context if necessary.
 	
`host1/Admin# changeto C1`
`host1/C1#`
 	
For details on creating contexts, see the *Cisco Application Control Engine Module Virtualization Configuration Guide*.

2. Enter configuration mode.
 	
`host1/Admin# config`
`Enter configuration commands, one per line. End with CNTL/Z`
`host1/Admin(config)#`

3. Create a Layer 7 class map that is used for the inspection of FTP request commands. If you do not specify **match-all** or **match-any**, traffic must match all the match criteria to be classified as part of the traffic class.
 	
The CLI displays the class map FTP command inspection configuration mode.
 	
`host1/Admin(config)# class-map type ftp inspect match-any FTP_INSPECT_L7CLASS`
`host1/Admin(config-cmap-ftp-insp)#`

Table 3-2 Layer 7 FTP Request Command Inspection Quick Start

Task and Command Example

4. Configure the Layer 7 class map to define FTP request command inspection decisions through the ACE. The **match request** command identifies the FTP commands that you want filtered by the ACE.
 	
`host1/Admin(config-cmap-ftp-insp)# match request-method mkdir`
`host1/Admin(config-cmap-ftp-insp)# exit`
`host1/Admin(config)#`

5. Create and configure a Layer 7 policy map that enables FTP command inspection. Specify the actions that you want to apply to the Layer 7 user-defined class map and, if appropriate, to the default class map.
 	
`host1/Admin(config)# policy-map type inspect ftp first-match FTP_INSPECT_L7POLICY`
`host1/Admin(config-pmap-ftp-ins)# class FTP_INSPECT_L7CLASS`
`host1/Admin(config-pmap-ftp-ins-c)# deny`
`host1/Admin(config-pmap-ftp-ins-c)# exit`
`host1/Admin(config)#`

6. Create a Layer 3 and Layer 4 class map to classify network traffic that passes through the ACE for FTP command inspection. If you do not specify **match-all** or **match-any**, traffic must match all the match criteria to be classified as part of the traffic class.
 	
The CLI displays the class map configuration mode.
 	
`host1/Admin(config)# class-map match-all FTP_INSPECT_L4CLASS`
`host1/Admin(config-cmap)#`
 	
Include one or more of the **match** commands as part of the Layer 3 and Layer 4 class map.
 	
`host1/Admin(config-cmap)# description FTP command inspection of incoming traffic`
`host1/Admin(config-cmap)# match port tcp eq 21`
`host1/Admin(config-cmap)# exit`
`host1/Admin(config)#`

Table 3-2 Layer 7 FTP Request Command Inspection Quick Start

Task and Command Example

7. Create a Layer 3 and Layer 4 policy map and associate the Layer 7 FTP command inspection policy map to activate the operation. Specify the actions that you want to apply to the Layer 3 and Layer 4 user-defined class map and, if appropriate, to the default class map.
`host1/Admin(config)# policy-map multi-match FTP_INSPECT_L4POLICY`
`host1/Admin(config-pmap)# class FTP_INSPECT_L4CLASS`
`host1/Admin(config-pmap-c) inspect ftp strict policy FTP_INSPECT_L7POLICY`
`host1/Admin(config-pmap-c)# exit`
`host1/Admin(config)#`

8. Attach the Layer 3 and Layer 4 traffic policy to a single VLAN interface or globally to all VLAN interfaces, and specify the direction in which the policy should be applied. For example, to specify a VLAN interface and apply multiple service policies to the VLAN, enter:
`host1/Admin(config)# interface vlan 50`
`host1/Admin(config-if)# ip address 172.16.1.100 255.255.255.0`
`host1/Admin(config-if)# service-policy input FTP_INSPECT_L4POLICY`

9. (Optional) Save your configuration changes to flash memory.
`host1/Admin(config)# exit`
`host1/Admin# copy running-config startup-config`

Table 3-3 Layer 7 HTTP Protocol Deep Inspection 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, log directly in to, or change to, the correct context.
`host1/Admin# changeto C1`
`host1/C1#`
The rest of the examples in this table use the Admin context, unless otherwise specified. For details on creating contexts, see the *Cisco Application Control Engine Module Virtualization Configuration Guide*.

2. Enter configuration mode.
`host1/Admin# config`
`Enter configuration commands, one per line. End with CNTL/Z`
`host1/Admin(config)#`

3. Create a Layer 7 class map that is used for the deep packet inspection of HTTP traffic. If you do not specify **match-all** or **match-any**, traffic must match all the match criteria to be classified as part of the traffic class.
The CLI displays the class map HTTP application protocol inspection configuration mode.
`host1/Admin(config)# class-map type http inspect match-any HTTP_INSPECT_L7CLASS`
`host1/Admin(config-cmap-http-insp)#`
Include one or more of the **match** commands listed in Steps 4 though 13 as part of the Layer 7 HTTP deep packet inspection class map.

4. (Optional) Configure the class map to define HTTP application inspection decisions based on content expressions contained within the HTTP content. `host1/Admin(config-cmap-http-insp)# match content .*newp2psig`
5. (Optional) Configure the class map to define application inspection decisions in the HTTP content up to the configured maximum content parse length. `host1/Admin(config-cmap-http-insp)# match content length eq 1000`

Table 3-3 Layer 7 HTTP Protocol Deep Inspection Quick Start (continued)

Task and Command Example
6. (Optional) Configure the class map to define application inspection decisions based on the name and value in an HTTP header. `host1/Admin(config-cmap-http-insp)# match header Host header-value .mycompanyexample.com`
7. (Optional) Limit the HTTP traffic allowed through the ACE based on the length of the entity-body in the HTTP message. `host1/Admin(config-cmap-http-insp)# match header length request eq 256`

8. (Optional) Specify a subset of the Multipurpose Internet Mail Extension (MIME)-type messages to be permitted or denied by the ACE.
`host1/Admin(config-cmap-http-insp)# match header mime-type audio\midi`
`host1/Admin(config-cmap-http-insp)# match header mime-type audio\mpeg`

9. (Optional) Configure the class map to define application inspection compliance decisions that restrict certain HTTP traffic from passing through the ACE. `host1/Admin(config-cmap-http-insp)# match port-misuse p2p`

10. (Optional) Configure the class map to define application inspection compliance decisions based on the request methods defined in RFC 2616 and by HTTP extension methods.
`host1/Admin(config-cmap-http-insp)# match request-method rfc connect`
`host1/Admin(config-cmap-http-insp)# match request-method rfc get`
`host1/Admin(config-cmap-http-insp)# match request-method rfc head`
`host1/Admin(config-cmap-http-insp)# match request-method ext index`

11. (Optional) Configure the class map to define application inspection decisions that limit the HTTP transfer-encoding types that can pass through the ACE. `host1/Admin(config-cmap-http-insp)# match transfer-encoding chunked`

Table 3-3 Layer 7 HTTP Protocol Deep Inspection Quick Start (continued)

Task and Command Example

12. (Optional) Configure the class map to define application inspection decisions based on the URL name.
`host1/Admin(config-cmap-http-insp)# match url .*.gif`
`host1/Admin(config-cmap-http-insp)# match url .*.html`

13. (Optional) Limit the HTTP traffic allowed through the ACE by specifying the maximum length of a URL in a request message that can be received by the ACE. `host1/Admin(config-cmap-http-insp)# match url length eq 10000`

14. Create and configure a Layer 7 policy map that enables the deep packet inspection of the HTTP protocol. Specify the actions that you want to apply to the Layer 7 user-defined class map and, if appropriate, to the default class map.
`host1/Admin(config)# policy-map type inspect http all-match HTTP_INSPECT_L7POLICY`
`host1/Admin(config-pmap-ins-http)# class HTTP_INSPECT_L7CLASS`
`host1/Admin(config-pmap-ins-http-c)# permit`
`host1/Admin(config-pmap-ins-http-c)# exit`
`host1/Admin(config-pmap-ins-http)# exit`
`host1/Admin(config)#`

15. Create a Layer 3 and Layer 4 class map to classify network traffic that passes through the ACE for HTTP deep packet inspection. If you do not specify **match-all** or **match-any**, traffic must match all the match criteria to be classified as part of the traffic class.
The CLI displays the class map configuration mode.
`host1/Admin(config)# class-map match-all HTTP_INSPECT_L4CLASS`
`host1/Admin(config-cmap)#`
Include one or more of the **match** commands as part of the Layer 3 and Layer 4 class map.
`host1/Admin(config-cmap)# description HTTP protocol deep inspection of incoming traffic`
`host1/Admin(config-cmap)# match port tcp eq 80`
`host1/Admin(config-cmap)# exit`
`host1/Admin(config)#`

Table 3-3 Layer 7 HTTP Protocol Deep Inspection Quick Start (continued)

Task and Command Example

16. Create a Layer 3 and Layer 4 policy map and associate the Layer 7 HTTP deep packet inspection policy map to activate the operation. Specify the actions that you want to apply to the Layer 3 and Layer 4 user-defined class map and, if appropriate, to the default class map.
`host1/Admin(config)# policy-map multi-match HTTP_INSPECT_L4POLICY`
`host1/Admin(config-pmap)# class HTTP_INSPECT_L4CLASS`
`host1/Admin(config-pmap-c)# inspect http policy HTTP_INSPECT_L7POLICY`
`host1/Admin(config-pmap-c)# exit`
`host1/Admin(config-pmap)# exit`
`host1/Admin(config)#`

17. Attach the Layer 3 and Layer 4 traffic policy to a single VLAN interface or globally to all VLAN interfaces. For example, to specify a VLAN interface and apply multiple service policies to the VLAN, enter:
`host1/Adminhost1/Admin(config)#interface vlan50`
`host1/Admin(config-if)# ip address 172.16.1.100 255.255.255.0`
`host1/Admin(config-if)# service-policy input HTTP_INSPECT_L4POLICY`

18. (Optional) Save your configuration changes to flash memory.
`host1/Admin(config)# exit`
`host1/Admin# copy running-config startup-config`

Table 3-4 Layer 3 and Layer 4 Application Protocol Inspection Quick Start

Task and Command Example

1. If you are operating in multiple context mode, observe the CLI prompt to verify that you are operating in the desired context. Change to the correct context if necessary.
`host1/Admin# changeto C1`
`host1/C1#`
For details on creating contexts, see the *Cisco Application Control Engine Module Virtualization Configuration Guide*.

2. Enter configuration mode.
`host1/Admin# config`
`Enter configuration commands, one per line. End with CNTL/Z`
`host1/Admin(config)#`

3. Create a Layer 3 and Layer 4 class map to classify network traffic that passes through the ACE for DNS, FTP, HTTP, ICMP, ILS, RTSP, SCCP, or SIP application protocol inspection. If you do not specify **match-all** or **match-any**, traffic must match all the match criteria to be classified as part of the traffic class.
The CLI displays the class map configuration mode.
`host1/Admin(config)# class-map match-all DNS_INSPECT_L4CLASS`
`host1/Admin(config-cmap)#`
Include one or more of the **match** commands as part of the Layer 3 and Layer 4 class map.
`host1/Admin(config-cmap)# description DNS application protocol inspection of incoming traffic`
`host1/Admin(config-cmap)# match port udp eq domain`
`host1/Admin(config-cmap)# exit`

Table 3-4 Layer 3 and Layer 4 Application Protocol Inspection Quick Start

Task and Command Example

4. Create a Layer 3 and Layer 4 policy map and include the appropriate **inspect** command (**inspect dns**, **inspect ftp**, **inspect http**, **inspect icmp**, **inspect ils**, **inspect rtsp**, **inspect sip**, or **inspect skinny** for SCCP). Specify the actions that you want to apply to the Layer 3 and Layer 4 user-defined class map and, if appropriate, to the default class map.
 	
For example, to specify the **inspect dns** command as an action for a DNS application protocol inspection policy map, enter:
 	
`host1/Admin(config)# policy-map multi-match DNS_INSPECT_L4POLICY`
`host1/Admin(config-pmap)# class DNS_INSPECT_L4CLASS`
`host1/Admin(config-pmap-c)# inspect dns maximum-length 1000`
`host1/Admin(config-pmap-c)# exit`
`host1/Admin(config-pmap)# exit`
`host1/Admin(config)#`

5. Attach the Layer 3 and Layer 4 traffic policy to a single VLAN interface or globally on all VLAN interfaces. For example, to specify a VLAN interface and apply multiple service policies to the VLAN, enter:
 	
`host1/Admin(config)# interface vlan50`
`host1/Admin(config-if)# mtu 1500`
`host1/Admin(config-if)# ip address 192.168.1.100 255.255.0.0`
`host1/Admin(config-if)# service-policy input DNS_INSPECT_L4POLICY`

6. (Optional) Save your configuration changes to flash memory.
 	
`host1/Admin(config)# exit`
`host1/Admin# copy running-config startup-config`

Configuring a Layer 7 FTP Command Inspection Policy

This section describes how to create a Layer 7 class map and policy map that allows the ACE to perform FTP command inspection, which is a security feature that prevents web browsers from sending embedded commands to the ACE in FTP requests. The ACE must acknowledge each FTP command before allowing a new command. FTP inspection allows traffic by default and restricts traffic that fails the security checks. Command filtering allows you to restrict specific commands through the ACE. When the ACE denies a command, it closes the connection.

Note You can associate a maximum of 1024 instances of the same type of regular expression (regex) with a a Layer 4 policy map. This limit applies to all Layer 7 policy-map types, including generic, HTTP, RADIUS, RDP, RTSP, and SIP. You configure regexes in the following:

- Match statements in Layer 7 class maps
- Inline match statements in Layer 7 policy maps
- Layer 7 hash predictors for server farms
- Layer 7 sticky expressions in sticky groups
- Header insertion and rewrite (including SSL URL rewrite) expressions in Layer 7 action lists

This section contains the following topics:

Configuring an FTP Inspection Class Map

This section contains the following topics:

Creating an FTP Inspection Class Map

You can define a class map to be used for the inspection of FTP request commands by using the **class-map type ftp inspect** command in configuration mode.

The syntax of this command is as follows:

class-map type ftp inspect match-any *map_name*

The keywords and arguments are as follows:

- **match-any**—Determines how the ACE inspects FTP request commands when multiple match criteria exist in a class map. Only one of the match criteria listed in the class map is satisfied to match the FTP command inspection class in the class map.
- *map_name*—Name assigned to 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.

The CLI displays the class map FTP command inspection configuration mode. To classify the FTP request commands for inspection by the ACE, include one or more of the **match request-method** commands to configure the match criteria for the Layer 7 class map. See the "Defining FTP Match Request Methods" section.

For example, to specify FTP_INSPECT_L7CLASS as the name of a class map and identify that at least one FTP inspection command in the class map must be satisfied for the ACE to indicate a match, enter:

```
host1/Admin(config)# class-map type ftp inspect match-any
FTP_INSPECT_L7CLASS
host1/Admin(config-cmap-ftp-insp)# match request-method cdup
host1/Admin(config-cmap-ftp-insp)# match request-method mkdir
host1/Admin(config-cmap-ftp-insp)# match request-method get
host1/Admin(config-cmap-ftp-insp)# match request-method put
```

To remove the FTP request inspection class map from the ACE, enter:

```
host1/Admin(config)#no class-map type ftp inspect match-any
FTP_INSPECT_L7CLASS
```

Adding a Layer 7 FTP Inspection Class Map Description

You can use the **description** command to provide a brief summary of the Layer 7 FTP inspection class map.

You must access the class map configuration mode to specify the **description** command.

The syntax of this command is as follows:

description *text*

Use the *text* argument to enter an unquoted text string with a maximum of 240 alphanumeric characters.

To add a description that the class map is to perform FTP command inspection, enter:

```
host1/Admin(config-cmap-ftp-insp)# description FTP command inspection
of incoming traffic
```

To remove the description from the class map, enter:

```
host1/Admin(config-cmap-ftp-insp)# no description FTP command
inspection of incoming traffic
```

Defining FTP Match Request Methods

You can use the **match request-method** command to configure the class map to define FTP command inspection decisions by the ACE. The **match** command identifies the FTP commands that you want filtered by the ACE.

You must access the class map configuration mode to specify the **match request-method** command.

The syntax of this command is as follows:

match request-method *ftp_commands*

The *ftp_commands* argument is the FTP command in the class map to be subjected to FTP inspection by the ACE. The possible *ftp_commands* are **appe**, **cd, cdup**, **dele**, **get, help**, **mkd**, **put**, **rmd**, **rnfr**, **rnto**, **site**, **stou**, and **syst**.

You can specify multiple **match request-methods** commands within a class map.

For example, to specify FTP_INSPECT_L7CLASS as the name of a class map and identify that at least one FTP inspection command in the class map must be satisfied for the ACE to indicate a match, enter:

```
host1/Admin(config)# class-map type ftp inspect match-any
FTP_INSPECT_L7CLASS
host1/Admin(config-cmap-ftp-insp)# match request-method cdup
host1/Admin(config-cmap-ftp-insp)# match request-method mkdir
host1/Admin(config-cmap-ftp-insp)# match request-method get
host1/Admin(config-cmap-ftp-insp)# match request-method stou
host1/Admin(config-cmap-ftp-insp)# match request-method put
```

Use the **no** form of the command to clear the FTP inspection request method from the class map:

```
host1/Admin(config-cmap-ftp-insp)# no match request-method cdup
```

Configuring a Layer 7 FTP Command Inspection Policy Map

This section outlines how to configure a Layer 7 FTP command inspection policy map. The Layer 7 policy map configures the applicable FTP command inspection actions executed on the network traffic that matches the classifications defined in a class map. You then associate the completed Layer 7 FTP command inspection policy with a Layer 3 and Layer 4 policy map to activate the operation on a VLAN interface (see the "Defining Layer 3 and Layer 4 Application Protocol Inspection Policy Actions" section).

This section contains the following topics:

Creating a Layer 7 FTP Command Inspection Policy Map

You can use the **policy-map type inspect ftp** command in configuration mode to name the traffic policy and initiate FTP command inspection.

The syntax of this command is as follows:

policy-map type inspect ftp first-match *map_name*

The keywords and arguments are as follows:

- **ftp first-match**—Specifies a Layer 7 policy map that defines the inspection of FTP commands by the ACE. The **first-match** keyword defines the execution for the Layer 7 FTP command inspection policy map. The ACE executes only the action specified against the first-matching classification.
- *map_name*—Name assigned to the policy map. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to create a Layer 7 FTP command inspection policy map, enter:

```
host/Admin(config)# policy-map type inspect ftp first-match
FTP_INSPECT_L7POLICY
host/Admin(config-pmap-ftp-ins)#
```

The CLI displays the policy map configuration mode.

To remove a Layer 7 command inspection policy map from the ACE, enter:

```
host1/Admin(config)# no policy-map type inspect ftp first-match
FTP_INSPECT_L7POLICY
```

Adding a Layer 7 FTP Inspection Policy Map Description

You can use the **description** command to provide a brief summary of the Layer 7 FTP inspection policy map.

You must access the policy map FTP inspection configuration mode to specify the **description** command.

The syntax of this command is as follows:

description *text*

Use the *text* argument to enter an unquoted text string with a maximum of 240 alphanumeric characters.

To add a description that the policy map is to perform FTP command inspection, enter:

```
host1/Admin(config-pmap-ftp-ins)# description FTP command inspection
of incoming traffic
```

To remove the description from the policy map, enter:

```
host1/Admin(config-pmap-ftp-ins)# no description FTP command
inspection of incoming traffic
```

Including Inline Match Statements in a Layer 7 FTP Command Inspection Policy Map

You can include a single inline match criteria in the policy map without specifying a traffic class by entering an applicable Layer 7 **match** command. The inline Layer 7 policy map **match** commands function in the same way as the Layer 7 class map **match** commands. However, when you use an inline **match** command, you can specify an action for only a single match statement in the Layer 7 policy map.

Note To specify actions for multiple match statements, use a class map as described in the "Associating a Layer 7 FTP Command Inspection Traffic Class with the Traffic Policy" section.

The syntax for this command is as follows:

match *name match_statement* [**insert-before** *map_name*]

The keywords, arguments, and options are as follows:

- *name*—Name assigned to the inline **match** command. Enter an unquoted text string with no spaces. The length of the inline match statement name plus the length of the policy map name with which it is associated cannot exceed a total maximum of 64 alphanumeric characters. For example, if the policy map name is L7_POLICY (nine characters), an inline match statement name under this policy cannot exceed 55 alphanumeric characters (64 - 9 = 55).
- *match_statement*—Inline match criteria to be used by the policy map. See below for details on the **match** commands associated with the Layer 7 FTP command inspection class map.

- **insert-before** *map_name*—(Optional) Places the inline **match** command ahead of an existing class map in the policy map configuration.

The syntax for the Layer 7 FTP inspection policy map inline **match** commands is as follows:

match *name* **request-method** {**appe** | **cd** | **cdup** | **dele** | **get** | **help** | **mkd** | **put** | **rmd** | **rnfr** | **rnto** | **site** | **stou** | **syst**}

See the "Defining FTP Match Request Methods" section for details about the inline **match** command.

For example, to add an inline **match** command to a Layer 7 FTP command policy map, enter:

```
host/Admin(config-pmap-ftp-ins)# match FTP_REQUEST_MATCH
request-method mkdir
host/Admin(config-pmap-ftp-ins-m)#
```

To remove the inline **match** command from the Layer 7 FTP command policy map, enter:

```
host/Admin(config-pmap-ftp-ins)# no match FTP_REQUEST_MATCH
```

Associating a Layer 7 FTP Command Inspection Traffic Class with the Traffic Policy

You can associate a traffic class created with the **class-map** command to associate network traffic with the traffic policy by using the **class** command.

The syntax of this command is as follows:

class *map_name*

The *map_name* argument is the name of a previously defined traffic class, configured with the **class-map** command, to associate traffic to the traffic policy. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

The CLI displays the policy map class configuration mode.

For example, to specify an existing class map in the Layer 7 policy map, enter:

```
host1/Admin(config-pmap-ftp-ins)# class FTP_INSPECT_L7CLASS
host1/Admin(config-pmap-ftp-ins-c)#
```

To remove a class map from a Layer 7 policy map, enter:

```
host1/Admin(config-pmap-ftp-ins)# no class FTP_INSPECT_L7CLASS
```

Specifying the Layer 7 FTP Command Inspection Policy Actions

By default, the ACE allows all FTP commands to pass. To explicitly deny specific FTP commands, use one of the following commands as the action if the specified FTP traffic matches the classification. You apply the specified action against the single inline **match** command or the specified class map.

{**deny** | **mask-reply**}

The keywords are as follows:

- **deny**—Denies the FTP request commands against the single inline **match** command or specified in the class map by resetting the FTP session.
- **mask-reply**—Applies only to the FTP SYST command and its associated reply. The SYST command is used to find out the type of operating system at the FTP server. The **mask-reply** keyword instructs the ACE to mask the system's reply to the FTP SYST command by filtering sensitive information from the command output.

For example, to specify the actions in the Layer 7 FTP inspection policy map, enter:

```
host1/Admin(config)# policy-map type inspect ftp first-match
FTP_INSPECT_L7POLICY
host1/Admin(config-pmap-ftp-ins)# class FTP_INSPECT_L7CLASS
host1/Admin(config-pmap-ftp-ins-c)# mask-reply
```

To disable an action from the Layer 7 FTP inspection policy map, enter:

```
host1/Admin(config-pmap-ftp-ins-c)# no mask-reply
```

Configuring a Layer 7 HTTP Deep Inspection Policy

This section describes how to create a Layer 7 class map and policy map to be used for HTTP deep packet inspection by the ACE. The ACE performs a stateful deep packet inspection of the HTTP protocol and permits or restricts traffic based on the actions in your configured policy maps. The following security features are included as part of HTTP deep packet inspection as performed by the ACE:

- Regular expression matching on name in an HTTP header, URL name, or content expressions in an HTTP entity-body
- Content, URL, and HTTP header length checks
- MIME-type message inspection
- Transfer-encoding methods
- Content type verification and filtering
- Port 80 misuse by tunneling protocols
- RFC compliance monitoring and RFC method filtering

Note You can associate a maximum of 1024 instances of the same type of regular expression (regex) with a a Layer 4 policy map. This limit applies to all Layer 7 policy-map types, including generic, HTTP, RADIUS, RDP, RTSP, and SIP. You configure regexes in the following:

- Match statements in Layer 7 class maps
- Inline match statements in Layer 7 policy maps
- Layer 7 hash predictors for server farms
- Layer 7 sticky expressions in sticky groups
- Header insertion and rewrite (including SSL URL rewrite) expressions in Layer 7 action lists

This section contains the following topics:

Configuring a Layer 7 HTTP Deep Inspection Class Map

This section contains the following topics:

Creating an HTTP Deep Inspection Class Map

You can create a Layer 7 class map for deep packet inspection of HTTP traffic by using the **class-map type http inspect** command in configuration mode.

The syntax of this command is as follows:

class-map type http inspect [**match-all** | **match-any**] *map_name*

The keywords, arguments, and options are as follows:

- **match-all** | **match-any**—(Optional) Determines how the ACE performs the deep packet inspection of HTTP 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) Network traffic needs to satisfy all of the match criteria (implicit AND) to match the Layer 7 HTTP deep packet inspection class map. The **match-all** keyword is applicable only for match statements of different HTTP deep packet inspection types. For example, specifying a **match-all** condition for URL, HTTP header, and URL content statements in the same class map is valid. However, specifying a **match-all** condition for multiple HTTP headers with the same names or multiple URLs in the same class map is invalid.
  - **match-any**—Network traffic needs to satisfy only one of the match criteria (implicit OR) to match the Layer 7 HTTP deep packet inspection class map. The **match-any** keyword is applicable for match statements of different Layer 7 HTTP deep packet inspection type or multiple instances of the same type with different names. For example, the ACE allows you to specify a **match-any** condition for cookie, HTTP header, and URL content statements in the same class map, but it does not allow you to specify a **match-any** condition for URL length, HTTP header length, and content length statements in the same class map.

- *map_name*—Name assigned to the class map. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

The CLI displays the class map HTTP application protocol inspection configuration mode. To classify the HTTP application inspection of traffic for evaluation by the ACE, include one or more of the following commands to configure the match criteria for the Layer 7 class map:

- **match content**—See the "Defining HTTP Content Match Criteria" section.
- **match content length**—See the "Defining the Length of the HTTP Content for Inspection" section.
- **match cookie secondary**—See the "Defining a Secondary Cookie for HTTP Inspection" section.
- **match header**—See the "Defining an HTTP Header for Inspection" section.
- **match header length**—See the "Defining the HTTP Maximum Header Length for Inspection" section.
- **match header mime-type**—See the "Defining a Header MIME-Type Messages for Inspection" section.
- **match port-misuse**—See the "Defining an HTTP Traffic Restricted Category" section.
- **match request-method**—See the "Defining HTTP Request Methods and Extension Methods" section.

- **match transfer-encoding**—See the "Defining an HTTP Transfer Encoding Type"
- **match url**—See the "Defining an HTTP URL for Inspection" section.
- **match url length**—See the "Defining an HTTP Maximum URL Length for Inspection" section.

You may include multiple **match** commands in the class map.

For example, to specify HTTP_INSPECT_L7CLASS as the name of a class map and identify that at least one command in the Layer 7 HTTP application inspection class map must be satisfied for the ACE to indicate a match, enter:

```
host1/Admin(config)# class-map type http inspect match-any
HTTP_INSPECT_L7CLASS
host1/Admin(config-cmap-http-insp)# match header length request eq 200
host1/Admin(config-cmap-http-insp)# match header Host header-value
.*mycompanyexample.com
host1/Admin(config-cmap-http-insp)# match url length eq 10000
host1/Admin(config-cmap-http-insp)# match url .*.gif
```

To remove the HTTP application inspection class map from the ACE, enter:

```
host1/Admin(config)#no class-map type http inspect match-any
HTTP_INSPECT_L7CLASS
```

Adding a Layer 7 HTTP Deep Packet Inspection Class Map Description

You can use the **description** command to provide a brief description of the Layer 7 HTTP deep packet inspection class map.

You must access the class map configuration mode to specify the **description** command.

The syntax of this command is as follows:

description *text*

Use the *text* argument to enter an unquoted text string with a maximum of 240 alphanumeric characters.

To add a description that the class map is to perform HTTP deep packet inspection, enter:

```
host1/Admin(config-cmap-http-insp)# description HTTP protocol deep
inspection of incoming traffic
```

To remove the description from the class map, enter:

```
host1/Admin(config-cmap-http-insp)# no description
```

Defining HTTP Content Match Criteria

You can use the **match content** command to configure the class map to define HTTP application inspection decisions based on content expressions contained within the HTTP entity-body.

You must access the class map configuration mode to specify the **match content** command.

The syntax of this command is as follows:

[*line_number*] **match content** *expression* [**offset** *number*]

The keywords, arguments, and options are as follows:

- *line_number*—(Optional) Argument that assists you in editing or deleting individual **match** commands. Enter an integer from 2 to 1024 as the line number. You can enter **no** *line_number* to delete long **match** commands instead of entering the entire line. The line numbers do not dictate a priority or sequence for the match statements.
- *expression*—Content expression contained within the HTTP entity-body. The range is from 1 to 255 alphanumeric characters. See Table 3-5 for a list of the supported characters that you can use in regular expressions.

Note When matching data strings, note that the period (.) and question mark (?) characters do not have a literal meaning in regular expressions. Use brackets ([]) to match these symbols (for example, enter www[.]xyz[.]com instead of www.xyz.com). You can also use a backslash (\) to escape a dot (.) or a question mark (?).

- **offset** *number*—Provides an absolute offset where the content expression search string starts. The offset starts at the first byte of the message body, after the empty line (CR,LF,CR,LF) between the headers and the body of the message. The offset value is between 1 to 4000 bytes.

For example, to create a class map that specifies a content expression contained within the entity-body sent with an HTTP request, enter:

```
host1/Admin(config)#class-map type http inspect match-any
HTTP_INSPECT_L7CLASS
host1/Admin(config-cmap-http-insp)# match content .*newp2psig
```

To clear the content expression checking match criteria from the class map, enter:

```
host1/Admin(config-cmap)# no match content .*newp2psig
```

Defining the Length of the HTTP Content for Inspection

You can use the **match content length** command to configure the class map to define application inspection decisions on HTTP traffic up to the configured maximum content parse length. Messages that meet the specified criteria will be either allowed or denied based on the Layer 7 HTTP deep packet inspection policy map action.

You must access the class map configuration mode to specify the **match content length** command.

The syntax of this command is as follows:

[*line_number*] **match content length** {**eq** *bytes* | **gt** *bytes* | **lt** *bytes* | **range** *bytes1 bytes 2*}

The keywords, arguments, and options are as follows:

- *line_number*—(Optional) Argument that assists you in editing or deleting individual **match** commands. Enter an integer from 2 to 1024 as the line number. You can enter **no** *line_number* to delete long **match** commands instead of entering the entire line. The line numbers do not dictate a priority or sequence for the match statements.
- **eq** *bytes*—Specifies a value for the content parse length in an HTTP message received by the ACE. Based on the policy map action, the ACE allows or denies messages with a content length equal to the specified value. Valid entries are from 1 to 65535 bytes.

- **gt** *bytes*—Specifies a minimum value for the content parse length in an HTTP message received by the ACE. Based on the policy map action, the ACE allows or denies messages with a content length greater than the specified value. Valid entries are from 1 to 65535 bytes.
- **lt** *bytes*—Specifies a maximum value for the content parse length in an HTTP message received by the ACE. Based on the policy map action, the ACE allows or denies messages with a content length size less than the specified value. Valid entries are from 1 to 65535 bytes.
- **range** *bytes1 bytes2*—Specifies a size range for the content parse length in an HTTP message received by the ACE. Based on the policy map action, the ACE allows or denies messages with a content length within this range. The range is from 1 to 65535 bytes.

For example, to create a class map that identifies the content length in an HTTP message that can be received by the ACE, enter:

```
host1/Admin(config)# class-map type http inspect match-any
HTTP_INSPECT_L7CLASS
host1/Admin(config-cmap-http-insp)# match content length eq 3495
```

To clear the HTTP content length match criteria from the class map, enter:

```
host1/Admin(config-cmap-http-insp)# no match content length eq 3495
```

Defining a Secondary Cookie for HTTP Inspection

You can use the **match cookie secondary** command in class map HTTP inspection configuration mode to configure a class map to define inspection decisions based on the name or prefix and value of a secondary cookie (URL query string). Normally, the ACE parses URLs up to, but not including, the question mark (?) in a URL string. This feature extends the URL parsing capabilities of the ACE to include the URL parameters beyond the question mark. The ACE also uses this command to match secondary cookies present in the HTTP content of POST requests. This command is available as either a match statement in a class map or an inline match statement (slightly different syntax) in a Layer 7 policy map. For details about inline match statements, see the "Including Inline Match Statements in a Layer 7 HTTP Deep Packet Inspection Policy Map" section.

The syntax of this command is as follows:

match cookie secondary [**name** *cookie_name* | **prefix** *prefix_name*] **value** *expression*

The keywords, options, and arguments are as follows:

- **name** *cookie_name*—Specifies the identifier of the secondary cookie to match. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.
- **prefix** *prefix_name*—(Optional) Specifies the prefix of the secondary cookie to match. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.
- **value** *expression*—(Optional) Specifies the regular expression of the secondary cookie to match. Enter an unquoted text string with no spaces and a maximum of 255 alphanumeric characters. See Table 3-5 for a list of the supported characters that you can use in regular expressions.

Note When matching data strings, note that the period (.) and question mark (?) characters do not have a literal meaning in regular expressions. Use brackets ([]) to match these symbols (for example, enter www[.]xyz[.]com instead of www.xyz.com). You can also use a backslash (\) to escape a dot (.) or a question mark (?).

Table 3-5 Special Characters for Matching String Expressions

Convention	Description
.	One of any character.
.*	Zero or more of any character.
\.	Period (escaped).
[charset]	Match any single character from the range.
[^charset]	Do not match any character in the range. All other characters represent themselves.
()	Expression grouping.
(expr1 | expr2)	OR of expressions.
(expr)*	0 or more of expression.
(expr)+	1 or more of expression.
expr{m,n}	Repeat the expression between *m* and *n* times, where *m* and *n* have a range from 1 to 255.

Table 3-5 Special Characters for Matching String Expressions (continued)

Convention	Description
expr{m}	Match the expression exactly *m* times. The range for *m* is from 1 to 255.
expr{m,}	Match the expression *m* or more times. The range for *m* is from 1 to 255.
\a	Alert (ASCII 7).
\b	Backspace (ASCII 8).
\f	Form-feed (ASCII 12).
\n	New line (ascii 10).
\r	Carriage return (ASCII 13).
\t	Tab (ASCII 9).
\v	Vertical tab (ASCII 11).
\0	Null (ASCII 0).
\\	Backslash.
\x##	Any ASCII character as specified in two-digit hexadecimal notation.

The following configuration guidelines apply when you configure a secondary cookie match statement for HTTP inspection:

- Ensure that secondary cookie names do not overlap with other secondary cookie names in the same match-all class map. For example, the following configuration is not allowed because the two match statements have overlapping cookie names:

```
host1/Admin(config)# class-map type http inspect match-any
HTTP_INSPECT_L7CLASS
host1/Admin(config-cmap-http-insp)# match cookie secondary prefix
id value .*
host1/Admin(config-cmap-http-insp)# match cookie secondary name
identity value bob
```

- When you configure a secondary cookie value match across all secondary cookie names in a match-all class map, you cannot configure any other secondary cookie match in the same class map because a secondary cookie match on a value alone is equivalent to a wildcard match on a name. In the following example, the second match statement is not allowed:

```
host1/Admin(config)# class-map type http inspect match-any
HTTP_INSPECT_L7CLASS
host1/Admin(config-cmap-http-insp)# match cookie secondary value
bob
host1/Admin(config-cmap-http-insp-m)# exit
host1/Admin(config-cmap-http-insp)# match cookie secondary name
identity value jane
```

For example, to match a secondary cookie called “matchme” with a regular expression value of .*abc123, enter:

```
host1/Admin(config)# class-map type http inspect match-any
HTTP_INSPECT_L7CLASS
host1/Admin(config-cmap-http-insp)# match cookie secondary name
matchme value .*abc123
```

For example, to match all cookie names starting with the letters “ab”, enter:

```
host1/Admin(config)# class-map type http inspect match-any
HTTP_INSPECT_L7CLASS
host1/Admin(config-cmap-http-insp)# match cookie secondary prefix ab
value .*
```

For example, to match a given regex in all secondary cookie values, enter:

```
host1/Admin(config)# class-map type http inspect match-any
HTTP_INSPECT_L7CLASS
host1/Admin(config-cmap-http-insp)# match cookie secondary value
.*machine-key
```

To remove a secondary cookie match statement from a class map, enter the **no** form of the command as follows:

```
host1/Admin(config)# class-map type http inspect match-any
HTTP_INSPECT_L7CLASS
host1/Admin(config-cmap-http-insp)# no match cookie secondary value
.*machine-key
```

Defining an HTTP Header for Inspection

You can use the **match header** command to configure the class map to define application inspection decisions based on the name and value in an HTTP header. The ACE performs regular expression matching against the received packet data from a particular connection based on the HTTP header expression.

You must access the class map configuration mode to specify the **match header** command.

The syntax of this command is as follows:

[*line_number*] **match header** {*header_name* | *header_field*} **header-value** *expression*

The keywords, arguments, and options are as follows:

- *line_number*—(Optional) Argument that assists you in editing or deleting individual **match** commands. Enter an integer from 2 to 1024 as the line number. You can enter **no** *line_number* to delete long **match** commands instead of entering the entire line. The line numbers do not dictate a priority or sequence for the match statements.
- *header_name*—Name of the HTTP header to match (for example, www.example1.com.) Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters. Alternatively, you can enter a text string with spaces if you enclose the entire string in quotation marks (“). For a list of predefined header fields, see Table 3-6.

Note The *header_name* argument cannot include the colon in the name of the HTTP header; the ACE rejects the colon as an invalid token.

- *header_field*—Standard HTTP/1.1 header field. Valid selections include request-header fields, general-header fields, and entity-header field. Table 3-6 lists the supported HTTP/1.1 header fields.

Table 3-6 HTTP/1.1 Header Fields

Field Name	Description
Accept	Semicolon-separated list of representation schemes (content type metainformation values) that will be accepted in the response to the request.
Accept-Charset	Character sets that are acceptable for the response. This field allows clients capable of understanding more comprehensive or special-purpose character sets to signal that capability to a server that can representing documents in those character sets.
Accept-Encoding	Restricts the content encoding that a user will accept from the server.
Accept-Language	ISO code for the language in which the document is written. The language code is an ISO 3316 language code with an optional ISO639 country code to specify a national variant.
Authorization	Specifies that the user agent wants to authenticate itself with a server, usually after receiving a 401 response.
Cache-Control	Directives that must be obeyed by all caching mechanisms in the request/response chain. The directives specify behavior intended to prevent caches from adversely interfering with the request or response.
Connection	Connection options that the sender can specify.
Content-MD5	MD5 digest of the entity-body that provides an end-to-end integrity check. Only a client or an origin server can generate this header field.
Expect	Used by a client to inform the server about the behaviors that the client requires.
From	E-mail address of the person that controls the requesting user agent.

Table 3-6 HTTP/1.1 Header Fields (continued)

Field Name	Description
Host	Internet host and port number of the resource being requested, as obtained from the original URI given by the user or referring resource. The Host field value must represent the naming authority of the origin server or gateway given by the original URL.
If-Match	Used with a method to make it conditional. A client that has one or more entities previously obtained from the resource can verify that one of those entities is current by including a list of their associated entity tags in the If-Match header field. This feature allows efficient updates of cached information with a minimum amount of transaction overhead. It is also used, on updating requests, to prevent inadvertent modification of the wrong version of a resource. As a special case, the value "*" matches any current entity of the resource.
Pragma	Pragma directives understood by servers to whom the directives are relevant. The syntax is the same as for other multiple-value fields in HTTP. For example, the **accept** field is a comma-separated list of entries for which the optional parameters are separated by semicolons.
Referer	Address (URI) of the resource from which the URI in the request was obtained.
Transfer-Encoding	What (if any) type of transformation has been applied to the message body in order to safely transfer it between the sender and the recipient.

Table 3-6 HTTP/1.1 Header Fields (continued)

Field Name	Description
User-Agent	Information about the user agent such as a software program that originates the request. This information is for statistical purposes, the tracing of protocol violations, and automated recognition of user agents so that you can customize responses to avoid particular user agent limitations.
Via	Used by gateways and proxies to indicate the intermediate protocols and recipients between the user agent and the server on requests, and between the origin server and the client on responses.

- **header-value** *expression*—Specifies the header value expression string to compare against the value in the specified field in the HTTP header. The range is from 1 to 255 alphanumeric characters. There are predefined header fields, such as Accept-Language, User-Agent, or Host. The ACE supports the use of regular expressions for matching. Expressions are stored in a header map in the form *header-name: expression*. Header expressions allow spaces, if the spaces are escaped or quoted. See Table 3-5 for a list of the supported characters that you can use in regular expressions.

Note When matching data strings, note that the period (.) and question mark (?) characters do not have a literal meaning in regular expressions. Use brackets ([]) to match these symbols (for example, enter www[.]xyz[.]com instead of www.xyz.com). You can also use a backslash (\) to escape a dot (.) or a question mark (?).

For example, to specify that the Layer 7 class map is to match and perform application inspection on HTTP headers, enter:

```
host1/Admin(config)# class-map type http inspect HTTP_INSPECT_L7CLASS
host1/Admin(config-cmap)# match header Host header-value
.mycompanyexample.com
```

For example, to specify regular expressions in a class map to emulate a wildcard search to match the header value expression string, enter:

```
host1/Admin(config)# class-map type http inspect match-any
HTTP_INSPECT_L7CLASS
host1/Admin(config-cmap-http-insp)# match header Host header-value
.*myfirstcompanyexample.com
host1/Admin(config-cmap-http-insp)# match header Host header-value
.*mysecondcompanyexample.com
```

To clear an HTTP header match criteria from the class map, enter:

```
host1/Admin(config-cmap-http-insp)# no match header Host header-value
.*mysecondcompanyexample.com
```

Defining the HTTP Maximum Header Length for Inspection

By default, the maximum header length for HTTP deep packet inspection is 2048 bytes. Use the **match header length** command to limit the HTTP traffic allowed through the ACE based on the length of the entity-body in the HTTP message. Messages are either allowed or denied based on the Layer 7 HTTP deep packet inspection policy map action.

You must access the class map configuration mode to specify the **match header length** command.

The syntax of this command is as follows:

[*line_number*] **match header length** {**request** | **response**} {**eq** *bytes* | **gt** *bytes* | **lt** *bytes* | **range** *bytes1 bytes 2*}

The keywords, arguments, and options are as follows:

- *line_number*—(Optional) Argument that assists you in editing or deleting individual **match** commands. Enter an integer from 2 to 1024 as the line number. You can enter **no** *line_number* to delete long **match** commands instead of entering the entire line. The line numbers do not dictate a priority or sequence for the match statements.
- **request**—Specifies the size of the HTTP header request message that can be received by the ACE.
- **response**—Specifies the size of the HTTP header response message sent by the ACE.

- **eq** *bytes*—Specifies a value for the entity-body in an HTTP message received by the ACE. Based on the policy map action, the ACE allows or denies messages with an entity-body size equal to the specified value. Valid entries are from 1 to 65535 bytes.
- **gt** *bytes*—Specifies a minimum value for the entity-body in an HTTP message received by the ACE. Based on the policy map action, the ACE allows or denies messages with an entity-body size greater than the specified value. Valid entries are from 1 to 65535 bytes.
- **lt** *bytes*—Specifies a maximum value for the entity-body in an HTTP message received by the ACE. Based on the policy map action, the ACE allows or denies messages with an entity-body size less than the specified value. Valid entries are from 1 to 65535 bytes.
- **range** *bytes1 bytes2*—Specifies a size range for the entity-body in an HTTP message received by the ACE. Based on the policy map action, the ACE allows or denies messages with a entity-body size within this range. The range is from 1 to 65535 bytes.

For example, to specify that the class map is to match on HTTP traffic received with a length less than or equal to 3600 bytes in the entity-body of the HTTP message, enter:

```
host1/Admin(config)# class-map type http inspect HTTP_INSPECT_L7CLASS
host1/Admin(config-cmap-http-insp)# match header length request eq
3600
```

To clear the maximum HTTP header length match criteria from the class map, enter:

```
host1/Admin(config-cmap-http-insp)# no match header length request eq
3600
```

Defining a Header MIME-Type Messages for Inspection

You can use the **match header mime-type** command to specify a subset of the Multipurpose Internet Mail Extension (MIME)-type messages that the ACE permits or denies based on the actions in the policy map. MIME-type validation extends the format of Internet mail to allow non-US-ASCII textual messages, nontextual messages, multipart message bodies, and non-US-ASCII information in message headers.

Note To define MIME-type messages in addition to what is supported under the **match header mime-type** command, use the **match header** command. For example, to define a match for a new MIME-type audio\myaudio, you could enter the following match statement: **match header Content-type header-value audio/myaudio**. See the "Defining an HTTP Header for Inspection" section for details.

The syntax of this command is as follows:

[*line_number*] **match header mime-type** *mime_type*

The keywords, arguments, and options are as follows:

- *line_number*—(Optional) Argument that assists you in editing or deleting individual **match** commands. Enter an integer from 2 to 1024 as the line number. You can enter **no** *line_number* to delete long **match** commands instead of entering the entire line. The line numbers do not dictate a priority or sequence for the match statements.
- *mime_type*—Predefined list of mime-types, such as image\Jpeg, text\html, application/msword, and audio/mpeg. Choose whether only the mime-types included in this list are permitted through the ACE or whether all mime-types are acceptable. The default behavior is to allow all mime-types.

  The supported mime-types are as follows:

  - application/msexcel
  - application/mspowerpoint
  - application/msword
  - application/octet-stream
  - application/pdf
  - application/postscript
  - application/x-gzip
  - application/x-java-archive
  - application/x-java-vm
  - application/x-messenger
  - application/zip
  - audio/*

- audio/basic
- audio/midi
- audio/mpeg
- audio/x-adpcm
- audio/x-aiff
- audio/x-ogg
- audio/x-wav
- image/*
- image/gif
- image/jpeg
- image/png
- image/tiff
- image/x-3ds
- image/x-bitmap
- image/x-niff
- image/x-portable-bitmap
- image/x-portable-greymap
- image/x-xpm
- text/*
- text/css
- text/html
- text/plain
- text/richtext
- text/sgml
- text/xmcd
- text/xml
- video/*
- video/flc
- video/mpeg

- video/quicktime
- video/sgi
- video/x-fli

Follow these guidelines when using the **match header mime-type** command:

- You can specify multiple **match header mime-type** commands within a class map.
- Each **match header mime-type** command configures a single application type.

For example, to create a class map that specifies the MIME-type audio/midi and audio/mpeg messages permitted through the ACE, enter:

```
host1/Admin(config)# class-map type http inspect match-any
HTTP_INSPECT_L7CLASS
host1/Admin(config-cmap-http-insp)# match header mime-type audio/midi
host1/Admin(config-cmap-http-insp)# match header mime-type audio/mpeg
```

To deselect the specified MIME message match criteria from the class map, enter:

```
host1/Admin(config-cmap-http-insp)# no match header mime-type
audio/midi
```

Defining an HTTP Traffic Restricted Category

You can use the **match port-misuse** command to configure the class map to define application inspection compliance decisions that restrict certain HTTP traffic from passing through the ACE. This class map detects the misuse of port 80 (or any other port running HTTP) for tunneling protocols such as peer-to-peer (p2p) applications, tunneling applications, and instant messaging.

You must access the class map configuration mode to specify the **match port-misuse** command.

The syntax of this command is as follows:

[*line_number*] **match port-misuse** *application_category*

The keywords, arguments, and options are as follows:

- *line_number*—(Optional) Argument that assists you in editing or deleting individual **match** commands. Enter an integer from 2 to 1024 as the line number. You can enter **no** *line_number* to delete long **match** commands instead of entering the entire line. The line numbers do not dictate a priority or sequence for the match statements.
- *application_categor*y—Restricted HTTP application category for the class map. The possible values for *application_category* are as follows:
  - **im**—Instant messaging application category. The ACE checks for the Yahoo Messenger instant messaging application.
  - **p2p**—Peer-to-peer application category. The applications checked include Kazaa and Gnutella.
  - **tunneling**—Tunneling application category. The applications checked include: HTTPort/HTTHost, GNU Httptunnel, and Firethru.

Follow these guidelines when using the **match port-misuse** command:

- You can specify multiple **match port-misuse** commands within a class map.
- Each **match port-misuse** command configures a single application type.
- The port misuse application inspection process requires a search of the entity-body of the HTTP message, which may degrade performance of the ACE.
- The ACE disables the **match port-misuse** command by default. If you do not configure a restricted HTTP application category, the default action by the ACE is to allow the applications without generating a log.

For example, to create a class map that identifies peer-to-peer applications as restricted HTTP traffic, enter:

```
host1/Admin(config)# class-map type http inspect HTTP_INSPECT_L7CLASS
host1/Admin(config-cmap-http-insp)# match port-misuse p2p
```

To clear the HTTP restricted application category match criteria from the class map, enter:

```
host1/Admin(config-cmap-http-insp)# no match port-misuse p2p
```

Defining HTTP Request Methods and Extension Methods

By default, the ACE allows all request and extension methods. You can use the **match request-method** command to configure the class map to define application inspection compliance decisions based on the request methods defined in RFC 2616 and by HTTP extension methods. If the HTTP request method or extension method compliance checks fails, the ACE denies or resets the specified HTTP traffic based on the policy map action.

You must access the class map HTTP inspection configuration mode to specify the **match request-method** command.

The syntax of this command is as follows:

[*line_number*] **match request-method** {**ext** *method* | **rfc** *method*}

The keywords, arguments, and options are as follows:

- *line_number*—(Optional) Argument that assists you in editing or deleting individual **match** commands. Enter an integer from 2 to 1024 as the line number. You can enter **no** *line_number* to delete long **match** commands instead of entering the entire line. The line numbers do not dictate a priority or sequence for the match statements.
- **ext** *method*—Specifies an HTTP extension method. If the RFC request messages does not contain one of the RFC 2616 HTTP request methods, the ACE verifies if it is an extension method. The ACE supports the inspection of the following HTTP request extension methods: **copy**, **edit**, **getattr**, **getattrname**, **getprops**, **index**, **lock**, **mkcol**, **mkdir**, **move**, **propfind**, **proppatch**, **revadd**, **revlabel**, **revlog**, **revnum**, **save**, **setattr**, **startrev**, **stoprev**, **unedit**, and **unlock**.
- **rfc** *method*—Specifies an RFC 2616 HTTP request method that you want to perform an RFC compliance check on. The ACE supports the inspection of the following RFC 2616 HTTP request methods: **connect**, **delete**, **get**, **head**, **options**, **post**, **put**, and **trace**.

Follow these guidelines when using the **match request-method** command:

- You can specify multiple **match request-method** commands within a class map.
- Each **match request-method** command configures a single request method.
- For unsupported HTTP request methods, include the **inspect http strict** command as an action in the Layer 3 and Layer 4 policy map.

- The ACE disables the **match request-method** command by default. If you do not configure a request method, the default action by the ACE is to allow the RFC 2616 HTTP request method without generating a log.

For example, to create a class map that identifies the **connect**, **get**, **head**, and **index** HTTP RFC 2616 protocols for HTTP application protocol inspection, enter:

```
host1/Admin(config)# class-map type http inspect match-any
HTTP_INSPECT_L7CLASS
host1/Admin(config-cmap-http-insp)# match request-method rfc connect
host1/Admin(config-cmap-http-insp)# match request-method rfc get
host1/Admin(config-cmap-http-insp)# match request-method rfc head
host1/Admin(config-cmap-http-insp)# match request-method ext index
```

To clear an RFC 2616 HTTP request method match criteria from the class map, enter:

```
host1/Admin(config-cmap-http-insp)# no match request-method rfc
connect
```

Defining an HTTP Transfer Encoding Type

You can use the **match transfer-encoding** command to configure the class map to define application inspection decisions that limit the HTTP transfer-encoding types that can pass through the ACE. The transfer-encoding general-header field indicates the type of transformation, if any, that has been applied to the HTTP message body to safely transfer it between the sender and the recipient. When an HTTP request message contains the configured transfer-encoding type, the ACE performs the configured action in the policy map.

You must access the class map HTTP inspection configuration mode to specify the **match transfer-encoding** command.

The syntax of this command is as follows:

[*line_number*] **match transfer-encoding** *coding_types*

The keywords, arguments, and options are as follows:

- *line_number*—(Optional) Argument that assists you in editing or deleting individual **match** commands. Enter an integer from 2 to 1024 as the line number. You can enter **no** *line_number* to delete long **match** commands instead of entering the entire line. The line numbers do not dictate a priority or sequence for the match statements.

- *coding_types*—HTTP transfer-encoding type for the class map. The possible values for *coding_types* are as follows:
  - **chunked**—Message body is transferred as a series of chunks.
  - **compress**—Encoding format produced by the common UNIX file compression program "compress." This format is an adaptive Lempel-Ziv-Welch coding (LZW).
  - **deflate**—The .zlib format defined in RFC 1950 with the deflate compression mechanism described in RFC 1951.
  - **gzip**—Encoding format produced by the file compression program gzip (GNU zip) as described in RFC 1952. This format is a Lempel-Ziv coding (LZ77) with a 32-bit CRC.
  - **identity**—Default (identity) encoding, which does not require the use of transformation.

Follow these guidelines when using the **match transfer-encoding** command:

- You can specify multiple **match transfer-encoding** commands within a class map.
- Each **match transfer-encoding** command configures a single application type.
- The ACE disables the **match transfer-encoding** command by default.

For example, to create a class map that specifies a chunked HTTP transfer encoding type to limit the HTTP traffic that flows through the ACE, enter:

```
host1/Admin(config)# class-map type http inspect HTTP_INSPECT_L7CLASS
host1/Admin(config-cmap-http-insp)# match transfer-encoding chunked
```

To clear the HTTP transfer-encoding match criteria from the class map, enter:

```
host1/Admin(config-cmap-http-insp)# no match transfer-encoding chunked
```

Defining an HTTP URL for Inspection

You can use the **match url** command to configure the class map to define application inspection decisions based on the URL name. HTTP performs regular expression matching against the received packet data from a particular connection based on the URL expression.

You must access the class map configuration mode to specify the **match url** command.

The syntax of this command is as follows:

[*line_number*] **match url** *expression*

The keywords, arguments, and options are as follows:

- *line_number*—(Optional) Argument that assists you in editing or deleting individual **match** commands. Enter an integer from 2 to 1024 as the line number. You can enter **no** *line_number* to delete long **match** commands instead of entering the entire line. The line numbers do not dictate a priority or sequence for the match statements.
- *expression*—URL, or portion of a URL, to match. The URL string range is from 1 to 255 characters. Include only the portion of the URL that follows www.hostname.domain in the match statement. For example, in the URL www.anydomain.com/latest/whatsnew.html, include only /latest/whatsnew.html. To match the www.anydomain.com portion, the URL string can take the form of a URL regular expression. The ACE supports the use of regular expressions for matching. See Table 3-5 for a list of the supported characters that you can use in regular expressions.

Note When matching URLs, the period (.) does not have a literal meaning in regular expressions. Use either brackets ([]) or the backslash (\) character to match this symbol; for example, specify www[.]xyz[.]com instead of www.xyz.com.

For example, to specify that the Layer 7 class map is to match and perform application inspection on a specific URL, enter:

```
host1/Admin(config)# class-map type http inspect HTTP_INSPECT_L7CLASS
host1/Admin(config-cmap-http-insp)# match url whatsnew/latest.*
```

For example, to use regular expressions to emulate a wildcard search to match on any .gif or .html file, enter:

```
host1/Admin(config)# class-map type http inspect match-any
HTTP_INSPECT_L7CLASS
host1/Admin(config-cmap-http-insp)# match url .*.gif
host1/Admin(config-cmap-http-insp)# match url .*.html
```

To clear a URL match criteria from the class map, enter:

```
host1/Admin(config-cmap-http-insp)# no match url .*.gif
```

Defining an HTTP Maximum URL Length for Inspection

You can use the **match url length** command to limit the HTTP traffic allowed through the ACE by specifying the maximum length of a URL in a request message that can be received by the ACE. Messages will be either allowed or denied based on the Layer 7 HTTP deep packet inspection policy map action.

You must access the class map configuration mode to specify the **match url length** command.

The syntax of this command is as follows:

[*line_number*] **match url length** {**eq** *bytes* | **gt** *bytes* | **lt** *bytes* | **range** *bytes1 bytes 2*}

The keywords, arguments, and options are as follows:

- *line_number*—(Optional) Argument that assists you in editing or deleting individual **match** commands. Enter an integer from 2 to 1024 as the line number. You can enter **no** *line_number* to delete long **match** commands instead of entering the entire line. The line numbers do not dictate a priority or sequence for the match statements.
- **eq** *bytes*—Specifies a value for the HTTP URL length received by the ACE. Based on the policy map action, the ACE allows or denies messages with an HTTP URL length equal to the specified value. Valid entries are from 1 to 65535 bytes.
- **gt** *bytes*—Specifies a minimum value for the HTTP URL length received by the ACE. Based on the policy map action, the ACE allows or denies messages with an HTTP URL length greater than the specified value. Valid entries are from 1 to 65535 bytes.
- **lt** *bytes*—Specifies a maximum value for the HTTP URL length received by the ACE. Based on the policy map action, the ACE allows or denies messages with an HTTP URL length less than the specified value. Valid entries are from 1 to 65535 bytes.
- **range** *bytes1 bytes2*—Specifies a size range for the HTTP URL length received by the ACE. Based on the policy map action, the ACE allows or denies messages with a URL length within this range. The range is from 1 to 65535 bytes.

For example, to specify that a class map is to match on a URL with a length equal to 10000 bytes in the request message, enter:

```
host1/Admin(config)# class-map type http inspect HTTP_INSPECT_L7CLASS
host1/Admin(config-cmap-http-insp)# match url length eq 10000
```

To clear a URL length match criteria from the class map, enter:

```
host1/Admin(config-cmap-http-insp)# no match url length eq 10000
```

Configuring a Layer 7 HTTP Deep Packet Inspection Policy Map

This section describes how to configure a Layer 7 HTTP deep inspection policy map. The Layer 7 policy map configures the applicable HTTP deep packet inspection actions executed on the network traffic that match the classifications defined in a class map. You then associate the completed Layer 7 HTTP deep packet inspection policy with a Layer 3 and Layer 4 policy map to activate the operation on a VLAN interface (see the "Defining Layer 3 and Layer 4 Application Protocol Inspection Policy Actions" section).

This section contains the following topics:

Creating a Layer 7 HTTP Deep Packet Inspection Policy Map

You can use the **policy-map type inspect http** command in configuration mode to name the traffic policy and initiate Layer 7 HTTP deep packet inspection.

The syntax of this command is as follows:

policy-map type inspect http all-match *map_name*

The keyword and arguments are as follows:

- **http all-match**—Specifies the policy map that initiates the deep packet inspection of the HTTP protocol by the ACE. The ACE attempts to match all specified conditions against the matching classification and executes the actions of all matching classes until it encounters a deny for a match request.
- *map_name*—Name assigned to the policy map. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to create a Layer 7 HTTP deep packet inspection policy map, enter:

```
host/Admin(config)# policy-map type inspect http all-match
HTTP_INSPECT_L7POLICY
host/Admin(config-pmap-ins-http)#
```

The CLI displays the policy map configuration mode.

To remove a Layer 7 HTTP deep packet inspection policy map from the ACE, enter:

```
host1/Admin(config)# no policy-map type inspect http all-match
HTTP_INSPECT_L7POLICY
```

Adding a Layer 7 HTTP Deep Packet Inspection Policy Map Description

You can use the **description** command to provide a brief summary of the Layer 7 HTTP deep packet inspection policy map.

You must access the policy map configuration mode to specify the **description** command.

The syntax of this command is as follows:

description *text*

Use the *text* argument to enter an unquoted text string with a maximum of 240 alphanumeric characters.

To add a description that the policy map is to perform HTTP deep packet inspection, enter:

```
host1/Admin(config-pmap-ins-http)# description HTTP protocol deep
inspection of incoming traffic
```

To remove the description from the policy map, enter:

```
host1/Admin(config-pmap-ins-http)# no description
```

Including Inline Match Statements in a Layer 7 HTTP Deep Packet Inspection Policy Map

You can include a single inline match criteria in the policy map without specifying a traffic class by entering an applicable Layer 7 **match** command. The inline Layer 7 policy map **match** commands function the same as with the Layer 7 class map **match** commands. However, when you use an inline **match** command, you can specify an action for only a single match statement in the Layer 7 policy map.

Note To specify actions for multiple match statements, use a class map as described in the "Associating a Layer 7 HTTP Inspection Traffic Class with the Traffic Policy" section.

The syntax of this command is as follows:

match *name match_statement* [**insert-before** *map_name*]

The keywords, arguments, and options are as follows:

- *name*—Name assigned to the inline **match** command. Enter an unquoted text string with no spaces. The length of the inline match statement name plus the length of the policy map name with which it is associated cannot exceed a total maximum of 64 alphanumeric characters. For example, if the policy map name is L7_POLICY (nine characters), an inline match statement name under this policy cannot exceed 55 alphanumeric characters (64 - 9 = 55).
- *match_statement*—Inline match criteria to be used by the policy map. See below for details on the individual **match** commands associated with the Layer 7 HTTP deep inspection class map.
- **insert-before** *map_name*—(Optional) Places the inline **match** command ahead of an existing class map in the policy map configuration.

The syntax for the HTTP deep packet inspection policy map inline match commands is as follows:

match *name* **content** *expression* [**offset** *number*]

match *name* **content length** {**eq** *bytes* | **gt** *bytes* | **lt** *bytes* | **range** *bytes1 bytes 2*}

match *name* **content-type-verification**

match *name* **cookie secondary** [**name** *cookie_name* | **prefix** *prefix_name*] **value** *expression*

match *name* **header** {*header_name* | *header_field*} **header-value** *expression*

match *name* **header length** {**request** | **response**} {**eq** *bytes* | **gt** *bytes* | **lt** *bytes* | **range** *bytes1 bytes 2*}

match *name* **header mime-type** *mime_type*

match *name* **port-misuse** *application_category*

match *name* **request-method** {**ext** *method* | **rfc** *method*}

match *name* **strict-http**

match *name* **transfer-encoding** *coding_types*

match *name* **url** *expression*

match *name* **url length** {**eq** *bytes* | **gt** *bytes* | **lt** *bytes* | **range** *bytes1 bytes 2*}

See the "Configuring a Layer 7 HTTP Deep Inspection Class Map" section for details on the individual inline match commands.

The **match content-type-verification** and **match strict-http** commands are available only as inline **match** commands under the Layer 7 **policy-map type inspect http** command. Because these two Layer 7 HTTP deep inspection match criteria cannot be combined with other match criteria, they appear as inline **match** commands for a policy map.

These two **match** commands perform the following HTTP deep inspection functions:

- **match content-type-verification**—Verifies the content MIME-type messages with the header MIME-type. This inline **match** command limits the MIME-types in HTTP messages allowed through the ACE. It verifies that the header MIME-type value is in the internal list of supported MIME-types, and the header MIME-type matches the actual content in the data or entity-body portion of the message. If they do not match, the ACE performs one of the specified Layer 7 policy map actions: **permit** or **reset**.

Note The MIME-type HTTP inspection process requires a search up to the configured maximum content parse length of the HTTP message, which may degrade performance of the ACE.

- **match strict-http**—Enforces that the internal compliance checks verify that a message is compliant with the HTTP RFC standard, RFC 2616. If the HTTP message is not compliant, the ACE performs one of the specified Layer 7 policy map actions: **permit** or **reset**.

For example, to add an inline **match** command to a Layer 7 HTTP deep inspection policy map, enter:

```
host/Admin(config-pmap-ins-http)# match L7httpinspect port-misuse p2p
```

Associating a Layer 7 HTTP Inspection Traffic Class with the Traffic Policy

You can associate a traffic class created with the **class-map** command to associate network traffic with the traffic policy by using the **class** command.

The syntax of this command is as follows:

class *map_name*

The *map_name* argument is the name of a previously defined traffic class, configured with the **class-map** command, to associate traffic to the traffic policy. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

The CLI displays the policy map class configuration mode.

For example, to specify an existing class map in the Layer 7 policy map, enter:

```
host1/Admin(config-pmap-ins-http)# class HTTP_INSPECT_L7CLASS
host1/Admin(config-pmap-ins-http-c)#
```

To remove a class map from a Layer 7 policy map, enter:

```
host1/Admin(config-pmap-ins-http)# no class HTTP_INSPECT_L7CLASS
```

To manually insert a class map ahead of a previously specified class map, use the **insert-before** command. The ACE does not save sequence reordering through the **insert-before** command as part of the configuration.

The syntax of this command is as follows:

class *map_name1* **insert-before** *map_name2*

The keywords and arguments are as follows:

- **class** *map_name1*—Specifies the 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 64 alphanumeric characters.
- **insert-before** *map_name2*—Places the current class map ahead of an existing class map as specified by the *map_name2* argument. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to use the **insert-before** command to define the sequential order of two class maps in the policy map, enter:

```
host1/Admin(config-pmap-ins-http)# class HTTP_INSPECT_L7CLASSMAP2
insert-before HTTP_INSPECT_L7CLASS
```

To specify the **class-default** class map for the traffic policy, use the **class class-default** command. 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 ACE then matches the action specified under the **class class-default** command. The **class-default** class map has an implicit **match any** statement in it so that it matches all traffic.

Note By default, all matches are applied to both HTTP request and response messages, but the **class class-default** command is applied only to HTTP requests.

For example, to use the **class class-default** command, enter:

```
host1/Admin(config-pmap-ins-http)# class class-default
host1/Admin(config-pmap-ins-http-c)#
```

The CLI displays the policy map class configuration mode.

Specifying the Layer 7 HTTP Deep Packet Policy Actions

The default behavior of the ACE is to permit HTTP traffic. For example, if a policy map explicitly permits the HTTP GET method, other methods such as PUT will also be permitted. Only an explicit deny can drop traffic.

Specify the **permit** or **reset** command to define the action that the ACE performs on the HTTP traffic depending on whether it matches the specified commands. You apply the specified command against the single inline **match** command or the specified class map.

The Layer 7 HTTP deep packet inspection policy commands are as follows:

{**permit** | **reset**}

The keywords are as follows:

- **permit**—Allows the specified HTTP traffic to be received by the ACE if it passes the HTTP deep packet inspection match criteria specified in either the class map or an inline **match** command.
- **reset**—Denies the specified HTTP traffic by sending a TCP reset message to the client or server to close the connection.

For example, to specify the actions in the Layer 7 HTTP deep packet inspection policy map, enter:

```
host1/Admin(config)# policy-map type inspect http all-match
HTTP_DEEPINSPECT_L7POLICY

host1/Admin(config-pmap-ins-http)# class http_check
host1/Admin(config-pmap-ins-http-c)# permit
```

Because the default is to permit all HTTP packets, you must remove the class map to disable this function. For example, enter:

```
host1/Admin(config-pmap-ins-http)# no class http_check
```

By default, HTTP inspection allows traffic that does not match any of the configured Layer 7 HTTP deep packet inspection matches. You can modify this behavior by including the **class class-default** command with the **reset** action to deny the specified Layer 7 HTTP traffic. In this case, if none of the class matches configured in the Layer 7 HTTP deep packet inspection policy map are hit, the **class-default** action will be taken by the ACE. For example, you can include a class map to allow the HTTP GET method and use the **class class-default** command to block all of the other requests.

Note By default, all matches are applied to both HTTP request and response messages, but the **class class-default** command is applied only to HTTP requests.

Configuring a Layer 7 SCCP Inspection Policy

This section describes how to configure a Layer 7 SCCP inspection policy map. Throughout the CLI, SCCP is referred to as "skinny." A Layer 7 class map is not required for this feature. The ACE uses the SCCP inspection policy to filter traffic based on the message ID and to perform user-configurable actions on that traffic.

Note You can associate a maximum of 1024 instances of the same type of regular expression (regex) with a a Layer 4 policy map. This limit applies to all Layer 7 policy-map types, including generic, HTTP, RADIUS, RDP, RTSP, and SIP. You configure regexes in the following:

- Match statements in Layer 7 class maps
- Inline match statements in Layer 7 policy maps
- Layer 7 hash predictors for server farms
- Layer 7 sticky expressions in sticky groups
- Header insertion and rewrite (including SSL URL rewrite) expressions in Layer 7 action lists

This section contains the following topics:

Creating a Layer 7 SCCP Inspection Policy Map

You can create a Layer 7 SCCP inspection policy map by using the **policy-map type inspect skinny** command in configuration mode. The syntax of this command is as follows:

policy-map type inspect skinny *name*

The *map_name* argument is the name assigned to the policy map. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to create a Layer 7 SCCP inspection policy map, enter:

```
host1/Admin(config)# policy-map type inspect skinny
SCCP_INSPECT_L7POLICY
host1/Admin(config-pmap-ins-skinny)#
```

To remove the Layer 7 SCCP inspection policy map from the configuration, enter:

```
host1/Admin(config)# no policy-map type inspect skinny
SCCP_INSPECT_L7POLICY
```

Adding a Description to the Layer 7 SCCP Inspection Policy Map

You can add a description to a Layer 7 SCCP inspection policy map by using the **description** command in policy map inspection Skinny configuration mode. The syntax of this command is as follows:

description

For example, enter:

```
host1/Admin(config-pmap-ins-skinny)# description this is an SCCP
inspection policy map
```

To remove the policy map description from the configuration, enter:

```
host1/Admin(config-pmap-ins-skinny)# no description
```

Including an Inline Match Statement in a Layer 7 SCCP Inspection Policy Map

You can include a single inline match criteria in the policy map without specifying a traffic class by using the **match message-id** command in policy map inspection Skinny configuration mode. When you use an inline **match** command, you can specify an action for only a single match statement in the Layer 7 policy map.

The syntax of this command is as follows:

match *name* **message-id** {*number1* {**insert-before** *map_name*} | **range** {*number2 number3*}}

The keywords and arguments are as follows:

- *name*—Name assigned to the inline **match** command. Enter an unquoted text string with no spaces. The length of the inline match statement name plus the length of the policy map name with which it is associated cannot exceed a total maximum of 64 alphanumeric characters. For example, if the policy map name is L7_POLICY (nine characters), an inline match statement name under this policy cannot exceed 55 alphanumeric characters (64 - 9 = 55).
- **message-id**—Specifies an SCCP StationMessageID.
- *number1*—Numerical identifier of the SCCP message. Enter an integer from 0 to 65535.
- **insert-before** *map_name*—(Optional) Places the inline **match** command ahead of an existing class map in the policy map configuration.
- **range** {*number2 number3*}—Specifies a range of SCCP message IDs. Enter an integer from 0 to 65535 for the lower and the upper limits of the range. The upper limit must be greater than or equal to the lower limit.

For example, to specify an inline **match** command for a Layer 7 SCCP inspection policy map, enter:

```
host1/Admin(config-pmap-ins-skinny)# match SCCP_MATCH message-id range
100 500
host1/Admin(config-pmap-ins-skinny-m)#
```

To remove the inline match statement from the policy map, enter:

```
host1/Admin(config-pmap-ins-skinny)# no match SCCP_MATCH message-id
range 100 500
```

Specifying the Layer 7 SCCP Inspection Policy Map Action

By default, the ACE allows all SCCP packets to pass. To explicitly drop SCCP traffic, use the **reset** command as the policy map action if the specified SCCP traffic matches the inline match statement. You apply the specified action against the single inline **match** command in policy map inspection Skinny match configuration mode.

The syntax of this command is as follows:

reset

The **reset** command causes the ACE to drop the SCCP traffic that matches the inline **match** command.

For example, to specify the ACE is to drop SCCP traffic that matches the **match message-id** inline command, enter:

```
host1/Admin(config)# policy-map type inspect sccp
SCCP_INSPECT_L7POLICY
host1/Admin(config-pmap-ins-skinny)# match SCCP_MATCH message-id range
100 500
host1/Admin(config-pmap-ins-skinny-m)# reset
```

To disable the action in the Layer 7 SCCP inspection policy map, enter:

```
host1/Admin(config-pmap-ins-skinny-m)# no reset
```

Configuring a Layer 7 SIP Inspection Policy

This section describes how to configure Layer 7 SIP inspection class maps and policy maps. The ACE uses class maps to filter SIP traffic based on a variety of parameters such as, the called party, the calling party, content type, SIP URI, and so on. The ACE uses policy maps to permit or deny that traffic, depending on the actions that you specify.

Note You can associate a maximum of 1024 instances of the same type of regular expression (regex) with a a Layer 4 policy map. This limit applies to all Layer 7 policy-map types, including generic, HTTP, RADIUS, RDP, RTSP, and SIP. You configure regexes in the following:

- Match statements in Layer 7 class maps
- Inline match statements in Layer 7 policy maps
- Layer 7 hash predictors for server farms
- Layer 7 sticky expressions in sticky groups
- Header insertion and rewrite (including SSL URL rewrite) expressions in Layer 7 action lists

This section contains the following topics:

Configuring a Layer 7 SIP Inspection Class Map

This section describes how to configure a Layer 7 class map for SIP application protocol inspection. It contains the following topics:

- Defining the SIP Instant Messaging Subscriber
- Defining the Message Path Taken by SIP Messages
- Defining the SIP Request Methods
- Defining the SIP Party Registration Entities
- Defining SIP URI Checks

Creating a Layer 7 SIP Inspection Class Map

You can create a Layer 7 SIP inspection class map by using the **class-map type sip inspect** command in configuration mode.

The syntax of this command is as follows:

class-map type sip inspect [**match-all** | **match-any**] *map_name*

The keywords, arguments, and options are as follows:

- **match-all | match-any**—(Optional) Determines how the ACE performs the inspection of SIP 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) Network traffic needs to satisfy all of the match criteria (implicit AND) to match the Layer 7 SIP inspection class map. The **match-all** keyword is applicable only for match statements of different SIP inspection types. For example, specifying a **match-all** condition for SIP URI, SIP header, and SIP content statements in the same class map is valid. However, specifying a **match-all** condition for multiple SIP headers with the same names or multiple URLs in the same class map is invalid.
  - **match-any**—Network traffic needs to satisfy only one of the match criteria (implicit OR) to match the Layer 7 SIP inspection class map. The **match-any** keyword is applicable only for match statements of the same Layer 7 SIP inspection type. For example, the ACE allows you to specify a **match-any** condition for SIP URI, SIP header, and SIP content statements in the same class map and allows you to specify a **match-any** condition for multiple URLs, multiple SIP headers, or multiple SIP content statements in the same class map as long as the statements are logical. For example, you could not have two **match uri sip length** statements in the same class map, but you could have one **match uri sip length** and one **match uri tel length** statement in one class map.

- *map_name*—Name assigned to the class map. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

The CLI displays the class map SIP inspection configuration mode. To classify the SIP application inspection of traffic for evaluation by the ACE, include one or more of the following commands to configure the match criteria for the Layer 7 class map:

- **match called-party**—See the "Defining the Called Party in the SIP To Header" section.
- **match calling-party**—See the "Defining the Calling Party in the SIP From Header" section.
- **match content**—See the "Defining SIP Content Checks" section.
- **match im-subscriber**—See the "Defining the SIP Instant Messaging Subscriber" section.
- **match message-path**—See the "Defining the Message Path Taken by SIP Messages" section.
- **match request-method**—See the "Defining the SIP Request Methods" section.
- **match third-party-registration**—See the "Defining the SIP Party Registration Entities" section.
- **match uri**—See the "Defining SIP URI Checks" section.

You may include multiple **match** commands in the class map.

For example, to specify SIP_INSPECT_L7CLASS as the name of a class map and identify that all commands in the Layer 7 SIP application inspection class map must be satisfied for the ACE to indicate a match, enter:

```
host1/Admin(config)# class-map type sip inspect match-all
SIP_INSPECT_L7CLASS
host1/Admin(config-cmap-sip-insp)# match calling-id .*ABC123
host1/Admin(config-cmap-sip-insp)# match im-subscriber JOHN_Q_PUBLIC
host1/Admin(config-cmap-sip-insp)# match content type sdp
```

To remove the SIP inspection class map from the ACE, enter:

```
host1/Admin(config)# no class-map type sip inspect match-any
SIP_INSPECT_L7CLASS
```

Adding a Layer 7 Class Map Description for SIP Inspection

You can add a Layer 7 Class map description by using the **description** command in class map SIP inspection configuration mode.

The syntax of this command is as follows:

description *text*

The *text* argument is an unquoted text string with a maximum of 240 alphanumeric characters.

For example, enter:

```
host1/Admin(config-cmap-sip-insp)# description SIP inspection class
map
```

To remove the description from the class map, enter:

```
host1/Admin(config-cmap-sip-insp)# no description
```

Defining the Called Party in the SIP To Header

You can filter SIP traffic based on the called party (callee or destination) as specified in the URI of the SIP To header. The ACE does not include the display name or tag part of the field. To filter SIP traffic based on the called party, use the **match called-party** command in class map SIP inspection configuration mode.

The syntax of this command is as follows:

[*line_number*] **match called-party** *expression*

The arguments and options are as follows:

- *line_number*—(Optional) Argument that assists you in editing or deleting individual **match** commands. Enter an integer from 2 to 1024 as the line number. You can enter **no** *line_number* to delete long **match** commands instead of entering the entire line. The line numbers do not dictate a priority or sequence for the match statements.
- *expression*—Regular expression that identifies the called party in the URI of the To header. Enter a regular expression from 1 to 255 alphanumeric characters. The ACE supports the use of regular expressions for matching. Expressions are stored in a header map in the form *header-name: expression*.

Header expressions allow spaces if the spaces are escaped or quoted. See Table 3-5 for a list of the supported characters that you can use in regular expressions.

Note When matching data strings, note that the period (.) and question mark (?) characters do not have a literal meaning in regular expressions. Use brackets ([]) to match these symbols (for example, enter www[.]xyz[.]com instead of www.xyz.com). You can also use a backslash (\) to escape a dot (.) or a question mark (?).

For example, to identify the called party in the SIP To header, enter:

```
host1/Admin(config-cmap-sip-insp)# match called-party
sip:some-user@somenetwork.com
```

To remove the **match** statement from the class map, enter:

```
host1/Admin(config-cmap-sip-insp)# no match called-party
sip:some-user@somenetwork.com
```

Defining the Calling Party in the SIP From Header

You can filter SIP traffic based on the calling party (caller or source) as specified in the URI of the SIP From header. The ACE does not include the display name or tag part of the field. To filter SIP traffic based on the calling party, use the **match calling-party** command in class map SIP inspection configuration mode.

The syntax of this command is as follows:

[*line_number*] **match calling-party** *expression*

The arguments are as follows:

- *line_number*—(Optional) Argument that assists you in editing or deleting individual **match** commands. Enter an integer from 2 to 1024 as the line number. You can enter **no** *line_number* to delete long **match** commands instead of entering the entire line. The line numbers do not dictate a priority or sequence for the match statements.
- *expression*—Regular expression that identifies the calling party in the URI of the SIP From header. Enter a regular expression from 1 to 255 alphanumeric characters. The ACE supports the use of regular expressions for matching. Expressions are stored in a header map in the form *header-name: expression*.

Header expressions allow spaces if the spaces are escaped or quoted. See Table 3-5 for a list of the supported characters that you can use in regular expressions.

Note When matching data strings, note that the period (.) and question mark (?) characters do not have a literal meaning in regular expressions. Use brackets ([]) to match these symbols (for example, enter www[.]xyz[.]com instead of www.xyz.com). You can also use a backslash (\) to escape a dot (.) or a question mark (?).

For example, to identify the calling party in the SIP From header, enter:

```
host1/Admin(config-cmap-sip-insp)# match calling-party
sip:this-user@thisnetwork.com;tag=745g8
```

To remove the **match** statement from the class map, enter:

```
host1/Admin(config-cmap-sip-insp)# no match calling-party
sip:this-user@thisnetwork.com;tag=745g8
```

Defining SIP Content Checks

You can configure the ACE to perform SIP content checks based on the content length or the content type. By default, the ACE allows all content types. To define SIP content checks, use the **match content** command in class map SIP inspection configuration mode.

The syntax of this command is as follows:

[*line_number*] **match content** {**length gt** *number*} | {**type sdp** | *expression*}

The keywords, arguments, and options are as follows:

- *line_number*—(Optional) Argument that assists you in editing or deleting individual **match** commands. Enter an integer from 2 to 1024 as the line number. You can enter **no** *line_number* to delete long **match** commands instead of entering the entire line. The line numbers do not dictate a priority or sequence for the match statements.
- **length**—Specifies the SIP message body length.
- **gt**—Specifies the greater than operator.

- *number*—Maximum size of a SIP message body that the ACE allows. Enter an integer from 0 to 65534 bytes. If the message body is greater than the configured value, the ACE performs the action that you configure in the policy map.
- **type**—Specifies a content type check.
- **sdp**—Specifies that the traffic must be of type Session Description Protocol (SDP) to match the class map.
- *expression*—Regular expression that identifies the content type in the SIP message body that is required to match the class map. Enter a regular expression from 1 to 255 alphanumeric characters. The ACE supports the use of regular expressions for matching. See Table 3-5 for a list of the supported characters that you can use in regular expressions.

Note When matching data strings, note that the period (.) and question mark (?) characters do not have a literal meaning in regular expressions. Use brackets ([]) to match these symbols (for example, enter www[.]xyz[.]com instead of www.xyz.com). You can also use a backslash (\) to escape a dot (.) or a question mark (?).

For example, to configure the ACE to drop SIP packets that have content with a length greater than 4000 bytes in length, enter:

```
host1/Admin(config)# class-map type sip inspect match-all
SIP_INSP_CLASS
host1/Admin(config-cmap-sip-insp)# match content length gt 200

host1/Admin(config)# policy-map type inspect sip all-match
SIP_INSP_POLICY
host1/Admin(config-pmap-ins-sip)# class SIP_INSP_CLASS
host1/Admin(config-pmap-ins-sip-c)# deny
```

To remove the **match** statement from the class map, enter:

```
host1/Admin(config-cmap-sip-insp)# no match content length gt 200
```

Defining the SIP Instant Messaging Subscriber

You can filter SIP traffic based on the IM subscriber by using the **match im-subscriber** command in class map SIP inspection configuration mode. The syntax of this command is as follows:

[*line_number*] **match im-subscriber** *expression*

The arguments and options are as follows:

- *line_number*—(Optional) Argument that assists you in editing or deleting individual **match** commands. Enter an integer from 2 to 1024 as the line number. You can enter **no** *line_number* to delete long **match** commands instead of entering the entire line. The line numbers do not dictate a priority or sequence for the match statements.
- *expression*—Regular expression that identifies the IM subscriber. Enter a regular expression from 1 to 255 alphanumeric characters. The ACE supports the use of regular expressions for matching. Expressions are stored in a header map in the form *header-name: expression*. Header expressions allow spaces, provided that the spaces are escaped or quoted. See Table 3-5 for a list of the supported characters that you can use in regular expressions.

Note When matching data strings, note that the period (.) and question mark (?) characters do not have a literal meaning in regular expressions. Use brackets ([]) to match these symbols (for example, enter www[.]xyz[.]com instead of www.xyz.com). You can also use a backslash (\) to escape a dot (.) or a question mark (?).

For example, enter:

```
host1/Admin(config-cmap-sip-insp)# match im-subscriber John_Q_Public
```

To remove the **match** statement from the class map, enter:

```
host1/Admin(config-cmap-sip-insp)# no match im-subscriber
John_Q_Public
```

Defining the Message Path Taken by SIP Messages

SIP inspection allows you to filter messages coming from or transiting through certain SIP proxy servers. The ACE maintains a list of unauthorized SIP proxy IP addresses or URIs in the form of regular expressions. The ACE checks this list against the VIA header field in each SIP packet. The default action is to drop SIP packets with VIA fields that match the regex list.

To filter SIP traffic based on the message path, use the **match message-path** command in class map SIP inspection configuration mode.

The syntax of this command is as follows:

[*line_number*] **match message-path** *expression*

The arguments and options are as follows:

- *line_number*—(Optional) Argument that assists you in editing or deleting individual **match** commands. Enter an integer from 2 to 1024 as the line number. You can enter **no** *line_number* to delete long **match** commands instead of entering the entire line. The line numbers do not dictate a priority or sequence for the match statements.
- *expression*—Regular expression that identifies a SIP proxy server. Enter a regular expression from 1 to 255 alphanumeric characters. The ACE supports the use of regular expressions for matching. Expressions are stored in a header map in the form *header-name: expression*. Header expressions allow spaces, provided that the spaces are escaped or quoted. See Table 3-5 for a list of the supported characters that you can use in regular expressions.

Note When matching data strings, note that the period (.) and question mark (?) characters do not have a literal meaning in regular expressions. Use brackets ([]) to match these symbols (for example, enter www[.]xyz[.]com instead of www.xyz.com). You can also use a backslash (\) to escape a dot (.) or a question mark (?).

For example, enter:

```
host1/Admin(config-cmap-sip-insp)# match message-path
192.168.12.3:5060
```

To remove the **match** statement from the class map, enter:

```
host1/Admin(config-cmap-sip-insp)# no match message-path
192.168.12.3:5060
```

Defining the SIP Request Methods

You can filter SIP traffic based on the request method by using the **match request-method** command in class map SIP inspection configuration mode.

The syntax of this command is as follows:

[*line_number*] **match request-method** *method_name*

The arguments and options are as follows:

- *line_number*—(Optional) Argument that assists you in editing or deleting individual **match** commands. Enter an integer from 2 to 1024 as the line number. You can enter **no** *line_number* to delete long **match** commands instead of entering the entire line. The line numbers do not dictate a priority or sequence for the match statements.
- *method_name*—Supported SIP method using one of the following keywords:
  - **ack**
  - **bye**
  - **cancel**
  - **info**
  - **invite**
  - **message**
  - **notify**
  - **options**
  - **prack**
  - **refer**
  - **register**
  - **subscribe**
  - **unknown**
  - **update**

Note Use the **unknown** keyword to permit or deny unknown or unsupported SIP methods.

For example, to filter SIP traffic based on the INVITE request method, enter:

```
host1/Admin(config-cmap-sip-insp)# match request-method invite
```

To remove the **match** statement from the class map, enter:

```
host1/Admin(config-cmap-sip-insp)# no match request-method invite
```

Defining the SIP Party Registration Entities

SIP allows users to register other users on their behalf by sending REGISTER messages with different values in the From and To header fields. This process may pose a security threat if the REGISTER message is actually a DEREGISTER message. A malicious user could cause a Denial of Service (DoS) attack by deregistering all users on their behalf. To prevent this security threat, you can specify a list of privileged users who can register or unregister someone else on their behalf. The ACE maintains the list as a regex table. If you configure this policy, the ACE drops REGISTER messages with mismatched From and To headers and a From header value that does not match any of the privileged user IDs.

To filter SIP traffic based on third-party registrations or deregistrations, use the **match third-party-registration** command in class map SIP inspection configuration mode. The syntax of this command is as follows:

[*line_number*] **match third-party-registration** *expression*

The arguments and options are as follows:

- *line_number*—(Optional) Argument that assists you in editing or deleting individual **match** commands. Enter an integer from 2 to 1024 as the line number. You can enter **no** *line_number* to delete long **match** commands instead of entering the entire line. The line numbers do not dictate a priority or sequence for the match statements.

- *expression*—Privileged user that is authorized for third-party registrations. Enter a regular expression from 1 to 255 alphanumeric characters. The ACE supports the use of regular expressions for matching. Expressions are stored in a header map in the form *header-name: expression*. Header expressions allow spaces, provided that the spaces are escaped or quoted. See Table 3-5 for a list of the supported characters that you can use in regular expressions.

Note When matching data strings, note that the period (.) and question mark (?) characters do not have a literal meaning in regular expressions. Use brackets ([]) to match these symbols (for example, enter www[.]xyz[.]com instead of www.xyz.com). You can also use a backslash (\) to escape a dot (.) or a question mark (?).

For example, to filter SIP traffic based on SIP registrations or deregistrations, enter:

```
host1/Admin(config-cmap-sip-insp)# match third-party-registration
USER1
```

To remove the **match** statement from the class map, enter:

```
host1/Admin(config-cmap-sip-insp)# no match third-party-registration
USER1
```

Defining SIP URI Checks

You can configure the ACE to validate the length of SIP URIs or Tel URIs. A SIP URI is a user identifier that a calling party (source) uses to contact the called party (destination). A Tel URI is a telephone number that identifies the endpoint of a SIP connection. For more information about SIP URIs and Tel URIs, see RFC 2534 and RFC 3966.

To filter SIP traffic based on URIs, use the **match uri** command in class map SIP inspection configuration mode.

The syntax of this command is as follows:

[*line_number*] **match uri** {**sip** | **tel**} **length gt** *value*

The keywords, arguments, and options are as follows:

- *line_number*—(Optional) Argument that assists you in editing or deleting individual **match** commands. Enter an integer from 2 to 1024 as the line number. You can enter **no** *line_number* to delete long **match** commands instead of entering the entire line. The line numbers do not dictate a priority or sequence for the match statements.
- **sip**—Specifies the ACE validates the length of a SIP URI.
- **tel**— Specifies the ACE validates the length of a Tel URI.
- **length**—Specifies the length of the SIP or Tel URI.
- **gt**—Greater than operator.
- *value*—Maximum value for the length of the SIP URI or Tel URI in bytes. Enter an integer from 0 to 254 bytes.

For example, enter:

```
host1/Admin(config-cmap-sip-insp)# match uri sip length gt 100
```

To remove the **match** statement from the class map, enter:

```
host1/Admin(config-cmap-sip-insp)# no match uri sip length gt 100
```

Configuring a Layer 7 SIP Inspection Policy Map

This section describes how to configure a Layer 7 SIP inspection policy map. The Layer 7 policy map configures the applicable SIP inspection actions executed on the network traffic that matches the classifications defined in a class map. You then associate the completed Layer 7 SIP inspection policy with a Layer 3 and Layer 4 policy map to activate the operation on a VLAN interface (see the "Defining Layer 3 and Layer 4 Application Protocol Inspection Policy Actions" section).

This section contains the following topics:

Creating a Layer 7 SIP Policy Map

You can create a Layer 7 SIP policy map by using the **policy-map type inspect sip** command in configuration mode.

The syntax of this command is as follows:

policy-map type inspect sip all-match *map_name*

The keywords and arguments are as follows:

- **sip all-match**—Specifies the policy map that initiates the inspection of the SIP protocol packets by the ACE. The ACE attempts to match all specified conditions against the matching classification and executes the actions of all matching classes until it encounters a deny for a match request.
- *map_name*—Name assigned to the policy map. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to create a Layer 7 SIP inspection policy map, enter:

```
host1/Admin(config)# policy-map type inspect sip all-match
SIP_INSPECT_L7POLICY
host1/Admin(config-pmap-ins-sip)#
```

To remove the SIP inspection policy map from the configuration, enter:

```
host1/Admin(config)# no policy-map type inspect sip all-match
SIP_INSPECT_L7POLICY
```

Adding a Layer 7 SIP Inspection Policy Map Description

You can configure a description for the Layer 7 SIP inspection policy map by using the **description** command in policy map inspection SIP configuration mode.

The syntax of this command is as follows:

description

For example, to add a description for a Layer 7 SIP inspection policy map, enter:

```
host1/Admin(config-pmap-ins-sip)# description layer 7 sip inspection
policy
```

To remove the description from the policy map, enter:

```
host1/Admin(config-pmap-ins-sip)# no description
```

Including Inline Match Statements in a Layer 7 SIP Inspection Policy Map

You can include a single inline match criteria in the policy map without specifying a traffic class by using an applicable Layer 7 **match** command. The inline Layer 7 policy map **match** commands function the same as the Layer 7 class map **match** commands. However, when you use an inline **match** command, you can specify an action for only a single match statement in the Layer 7 policy map.

Note To specify actions for multiple match statements, use a class map as described in the "Configuring a Layer 7 SIP Inspection Class Map" section.

The syntax for an inline **match** command is as follows:

match *name match_statement* [**insert-before** *map_name*]

The keywords, arguments, and options are as follows:

- *name*—Name assigned to the inline **match** command. Enter an unquoted text string with no spaces. The length of the inline match statement name plus the length of the policy map name with which it is associated cannot exceed a total maximum of 64 alphanumeric characters. For example, if the policy map name is L7_POLICY (nine characters), an inline match statement name under this policy cannot exceed 55 alphanumeric characters (64 - 9 = 55).
- *match_statement*—Inline match criteria to be used by the policy map. See the details on the **match** commands associated with the Layer 7 SIP inspection class map.
- **insert-before** *map_name*—(Optional) Places the inline **match** command ahead of an existing class map in the policy map configuration.

The syntax for the Layer 7 SIP inspection policy map inline **match** commands is as follows:

match *name* **called-party** *expression*

match *name* **calling-party** *expression*

match *name* **content** {**length gt** *number*} | {**type sdp** | *expression*}

match *name* **im-subscriber** *expression*

match *name* **message-path** *expression*

match *name* **request-method** *method_name*

match *name* **third-party-registration** *expression*

match *name* **uri** {**sip** | **tel**} **length gt** *value*

See the "Configuring a Layer 7 SIP Inspection Class Map" section for details about the inline **match** commands.

For example, to add an inline **match** command to a Layer 7 SIP inspection policy map, enter:

```
host/Admin(config-pmap-ins-sip)# match sip_match called-party abc123.*
host/Admin(config-pmap-ins-sip-m)#
```

To remove the inline **match** command from the policy map, enter:

```
host/Admin(config-pmap-ins-sip)# no match sip_match called-party
abc123.*
```

Associating the Layer 7 SIP Inspection Class Map with the Policy Map

You can associate the Layer 7 SIP inspection class map with the Layer 7 SIP inspection policy map by using the **class** command in policy map inspection SIP configuration mode.

The syntax of this command is as follows:

class *map_name*

The *map_name* argument is the identifier of an existing Layer 7 SIP inspection class map. Enter the name as an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to associate a Layer 7 SIP inspection class map with a Layer 7 SIP inspection policy map, enter:

```
host/Admin(config-pmap-ins-sip)# class SIP_INSPECT_L7CLASS
host/Admin(config-pmap-ins-sip-c)#
```

To dissociate the class map from the policy map, enter:

```
host/Admin(config-pmap-ins-sip)# no class SIP_INSPECT_L7CLASS
```

Specifying the Layer 7 SIP Inspection Policy Map Actions

By default, the ACE allows all SIP packets to pass. To explicitly deny specific SIP commands, use one of the following commands as the action if the specified SIP traffic matches the classification. You apply the specified action against the single inline **match** command in policy map SIP inspection match configuration mode or against the specified class map in policy map SIP inspection class configuration mode.

{{**drop** | **permit** | **reset**} [**log**]} | **log**

The keywords and options are as follows:

- **drop**—Drops the SIP packet that matches the class map or the single inline **match** command.
- **permit**—(Default) Allows SIP traffic that matches the class map or the single inline **match** command to pass through the ACE.
- **reset**—Denies SIP traffic that matches the class map or the single inline **match** command and resets the connection using the TCP RESET message.
- **log**—Generates a log message for traffic that matches the class map or the single inline **match** command.

For example, to specify an action in the Layer 7 SIP inspection policy map for traffic that matches the associated Layer 7 SIP inspection class map, enter:

```
host1/Admin(config)# policy-map type inspect sip first-match
SIP_INSPECT_L7POLICY
host1/Admin(config-pmap-ins-sip)# class SIP_INSPECT_L7CLASS
host1/Admin(config-pmap-ins-sip-c)# drop
```

To specify an action in a Layer 7 SIP inspection policy map for traffic that matches a single inline **match** command, enter:

```
host1/Admin(config)# policy-map type inspect sip first-match
SIP_INSPECT_L7POLICY
host1/Admin(config-pmap-ins-sip)# match SIP_MATCH calling-party
123abc.*
host1/Admin(config-pmap-ins-sip-m)# drop
```

To disable an action in the Layer 7 SIP inspection policy map, enter:

```
host1/Admin(config-pmap-ins-sip-m)# no drop
```

Configuring a Layer 3 and Layer 4 Application Protocol Inspection Traffic Policy

This section describes how to create a Layer 3 and Layer 4 class map and policy map to classify network traffic that passes through the ACE and to perform applicable application protocol inspection actions to that traffic. The Layer 3 and Layer 4 traffic policy defines the Layer 3 and Layer 4 HTTP deep packet inspection, FTP command inspection, or application protocol inspection policy actions. Application inspection involves the examination of protocols such as DNS, FTP, HTTP, ICMP, ILS, RTSP, SCCP, and SIP to verify the protocol behavior and identify unwanted or malicious traffic that passes through the ACE.

Configuration Guidelines for Inspection Traffic Policies

Because the version A2(1.0) ACE software has strict error checks for application protocol inspection configurations, be sure that your inspection configurations meet the guidelines in this section. The error checking process in the software denies misconfigurations in inspection classifications (class maps) and displays appropriate error messages. If such misconfigurations exist in your startup- or running-configuration file before you load the software, the standby ACE in a redundant configuration may boot up to the STANDBY_COLD state. For information about redundancy states, see the *Cisco Application Control Engine Module Administration Guide*.

If the class map for the inspection traffic is generic (**match . . . any** or **class-default** is configured) so that noninspection traffic is also matched, the ACE displays an error message and does not accept the inspection configuration. For example:

```
switch/Admin(config)# class-map match-all TCP_ANY
switch/Admin(config-cmap)# match port tcp any

switch/Admin(config)# policy-map multi-match FTP_POLICY
switch/Admin(config-pmap)# class TCP_ANY
switch/Admin(config-pmap-c)# inspect ftp
Error: This class doesn't have tcp protocol and a specific port
```

The following examples show some of the generic class-map **match** statements and an ACL that are not allowed in inspection configurations:

- **match port tcp any**
- **match port udp any**
- **match port tcp range 0 65535**
- **match port udp range 0 65535**
- **match virtual-address 192.168.12.15 255.255.255.0 any**
- **match virtual-address 192.168.12.15 255.255.255.0 tcp any**
- **access-list acl1 line 10 extended permit ip any any**

For application protocol inspection, the class map must have a specific protocol (related to the inspection type) configured and a specific port or range of port numbers.

For HTTP, FTP, RTSP, Skinny, and ILS protocol inspection, the class map must have TCP as the configured protocol and a specific port or range of ports. For example, enter the following commands:

```
host1/Admin(config)# class-map match-all L4_CLASS
host1/Admin(config-cmap)# match port tcp eq www
```

For SIP protocol inspection, the class map must have TCP or UDP as the configured protocol and a specific port or range of ports. For example, enter the following commands:

```
host1/Admin(config)# class-map match-all L4_CLASS
host1/Admin(config-cmap)# match port tcp eq 124
```

or

```
host1/Admin(config-cmap)# match port udp eq 135
```

For DNS inspection, the class map must have UDP as the configured protocol and a specific port or range of ports. For example, enter the following commands:

```
host1/Admin(config)# class-map match-all L4_CLASS
host1/Admin(config-cmap)# match port udp eq domain
```

For ICMP protocol inspection, the class map must have ICMP as the configured protocol. For example, enter the following commands:

```
host1/Admin(config)# access-list ACL1 extended permit icmp
192.168.12.15 255.255.255.0 192.168.16.25 255.255.255.0 echo
```

```
host1/Admin(config)# class-map match-all L4_CLASS
host1/Admin(config-cmap)# match access-list ACL1
```

Configuring a Layer 3 and Layer 4 Class Map

You can create a Layer 3 and Layer 4 class map to classify network traffic that passes through the ACE to perform an applicable application protocol inspection policy by using the **class-map** command in configuration mode.

You can have multiple **match** commands in a single class map to specify the matching criteria. For example, you can configure class maps to define multiple access group or port commands in a group that you then associate with an application protocol inspection policy. The **match-all** and **match-any** keywords determine how the ACE evaluates the operations for multiple match statements when multiple match criteria exist in a class map.

The syntax of this command is as follows:

class-map [**match-all** | **match-any**] *map_name*

The keywords, arguments, and options are as follows:

- **match-all | match-any**—(Optional) Determines how the ACE evaluates Layer 3 and 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) All of the match criteria listed in the class map are satisfied to match the network traffic class in the class map, typically, **match** commands of different types.
  - **match-any**—Only one of the match criteria listed in the class map is satisfied to match the network traffic class in the class map, typically, **match** commands of the same type.
- *map_name*—Name assigned to the class map. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

The CLI displays the class map configuration mode. To classify network traffic that passes through the ACE for application protocol inspection, include one or more of the following commands to configure the match criteria for the class map:

- **description**—See the "Adding a Layer 3 and Layer 4 Class Map Description" section.
- **match access-list**—See the "Defining Access-List Match Criteria" section.
- **match port** —See the "Defining TCP/UDP Port Number or Port Range Match Criteria" section.

Follow these guidelines when creating a class map to define a Layer 3 and Layer 4 match classification:

- You may combine multiple **match access-list** and **match port** commands in a class map.
- The matched traffic depends on the individual **inspect** command specified in the policy map. See Table 3-1 for a summary of the application inspection protocols supported by the ACE with the IP protocol and port.

For example, to define a Layer 3 and Layer 4 class map, enter:

```
host1/Admin(config)# class-map match-all DNS_INSPECT_L4CLASS
host1/Admin(config-cmap)# description DNS application protocol
inspection of incoming traffic
host1/Admin(config-cmap)# match port udp eq domain
```

To remove a Layer 3 and Layer 4 network traffic class map from the ACE, enter:

```
host1/Admin(config)# no class-map match-all DNS_INSPECT_L4CLASS
```

This section contains the following topics:

- Adding a Layer 3 and Layer 4 Class Map Description
- Defining Access-List Match Criteria
- Defining TCP/UDP Port Number or Port Range Match Criteria

Adding a Layer 3 and Layer 4 Class Map Description

You can use the **description** command to provide a brief summary of the Layer 3 and Layer 4 class map. You must access the class map configuration mode to specify the **description** command.

The syntax of this command is as follows:

description *text*

The *text* argument is an unquoted text string with a maximum of 240 alphanumeric characters.

For example, to specify a description that the class map is to perform DNS application protocol inspection, enter:

```
host1/Admin(config)# class-map DNS_INSPECT_L4CLASS
host1/Admin(config-cmap)# description DNS application protocol
inspection of incoming traffic
```

To remove the description from the class map, enter:

```
host1/Admin(config-cmap)# no description
```

Defining Access-List Match Criteria

You can use the **match access-list** command to configure the class map to filter Layer 3 and Layer 4 network traffic on a per-flow basis by using a predefined access control list. When a packet matches an entry in an access list, and if it is a **permit** entry, the ACE allows the matching result. If it is a **deny** entry, the ACE blocks the matching result. See Chapter 1, Configuring Security Access Control Lists, for details about the creating access control lists in the ACE.

For application protocol inspection, an access list must specify explicitly the IP addresses and ports in the ACL entries. Otherwise, the ACE displays an error message.

You must access the class map configuration mode to specify the **match access-list** command.

The syntax of this command is as follows:

[*line_number*] **match access-list** *identifier*

The keywords, arguments, and options are as follows:

- *line_number*—(Optional) Argument that assists you in editing or deleting individual **match** commands. Enter an integer from 2 to 255 as the line number. You can enter **no** *line_number* to delete long **match** commands instead of entering the entire line. The line numbers do not dictate a priority or sequence for the match statements.
- *identifier*—Previously created access list identifier. Enter an unquoted text string with a maximum of 64 characters.

You can enter multiple **match access-list** commands within a single class map. You may combine multiple **match access-list** and **match port** commands in a class map.

For example, to specify that the class map is to match on access control list INBOUND_ACL1, enter:

```
host1/Admin(config)# class-map match-any DNS_INSPECT_L4CLASS
host1/Admin(config-cmap)# match access-list INBOUND_ACL1
```

To clear the access control list match criteria from the class map, enter:

```
host1/Admin(config-cmap)# no match access-list inboundacl1
```

Defining TCP/UDP Port Number or Port Range Match Criteria

You can use the **match port** command to specify a TCP or UDP port number or port range as the Layer 3 and Layer 4 network traffic matching criteria.

You must access the class map configuration mode to specify the **match port** command.

The syntax of this command is as follows:

[*line_number*] **match port** {**tcp** | **udp**} {**any** | **eq** {*port_number*} | **range** *port1 port2*}

The keywords, arguments, and options are as follows:

- *line_number*—(Optional) Argument that assists you in editing or deleting individual **match** commands. Enter an integer from 2 to 255 as the line number. You can enter **no** *line_number* to delete long **match** commands instead of entering the entire line. The line numbers do not dictate a priority or sequence for the match statements.

- **tcp | udp —**Specifies the protocol, TCP or UDP, as follows:
  - **any**—Specifies the wildcard value for the TCP or UDP port number. If you use **any** in place of either the **eq** or **range** values, packets from any incoming port will match.
  - **eq** *port_number*—Specifies that the TCP or UDP port number must match the specified value. Enter an integer from 0 to 65535. A value of 0 instructs the ACE to include all ports. Alternatively, you can enter the name of a well-known TCP port as listed in Table 3-7 or a well-known UDP port as listed in Table 3-8.
  - **range** *port1 port2*—Specifies a port range to use for the TCP or UDP port. Valid port ranges are from 0 to 65535. A value of 0 instructs the ACE to match all ports.

Table 3-7 Well-Known TCP Ports and Keywords

Port	Port Number	Description
domain	53	Domain Name System
ftp	21	File Transfer Protocol
ftp-data	20	File Transfer Protocol Data
http	80	Hypertext Transfer Protocol
https	443	HTTP over SSL protocol
irc	194	Internet Relay Chat protocol
matip-a	350	Matip Type A protocol
nntp	119	Network News Transport Protocol
pop2	109	Post Office Protocol v2
pop3	110	Post Office Protocol v3
rtsp	554	Real Time Streaming Protocol
sip	5060	Session Initiation Protocol
skinny	2000	Cisco Skinny Client Control Protocol (SCCP)
smtp	25	Simple Mail Transfer Protocol
sunrpc	111	Sun Remote Procedure Call (RPC)
telnet	23	Telnet protocol

Table 3-7 Well-Known TCP Ports and Keywords (continued)

Port	Port Number	Description
domain	53	Domain Name System
www	80	World Wide Web
xot	1998	X25 over TCP

Table 3-8 Well-Known UDP Port Numbers and Keywords

Key Word	Port Number	Description
domain	53	Domain Name System
sip	5060	Session Initiation Protocol
wsp	9200	Connectionless Wireless Session Protocol (WSP)
wsp-wtls	9202	Secure Connectionless WSP
wsp-wtp	9201	Connection-based WSP
wsp-wtp-wtls	9203	Secure Connection-based WSP

You can enter multiple **match port** commands within a single class map. You may combine multiple **match access-list** and **match port** commands in a class map.

For example, to specify that the class map is to match on TCP port number 23 (Telnet client), enter:

```
host1/Admin(config)# class-map DNS_INSPECT_L4CLASS
host1/Admin(config-cmap)# match port tcp eq 23
```

To clear the TCP or UDP port number match criteria from the class map, enter:

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

Configuring a Layer 3 and Layer 4 Policy Map

This section describes how to configure a Layer 3 and Layer 4 policy that defines an HTTP deep packet inspection, FTP command inspection, or application protocol inspection traffic policy.

This section contains the following topics:

Creating a Layer 3 and Layer 4 Policy Map

You can use the **policy-map multi-match** configuration command to configure a Layer 3 and Layer 4 policy map that defines the application inspection policies. The ACE attempts to match multiple classes within the Layer 3 and Layer 4 policy map but can match only one class within each of the sets of traffic classes. If a classification matches more than one class map, then the ACE executes all of the corresponding actions. However, for a specific feature, the ACE executes only the first matching classification action.

The syntax of this command is as follows:

policy-map multi-match *map_name*

The *map_name* argument is the name assigned to the policy map. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to create a Layer 3 and Layer 4 network traffic policy map, enter:

```
host1/Admin(config)# policy-map multi-match HTTP_INSPECT_L4POLICY
host1/Admin(config-pmap)#
```

The CLI displays the policy map configuration mode.

To remove a Layer 3 and Layer 4 policy map from the ACE, enter:

```
host1/Admin(config)# no policy-map multi-match HTTP_INSPECT_L4POLICY
```

Adding a Layer 3 and Layer 4 Policy Map Description

You can use the **description** command to provide a brief summary of the Layer 3 and Layer 4 policy map. You must access the policy map configuration mode to specify the **description** command.

The syntax of this command is as follows:

description *text*

The *text* argument is an unquoted text string with a maximum of 240 alphanumeric characters.

For example, to specify a description that the policy map is to perform DNS application protocol inspection, enter:

```
host1/Admin(config-pmap)# description DNS application protocol
inspection of incoming traffic
```

To remove the description from the policy map, enter:

```
host1/Admin(config-pmap)# no description
```

Specifying a Layer 3 and Layer 4 Traffic Class with the Traffic Policy

You can specify a traffic class created with the **class-map** command to associate network traffic with the traffic policy by using the **class** command.

The syntax of this command is as follows:

class *map_name*

The *map_name* argument is the name of a previously defined traffic class, configured with the **class-map** command, to associate traffic to the traffic policy. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

The CLI displays the policy map class configuration mode.

For example, to specify an existing class map within the Layer 3 and Layer 4 policy map, enter:

```
host1/Admin(config-pmap)# class HTTP_INSPECT_L4CLASS
host1/Admin(config-pmap-c)#
```

To remove a class map from a Layer 3 and Layer 4 policy map, enter:

```
host1/Admin(config-pmap)# no class HTTP_INSPECT_L4CLASS
```

To manually insert a class map ahead of a previously specified class map, use the **insert-before** command. The ACE does not save sequence reordering through the **insert-before** command as part of the configuration.

The syntax of this command is as follows:

class *map_name1* **insert-before** *map_name2*

The keywords and arguments are as follows:

- **class** *map_name1*—Specifies the 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 64 alphanumeric characters.
- **insert-before** *map_name2*—Places the current class map ahead of an existing class map as specified by the *map_name2* argument. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to use the **insert-before** command to define the sequential order of two class maps in the policy map, enter:

```
host1/Admin(config-pmap-c)# 5 class FTP_INSPECT_L4CLASS insert-before
HTTP_INSPECT_L4CLASS
```

To specify the **class-default** class map for the Layer 3 and Layer 4 traffic policy, use the **class class-default** command. All network 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 ACE then matches the action specified under the **class class-default** command. The **class-default** class map has an implicit **match any** statement in it so that it matches all traffic.

For example, to use the **class class-default** command, enter:

```
host1/Admin(config-pmap)# class class-default
host1/Admin(config-pmap-c)#
```

The CLI displays the policy map class configuration mode.

Defining Layer 3 and Layer 4 Application Protocol Inspection Policy Actions

You can use the **inspect** command in policy map class configuration mode to define the Layer 3 and Layer 4 HTTP deep packet inspection, FTP command inspection, or application protocol inspection policy actions. Application inspection involves the examination of protocols such as DNS, FTP, HTTP, ICMP, ILS, RTSP, SCCP, and SIP to verify the protocol behavior and identify unwanted or malicious traffic that passes through the ACE.

If you intend to perform Layer 7 application inspection of network traffic, first create a Layer 7 policy as follows:

- To perform the deep packet inspection of Layer 7 HTTP application traffic by the ACE, first create a Layer 7 policy using the **policy-map type inspect http** command (see the "Configuring a Layer 7 HTTP Deep Packet Inspection Policy Map" section). You nest the Layer 7 HTTP inspection policy by using the Layer 3 and Layer 4 **inspect http** command.
- To perform the request inspection of FTP commands, first create a Layer 7 policy by using the **policy-map type inspect ftp** command (see the "Configuring a Layer 7 FTP Command Inspection Policy Map" section). You nest the Layer 7 FTP inspection policy by using the Layer 3 and Layer 4 **inspect ftp** command.

You associate the Layer 7 policy map within the appropriate Layer 3 and Layer 4 policy map to provide an entry point for the traffic classification. Layer 7 policy maps are considered to be child policies and can only be associated within a Layer 3 and Layer 4 policy map. Only a Layer 3 and Layer 4 policy map can be applied to a VLAN interface or applied globally to all VLAN interfaces in the same context; a Layer 7 policy map cannot be directly applied on an interface.

Note If you do not specify a Layer 7 HTTP or FTP policy map, the ACE performs a general set of Layer 3 and Layer 4 HTTP or FTP protocol fixup actions. For example, the ACE performs strict HTTP.

The syntax of this command is as follows:

inspect dns [**maximum-length** *bytes*]

inspect ftp [**strict policy** *name1* | **sec-param** *conn_parammap_name1*]

inspect http [**policy** *name4* | **url-logging**]

inspect icmp [**error**]

inspect ils

inspect rtsp [**sec-param** *conn_parammap_name3*]

inspect sip [**sec-param** *conn_parammap_name4*] [**policy** *name5*]

inspect skinny [**sec-param** *conn_parammap_name5*] [**policy** *name6*]

The keywords, arguments, and options are as follows:

- **dns**—Enables DNS query inspection. DNS requires an application inspection so that DNS queries will not be subject to the generic UDP handling based on activity timeouts. Instead, the UDP connections associated with DNS queries and responses are torn down as soon as a reply to a DNS query has been received. The ACE performs the reassembly of DNS packets to verify that the packet length is less than the configured maximum length.

  maximum-length *bytes*—(Optional) Sets the maximum length of a DNS reply. Valid entries are from 512 to 65536 bytes. There is no default. If you do not set a maximum-length value, the ACE does not check the size of the reply from the DNS server.

- **ftp**—Enables FTP inspection. The ACE inspects FTP packets, translates addresses and ports embedded in the payload, and opens up a secondary channel for data.

  – **strict**—(Optional) Checks for protocol RFC compliance and prevents web browsers from sending embedded commands in FTP requests. The **strict** keyword prevents an FTP client from determining valid usernames that are supported on an FTP server. When an FTP server replies to the USER command, the ACE intercepts the 530 reply code from the FTP server and replaces it with the 331 reply code. Specifying an FTP inspection policy allows selective command filtering and also prevent the display of the FTP server system type to the FTP client. The ACE intercepts the FTP server 215 reply code and message to the SYST command and replaces the text following the reply code with asterisks.

  – **sec-param** *conn_parammap_name1*—(Optional) Specifies the name of a previously created connection parameter map used to define parameters for FTP inspection.

Note If you do not specify a Layer 7 policy map, the ACE performs a general set of Layer 3 and Layer 4 FTP protocol fixup actions.

  - **policy** *name1*—(Optional) Specifies the name assigned to a previously created Layer 7 FTP command inspection policy map to implement the inspection of Layer 7 FTP commands by the ACE. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters. Use the **inspect ftp** command in policy map class configuration mode to define the FTP command request inspection policy.

- **http**—Enables enhanced HTTP inspection on HTTP traffic. By default, the ACE allows all request methods.
  - **policy** *name4*—(Optional) Specifies the name assigned to a previously created Layer 7 HTTP application inspection policy map to implement the deep packet inspection of Layer 7 HTTP application traffic by the ACE. The inspection checks are based on configured parameters in an existing Layer 7 policy map and internal RFC compliance checks performed by the ACE. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

    Note If you do not specify a Layer 7 policy map, the ACE performs a general set of Layer 3 and Layer 4 HTTP fixup actions and internal RFC compliance checks.

  - **url-logging**—(Optional) Enables the monitoring of Layer 3 and Layer 4 traffic. This function logs every URL request that is sent in the specified class of traffic, including the source or destination IP address and the URL that is accessed.
- **icmp**—Enables ICMP payload inspection. ICMP inspection allows ICMP traffic to have a "session" so it can be inspected similarly to TCP and UDP traffic.
- **error**—(Optional) Performs a NAT of ICMP error messages. The ACE creates translation sessions for intermediate or endpoint nodes that send ICMP error messages based on the NAT configuration. The ACE overwrites the packet with the translated IP addresses.
- **ils**—Enables Internet Locator Service (ILS) protocol inspection.
- **rtsp**—Enables RTSP packet inspection. RTSP is used by RealAudio, RealNetworks, Apple QuickTime 4, RealPlayer, and Cisco IP/TV connections. The ACE monitors Setup and Response (200 OK) messages in the control channel established using TCP port 554 (no UDP support).

- **sec-param** *conn_parammap_name3*—(Optional) Specifies the name of a previously created connection parameter map used to define parameters for RTSP inspection.
- **sip**—Enables Session Initiation Protocol (SIP) inspection. SIP is used for call handling sessions and instant messaging. The ACE inspects signaling messages for media connection addresses, media ports, and embryonic connections. The ACE also performs Network Address Translations (NATs) on IP addresses that are embedded in the user-data portion of the packet.
  - **sec-param** *conn_parammap_name4*—(Optional) Specifies the name of a previously created connection parameter map used to define parameters for SIP inspection.
  - **policy** *name5*—(Optional) Specifies the name of a previously created Layer 7 SIP application inspection policy map to implement packet inspection of Layer 7 SIP application traffic by the ACE. The inspection checks are based on configured parameters in an existing Layer 7 policy map and internal RFC compliance checks performed by the ACE. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

    Note If you do not specify a Layer 7 policy map, the ACE performs a general set of Layer 3 and Layer 4 SIP protocol fixup actions and internal RFC compliance checks.

- **skinny**—Enables Cisco Skinny Client Control Protocol (SCCP) inspection. The SCCP is a Cisco proprietary protocol that is used between Cisco CallManager and Cisco VoIP phones. The ACE performs a NAT on embedded IP addresses and port numbers in SCCP packet data.
  - **policy** *name6*—(Optional) Specifies the name of a previously created deep packet inspection of Layer 7 SCCP application traffic by the ACE. The inspection checks are based on configured parameters in an existing Layer 7 policy map and internal RFC compliance checks performed by the ACE. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

    Note If you do not specify a Layer 7 policy map, the ACE performs a general set of Layer 3 and Layer 4 SCCP protocol fixup actions and internal RFC compliance checks.

– **sec-param** *conn_parammap_name5*—(Optional) Specifies the name of a previously created connection parameter map used to define parameters for SCCP inspection.

For example, to specify the **inspect http** command as an action for an HTTP application protocol inspection policy map, enter:

```
host1/Admin(config)# policy-map multi-match HTTP_INSPECT_L4POLICY
host1/Admin(config-pmap)# class HTTP_INSPECT_L4CLASS
host1/Admin(config-pmap-c)# inspect http policy
HTTP_DEEPINSPECT_L7POLICY
```

For example, to specify the **inspect dns** command as an action for a DNS application protocol inspection policy map, enter:

```
host1/Admin(config)# policy-map multi-match DNS_INSPECT_L4POLICY
host1/Admin(config-pmap)# class DNS_INSPECT_L4CLASS
host1/Admin(config-pmap-c)# inspect dns 1000
```

For example, to specify the **inspect ftp** command as an action for an FTP command inspection policy map, enter:

```
host1/Admin(config)# policy-map multi-match FTP_INSPECT_L4POLICY
host1/Admin(config-pmap)# class FTP_INSPECT_L7CLASS
host1/Admin(config-pmap-c)# inspect ftp strict policy
FTP_INSPECT_L7POLICY
host1/Admin(config-pmap-c)# exit
host1/Admin(config)#
```

To disable an application protocol inspection action from a policy map, enter:

```
host1/Admin(config-pmap-c)# no inspect dns 1000
```

Configuring a DNS Parameter Map

You can use a parameter map to apply actions to a Layer 3 and Layer 4 DNS inspection policy map. You reference this parameter map in the **appl-parameter** command in policy map class configuration mode. See the "Associating a DNS Parameter Map with a Layer 3 and Layer 4 Policy Map" section.

You can configure DNS actions for DNS packet inspection by using the **parameter-map type dns** command in configuration mode. The syntax of this command is as follows:

parameter-map type dns *name*

The *name* argument is the identifier assigned to the parameter map. Enter an unquoted text string with no spaces and a maximum of 32 alphanumeric characters.

For example, to create a parameter map called DNS_PARAMMAP, enter the following command:

```
host1/Admin(config)# parameter-map type dns DNS_PARAMMAP
host1/Admin(config-parammap-dns)#
```

To remove a DNS parameter map from the configuration, enter the following command:

```
host1/Admin(config)# no parameter-map type dns DNS_PARAMMAP
```

This section contains the following subsections:

- Configuring a DNS Query Timeout
- Associating a DNS Parameter Map with a Layer 3 and Layer 4 Policy Map

Configuring a DNS Query Timeout

When you enable DNS inspection using the **inspect dns** command as a Layer 4 policy-map action (see the "Defining Layer 3 and Layer 4 Application Protocol Inspection Policy Actions" section), the ACE stores DNS queries that it receives from clients in a hash table. When it receives a response from the DNS server, the ACE forwards the server response to the client if it finds a matching query in the table and then deletes the entry in the table. Queries, for which the ACE does not receive a response, remain in the table until they time out. The ACE may not receive an answer for a DNS query because the server is down, the query was spoofed, and so on.

If the underlying UDP connection times out, the ACE removes all DNS query hash entries using that UDP connection in 2 seconds. You can configure the UDP inactivity timeout using a connection parameter map. For details, see Chapter 1, Configuring TCP/IP Normalization and IP Reassembly Parameters.

If the ACE continues to receive DNS queries on the same UDP connection, the UDP connection does not time out. In this case, the queries without answers will time out in 10 seconds. To change this time-out value, use the **timeout query** command in DNS parameter map configuration mode. The syntax of this command is as follows:

timeout query *number*

The *number* argument specifies the length of time in seconds that the ACE keeps the query entries without answers in the hash table before timing them out. Enter an integer from 2 to 120 seconds. The default is 10 seconds.

For example, to time out DNS queries with no responses after 20 seconds, enter the following commands:

```
host1/Admin(config)# parameter-map type dns DNS_PARAMMAP
host1/Admin(config-parammap-dns)# timeout query 20
```

To reset the query timeout value to the default of 10 seconds, enter the following commands:

```
host1/Admin(config)# parameter-map type dns DNS_PARAMMAP
host1/Admin(config-parammap-dns)# no timeout query 20
```

Associating a DNS Parameter Map with a Layer 3 and Layer 4 Policy Map

You can associate a DNS parameter map with a Layer 3 and Layer 4 policy map by using the **appl-parameter dns advanced-options** command in policy map class configuration mode.

The syntax of this command is as follows:

appl-parameter dns advanced-options *name*

The *name* argument is the name of an existing DNS parameter map. Parameter maps aggregate DNS traffic-related actions together. Enter the name of an existing DNS parameter map as an unquoted text string with no spaces and a maximum of 64 alphanumeric characters. For details about configuring a DNS parameter map, see the "Configuring a DNS Parameter Map" section.

For example, to associate the DNS_PARAMMAP DNS parameter map with the L4_POLICY Layer 3 and Layer 4 policy map, enter the following commands:

```
host1/Admin(config)# policy-map multi-match L4_POLICY
host1/Admin(config-pmap)# class L4_CLASS
host1/Admin(config-pmap-c)# appl-parameter dns advanced-options
DNS_PARAMMAP
```

To disassociate the DNS parameter map from the Layer 4 policy map, enter the following commands:

```
host1/Admin(config)# policy-map multi-match L4_POLICY
host1/Admin(config-pmap)# class L4_CLASS
host1/Admin(config-pmap-c)# no appl-parameter dns advanced-options
DNS_PARAMMAP
```

Configuring an HTTP Parameter Map

You can use a parameter map to combine related actions for use in a Layer 3 and Layer 4 HTTP deep packet inspection policy map. You reference this parameter map in the **appl-parameter** command in policy map class configuration mode. See the "Associating an HTTP Parameter Map with a Layer 3 and Layer 4 Policy Map" section.

You can configure advanced HTTP behavior for HTTP deep packet inspection by using the **parameter-map type http** command in configuration mode. The syntax of this command is as follows:

parameter-map type http *name*

The *name* argument is the identifier assigned to the parameter map. Enter an unquoted text string with no spaces and a maximum of 32 alphanumeric characters.

This section contains the following topics to define the advanced HTTP parameter map:

Disabling Case-Sensitivity Matching

By default, the ACE CLI is case sensitive. To disable case-sensitivity matching for HTTP only, use the **case-insensitive** command in HTTP parameter map configuration mode. With case-insensitive matching enabled, the CLI does not distinguish between uppercase and lowercase letters. When case sensitivity is disabled, it applies to the following:

- HTTP header names and values
- URL strings
- HTTP content inspection

The syntax of this command is as follows:

case-insensitive

For example, to disable case sensitivity, enter:

```
host1/Admin(config-parammap-http)# case-insensitive
```

To reenable case-sensitive matching after it has been disabled, enter:

```
host1/Admin(config-parammap-http)# no case-insensitive
```

Setting the Maximum Number of Bytes to Parse in HTTP Headers

You can set the maximum number of bytes to parse in HTTP headers by using the **set header-maxparse-length** command in HTTP parameter-map configuration mode. The syntax of this command is as follows:

set header-maxparse-length *bytes*

The *bytes* argument is the maximum number of bytes to parse for HTTP headers. Enter an integer from 1 to 65535. The default is 2048 bytes.

For example, to set the HTTP header maximum parse length to 8192, enter:

```
host1/Admin(config-parammap-http)# set header-maxparse-length 8192
```

To reset the HTTP header maximum parse length to the default of 2048 bytes, enter:

```
host1/Admin(config-parammap-http)# no set-header maxparse-length
```

Setting the Maximum Number of Bytes to Parse in HTTP Content

You can set the maximum number of bytes to parse in HTTP content by using the **set content-maxparse-length** command in HTTP parameter map configuration mode. The syntax of this command is as follows:

set content-maxparse-length *bytes*

The *bytes* argument is the maximum number of bytes to parse in HTTP content. Enter an integer from 1 to 65535. The default is 4096 bytes.

For example, to set the maximum parse length to 8192, enter:

```
host1/Admin(config-parammap-http)# set content-maxparse-length 8192
```

To reset the maximum parse length to the default of 4096 bytes, enter:

```
host1/Admin(config-parammap-http)# no set content-maxparse-length
```

Associating an HTTP Parameter Map with a Layer 3 and Layer 4 Policy Map

You can associate an HTTP parameter map with a Layer 3 and Layer 4 policy map by using the **appl-parameter http advanced-options** command in policy map class configuration mode.

The syntax of this command is as follows:

appl-parameter http advanced-options *name*

The *name* argument is the name of an existing HTTP parameter map. Parameter maps aggregate HTTP traffic-related actions together. Enter the name of an existing HTTP parameter map as an unquoted text string with no spaces and a maximum of 64 alphanumeric characters. For details about configuring an HTTP parameter map, see the "Configuring an HTTP Parameter Map" section.

For example, to specify the **appl-parameter http advanced-options** command as an action for the HTTP deep packet inspection policy map, enter:

```
host1/Admin(config)# policy-map multi-match HTTP_INSPECT_L4POLICY
host1/Admin(config-pmap)# class HTTP_INSPECT_L4CLASS
host1/Admin(config-pmap-c)# appl-parameter http advanced-options
HTTP_PARAM_MAP1
```

To dissociate the HTTP parameter map as an action from the HTTP deep packet inspection policy map, enter:

```
host1/Admin(config-pmap-c)# no appl-parameter http advanced-options
HTTP_PARAM_MAP1
```

Configuring an SCCP Parameter Map

You can use a parameter map to combine related actions for use in a Layer 3 and Layer 4 Skinny Client Control Protocol (SCCP) application protocol inspection policy map. You reference this parameter map in the **appl-parameter** command in policy map class configuration mode. See the "Associating an SCCP Parameter Map with a Layer 3 and Layer 4 Policy Map" section.

This section contains the following topics:

SCCP Inspection Configuration Considerations

Be aware of the following considerations when you configure SCCP inspection on the ACE:

- If the ACE resides between the Cisco CallManager (CCM) and the IP phones, then explicit security ACLs are required to permit the TFTP traffic between the CCM and the phones because the ACE does not support TFTP fixup.
- If the IP address of an internal CCM is configured for NAT or PAT to a different IP address or port, registrations for external Cisco IP phones fail because the ACE does not support NAT or PAT of the file content transferred over TFTP. Although the ACE supports NAT of TFTP messages, it does not open a secure port for TFTP. In addition, the ACE cannot translate the CCM IP address and port that are embedded in the Cisco IP phone configuration files. The configuration files are transferred using TFTP during phone registration.

Creating an SCCP Parameter Map

You can configure SCCP packet inspection by using the **parameter-map type skinny** command in configuration mode. The syntax of this command is as follows:

parameter-map type skinny *name*

The *name* argument is the identifier assigned to the parameter map. Enter an unquoted text string with no spaces and a maximum of 32 alphanumeric characters.

For example, enter:

```
host1/Admin(config)# parameter-map type skinny SCCP_PARAMMAP
host1/Admin(config-parammap-skinny)#
```

To remove the parameter map from the configuration, enter:

```
host1/Admin(config)# no parameter-map type skinny SCCP_PARAMMAP
```

Enabling Registration Enforcement

You can configure the ACE to allow only registered Skinny clients to make calls. To accomplish this task, the ACE maintains the state of each Skinny client. After a client registers with CCM, the ACE opens a secure port (pinhole) to allow that client to make a call. By default, this feature is disabled.

To enable registration enforcement, use the **enforce-registration** command in parameter map Skinny configuration mode. The syntax of this command is as follows:

enforce-registration

For example, to enable registration enforcement for Skinny clients, enter:

```
host1/Admin(config-parammap-skinny)# enforce-registration
```

To disable registration enforcement, enter:

```
host1/Admin(config-parammap-skinny)# no enforce-registration
```

Setting the Maximum Message ID

You can set the maximum SCCP StationMessageID that the ACE allows by using the **message-id max** command in parameter map Skinny configuration mode. The syntax of this command is as follows:

message-id max *number*

The *number* argument is the largest value for the station message ID in hexadecimal that the ACE accepts. Enter a hexadecimal value from 0 to 4000. If a packet arrives with a station message ID greater than the maximum configured value or greater than the default value, the ACE drops the packet and generates a syslog message.

For example, to set the maximum SCCP message ID to 0x3000, enter:

```
host1/Admin(config-parammap-skinny)# message-id max 3000
```

To reset the maximum message ID to the default of 0x181, enter

```
host1/Admin(config-parammap-skinny)# no message-id max 3000
```

Setting the Minimum and Maximum SCCP Prefix Length

By default, the ACE drops SCCP messages that have an SCCP prefix length that is less than the message ID. You can configure the ACE to check for a specific minimum prefix length. You can also configure the ACE to check for a maximum prefix length, but this check is disabled by default. The ACE drops any Skinny message packets that fail these checks and generates a syslog message.

To set the minimum and maximum SCCP prefix length, use the **sccp-prefix-len** command in parameter map Skinny configuration mode. The syntax of this command is as follows:

sccp-prefix-len {**max** *number* | **min** *number*}

The keywords and arguments are as follows:

- **max** *number*—Enables the check of the maximum SCCP prefix length. Enter an integer from 4 to 4000 bytes. The default is 4 bytes.
- **min** *number*—Specifies the minimum SCCP prefix length. Enter an integer from 4 to 4000 bytes.

For example, to set the minimum SCCP prefix length, enter:

```
host1/Admin(config-parammap-skinny)# sccp-prefix-len min 4
```

To reset the minimum SCCP prefix length to the default behavior, enter:

```
host1/Admin(config-parammap-skinny)# no sccp-prefix-len min 4
```

Associating an SCCP Parameter Map with a Layer 3 and Layer 4 Policy Map

You can associate an SCCP parameter map with a Layer 3 and Layer 4 policy map by using the **appl-parameter skinny advanced-options** command in policy map class configuration mode.

The syntax of this command is as follows:

appl-parameter skinny advanced-options *name*

The *name* argument is the name of an existing SCCP parameter map. Parameter maps aggregate SCCP traffic-related actions together. Enter the name of an existing SCCP parameter map as an unquoted text string with no spaces and a maximum of 64 alphanumeric characters. For details about configuring an SCCP parameter map, see the "Configuring an SCCP Parameter Map" section.

For example, to specify the **appl-parameter skinny advanced-options** command as an action for the SCCP deep packet inspection policy map, enter:

```
host1/Admin(config)# policy-map multi-match SCCP_INSPECT_L4POLICY
host1/Admin(config-pmap)# class SCCP_INSPECT_L4CLASS
host1/Admin(config-pmap-c)# appl-parameter skinny advanced-options
SCCP_PARAM_MAP1
```

To dissociate the SCCP parameter map as an action from the SCCP packet inspection policy map, enter:

```
host1/Admin(config-pmap-c)# no appl-parameter skinny advanced-options
SCCP_PARAM_MAP1
```

Configuring a SIP Parameter Map

You can use a parameter map to combine related actions for use in a Layer 3 and Layer 4 SIP deep packet inspection policy map. You reference this parameter map in the **appl-parameter** command in policy map class configuration mode. See the "Associating a SIP Parameter Map with a Layer 3 and Layer 4 Policy Map" section.

This section contains the following topics:

- SIP Inspection Configuration Considerations
- Creating a SIP Parameter Map
- Configuring a Timeout for a SIP Media Secure Port
- Enabling Instant Messaging
- Enabling Maximum Forward Field Validation
- Configuring User Agent Software Version Options
- Enabling Strict Header Validation
- Enabling Non-SIP URI Detection in SIP Messages
- Associating a SIP Parameter Map with a Layer 3 and Layer 4 Policy Map

SIP Inspection Configuration Considerations

Be aware of the following considerations when you configure SIP inspection on the ACE:

- If the IP address in the owner field (o=) is different from the IP address in the connection field (c=) of the Session Description Protocol (SDP) portion of a SIP packet, the ACE may not translate the IP address correctly. This incorrect IP address translation is caused by a limitation of the SIP protocol, which does not provide a port value in the owner field (o=).
- If a remote endpoint attempts to register with a SIP proxy server on a network protected by the ACE, the registration fails under the following conditions:
  - PAT is configured on the remote endpoint.
  - The SIP registration server is on the outside network.
  - The port value is missing in the contact field of the REGISTER message that the endpoint sends to the proxy server.

Creating a SIP Parameter Map

You can configure advanced SIP behavior for SIP deep packet inspection by using the **parameter-map type sip** command in configuration mode.

The syntax of this command is as follows:

parameter-map type sip *name*

The *name* argument is the identifier assigned to the parameter map. Enter an unquoted text string with no spaces and a maximum of 32 alphanumeric characters.

For example, enter:

```
host1/Admin(config)# parameter-map type sip SIP_PARAMMAP
host1/Admin(config-parammap-sip)#
```

To remove the parameter map from the configuration, enter:

```
host1/Admin(config)# no parameter-map type sip SIP_PARAMMAP
```

Configuring a Timeout for a SIP Media Secure Port

The ACE opens a temporary secure port (pinhole) to stream media to a SIP client. To prevent a hacker from exploiting this port, set a timeout for SIP media by using the **timeout** command in parameter map SIP configuration mode.

The syntax of this command is as follows:

timeout sip-media *number*

The *number* argument is the timeout in seconds for the media port. Enter an integer from 1 to 65535 seconds. The default is 5 seconds. Be sure to provide a timeout value that is large enough for streaming media applications to complete.

For example, to specify a secure streaming media port timeout value of 1 hour, enter:

```
host1/Admin(config)# parameter-map type sip SIP_PARAMMAP
host1/Admin(config-parammap-sip)# timeout sip-media 3600
```

To return the streaming media port timeout value to the default of 5 seconds, enter:

```
host1/Admin(config-parammap-sip)# no timeout sip-media 3600
```

Enabling Instant Messaging

You can enable instant messaging (IM) over SIP after it has been disabled by using the **im** command in parameter map SIP configuration mode. By default, IM is enabled.

The syntax of this command is as follows:

im

For example, to enable instant messaging, enter:

```
host1/Admin(config)# parameter-map type sip SIP_PARAMMAP
host1/Admin(config-parammap-sip)# im
```

To disable instant messaging, enter:

```
host1/Admin(config-parammap-sip)# no im
```

Note Disabling IM results in the ACE dropping all messages that belong to the IM.

Enabling Maximum Forward Field Validation

The Max-Forwards header field limits the number of hops that a SIP request can take on the way to its destination. This header field contains an integer that is decremented by one at each hop. If the Max-Forwards value reaches zero before the request reaches its destination, the request is rejected with a 483 Too Many Hops error response. You can instruct the ACE to validate the Max-Forwards header field value and to take appropriate action if the validation fails.

To instruct the ACE to validate the value of the Max-Forwards header field, use the **max-forward-validation** command in parameter map configuration mode.

The syntax of this command is as follows:

max-forward-validation {**log**} | {{**drop** | **reset**} [**log**]}

The keywords and options are as follows:

- **log**—Specifies that the ACE log a max forward validation event.
- **drop**—Specifies that the ACE drop the SIP message.
- **reset**—Specifies that the ACE reset the SIP connection.

For example, to enable Max-Forwards header field validation, enter:

```
host1/Admin(config-parammap-sip)# max-forward-validation drop log
```

To disable maximum forward field validation, enter:

```
host1/Admin(config-parammap-sip)# no max-forward-validation
```

Configuring User Agent Software Version Options

If the software version of a user agent (UA) were exposed, the UA may be more vulnerable to attacks from hackers who exploit the security holes present in that particular version of software. To protect the UA from such attacks, the ACE allows you to log or mask the UA software version.

To configure the UA software version options, use the **software-version** command in parameter map SIP configuration mode.

The syntax of this command is as follows:

software-version {**log**} | {**mask** [**log**]}

The keywords are as follows:

- **log**—Specifies that the ACE log the UA software version.
- **mask**—Specifies that the ACE mask the UA software version.

For example, to configure the ACE to mask the UA software version, enter:

```
host1/Admin(config-parammap-sip)# software-version mask
```

To return the ACE behavior to the default of not checking the software version, enter:

```
host1/Admin(config-parammap-sip)# no software-version mask
```

Enabling Strict Header Validation

You can ensure the validity of SIP packet headers by configuring the ACE to check for the presence of the following mandatory SIP header fields:

- From
- To
- Call-ID
- CSeq
- Via
- Max-Forwards

If one of these header fields is missing in a SIP packet, the ACE considers that packet invalid. The ACE also checks for forbidden header fields, according to RFC 3261.

To enable strict header validation and the action that you want the ACE to perform if a SIP header does not meet the validation requirements, use the **strict-header-validation** command in parameter map SIP configuration mode.

The syntax of this command is as follows:

strict-header-validation {**log**} | {{**drop** | **reset**} [**log**]}

The keywords and options are as follows:

- **log**—Specifies that the ACE log the header validation event.
- **drop**—Specifies that the ACE drop the SIP message.

Note Use care if you plan to enable the **drop** option to ensure the validity of SIP packet headers. The **drop** option results in dropping requests which do not include the mandatory headers of that request. In some cases, the use of the **drop** option can lead to problems with some phones which do not utilize the mandatory headers in the request. For example, when a call is made and then cancelled, the phone receives a 487 Request Terminated cancel status request and transmits an ACK. However, for the Cisco IP Phone 7960, the transmitted ACK does not contain the MAX-FORWARDS header, which is a mandatory header for ACK. The ACE will then drop this packet, which can result in operational issues with the phone.

- **reset**—Specifies that the ACE reset the connection.

For example, to enable strict header validation to instruct the ACE to drop the connection if the packet header does not meet the header validation requirements, and to log the event, enter:

```
host1/Admin(config-parammap-sip)# strict-header-validation drop log
```

To disable strict header validation, enter:

```
host1/Admin(config-parammap-sip)# no strict-header-validation drop log
```

Enabling Non-SIP URI Detection in SIP Messages

You can enable detection of non-SIP URIs in SIP messages by using the **uri-non-sip** command in parameter map SIP configuration mode.

The syntax of this command is as follows:

uri-non-sip {**log**} | {**mask** [**log**]}

The keywords and options are as follows:

- **log**—Specifies that the ACE log the non-SIP URI.
- **mask**—Specifies that the ACE mask the non-SIP URI.

For example, to enable the detection of non-SIP URIs in SIP messages and to log the event, enter:

```
host1/Admin(config-parammap-sip)# uri-non-sip log
```

To disable the detection of non-SIP URIs in SIP messages, enter:

```
host1/Admin(config-parammap-sip)# no uri-non-sip log
```

Associating a SIP Parameter Map with a Layer 3 and Layer 4 Policy Map

You can associate a SIP application protocol inspection parameter map with a Layer 3 and Layer 4 policy map by using the **appl-parameter sip advanced-options** command in policy map class configuration mode.

The syntax of this command is as follows:

appl-parameter sip advanced-options *name*

The *name* argument is the name of an existing SIP parameter map. Parameter maps aggregate SIP traffic-related actions together. Enter the name of an existing SCCP parameter map as an unquoted text string with no spaces and a maximum of 64 alphanumeric characters. For details about configuring a SIP parameter map, see the "Configuring a SIP Parameter Map" section.

For example, to specify the **appl-parameter http advanced-options** command as an action for the SIP packet inspection policy map, enter:

```
host1/Admin(config)# policy-map multi-match SIP_INSPECT_L4POLICY
host1/Admin(config-pmap)# class SIP_INSPECT_L4CLASS
host1/Admin(config-pmap-c)# appl-parameter sip advanced-options
SIP_PARAM_MAP1
```

To dissociate the SIP parameter map as an action from the SIP packet inspection policy map, enter:

```
host1/Admin(config-pmap-c)# no appl-parameter sip advanced-options
SIP_PARAM_MAP1
```

Applying a Service Policy

You can use the **service-policy** command to do the following tasks:

- Apply a previously created policy map.
- Attach the traffic policy to a specific VLAN interface or globally to all VLAN interfaces in the same context.
- Specify that the traffic policy is to be attached to the input direction of an interface.

The **service-policy** command is available at both the interface configuration mode and at the configuration mode. Specifying a policy map in the interface configuration mode applies the policy map to a specific VLAN interface. Specifying a policy map in the configuration mode applies the policy to all of the VLAN interfaces associated with a context.

The syntax of this command is as follows:

service-policy input *policy_name*

The keywords and arguments are as follows:

- **input**—Specifies that the traffic policy is to be attached to the input direction of a VLAN interface. The traffic policy evaluates all traffic received by that interface.
- *policy_name*—Name of a previously defined policy map, configured with a previously created **policy-map** command. The name can be a maximum of 64 alphanumeric characters.

For example, to specify a VLAN interface and apply multiple service policies to a VLAN, enter:

```
host1/Admin(config)# interface vlan 50
host1/Admin(config-if)# ip address 172.16.1.100 255.255.255.0
host1/Admin(config-if)# service-policy input FTP_INSPECT_L4POLICY
host1/Admin(config-if)# service-policy input HTTP_INSPECT_L4POLICY
host1/Admin(config-if)# service-policy input DNS_INSPECT_L4POLICY
```

For example, to globally apply multiple service policies to all of the VLANs associated with a context, enter:

```
host1/Admin(config)# service-policy input FTP_INSPECT_L4POLICY
host1/Admin(config)# service-policy input HTTP_INSPECT_L4POLICY
host1/Admin(config)# service-policy input DNS_INSPECT_L4POLICY
```

To detach a traffic policy from a VLAN interface, enter:

```
host1/Admin(config-if)# no service-policy input DNS_INSPECT_L4POLICY
```

To globally detach a traffic policy from all VLANs associated with a context, enter:

```
host1/Admin(config)# no service-policy input DNS_INSPECT_L4POLICY
```

When you detach a traffic policy either individually from the last VLAN interface on which you applied the service policy or globally from all VLAN interfaces in the same context, the ACE automatically resets the associated service policy statistics. The ACE performs this action to provide a new starting point for the service policy statistics the next time that you attach a traffic policy to a specific VLAN interface or globally to all VLAN interfaces in the same context.

Follow these guidelines when creating a service policy:

- Policy maps, applied globally in a context, are internally applied on all interfaces existing in the context.
- A policy activated on a VLAN interface overwrites any specified global policies for overlapping classification and actions.
- The ACE allows only one policy of a specific feature type to be activated on a given interface.

Examples of Application Protocol Inspection Configurations

The following examples each illustrate a running-configuration for performing:

- Layer 7 deep packet inspection of the HTTP protocol
- Layer 7 FTP command inspection
- Layer 3 and Layer 4 DNS application protocol inspection

The application protocol inspection configurations appear in bold in each example.

Layer 7 HTTP Protocol Deep Packet Inspection

In the following HTTP protocol deep packet inspection configuration, the ACE does the following:

- Includes an ACL that allows the IM to receive any HTTP traffic through the VLAN.
- Filters on content to allow only HTTL headers that contain the “html” expression.
- Filters a subset of the HTTP traffic using a content filtering rule that permits the following packet types:
  - With an HTTP header length greater than 400 bytes
  - Without the string “BAD” included in the URL

```
access-list ACL1 extended permit tcp any any eq http

rserver host SERVER1
  ip address 192.168.252.245
  inservice
rserver host SERVER2
  ip address 192.168.252.246
  inservice
rserver host SERVER3
  ip address 192.168.252.247
  inservice
```

```
serverfarm host SFARM1
  probe HTTP
  rserver SERVER1
    inservice
  rserver SERVER2
    inservice
  rserver SERVER3
    inservice

class-map match-all L4_FILTERHTTP_CLASS
  2 match access-list ACL1
class-map type http inspect match-all L7_FILTERHTML1_CLASS
  2 match header Accept header-value "html"
  3 match header length request gt 400
class-map type http inspect match-all L7_FILTERHTML2_CLASS
  2 match url BAD
policy-map type loadbalance first-match L7_HTTP-LB-HTTP_POLICY
  class class-default
    serverfarm SFARM1
policy-map type inspect http all-match L7_FILTERHTML_POLICY
  class L7_FILTERHTML1_CLASS
    permit
  class L7_FILTERHTML2_CLASS
    reset
policy-map multi-match L4_FILTER_POLICY
  class L4_FILTERHTTP_CLASS
    inspect http policy L7_FILTERHTML_POLICY

interface vlan 50
  access-group input ACL1
  ip address 192.168.1.100 255.255.255.0
  service-policy input L4_FILTER_POLICY
  no shutdown
```

Layer 7 FTP Command Inspection

In the following FTP command inspection configuration, the ACE does the following:

- Masks the responses from the SYST and USER commands
- Denies selected FTP commands from executing
- Allows the remaining FTP commands to execute

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

rserver host SERVER1
  ip address 192.168.252.245
  inservice
rserver host SERVER2
  ip address 192.168.252.246
  inservice
rserver host SERVER3
  ip address 192.168.252.247
  inservice

serverfarm host SFARM1
  probe FTP
  rserver SERVER1
    inservice
  rserver SERVER2
    inservice
  rserver SERVER3
    inservice

class-map type ftp inspect match-any L7_FTP-MAX-DENY_CLASS
  2 match request-method appe
  3 match request-method cdup
  4 match request-method get
  5 match request-method help
  6 match request-method mkd
  7 match request-method rmd
  8 match request-method rnfr
  9 match request-method rnto
  10 match request-method site
  11 match request-method stou
  12 match request-method cwd
class-map type ftp inspect match-any L7_FTP-MAX-DENY2_CLASS
  2 match request-method syst
  3 match request-method user
class-map match-all L4_FTP-VIP_CLASS
  2 match virtual-address 192.168.120.119 tcp range 3333 4444
policy-map type loadbalance first-match L7_FTP-LB-SF-FTP_POLICY
  class class-default
    serverfarm SFARM1
policy-map type inspect ftp first-match L7_FTP-INSPSF-FTP_POLICY
  class L7_FTP-MAX-DENY_CLASS
    deny
  class L7_FTP-MAX-DENY2_CLASS
    mask-reply
```

```
policy-map multi-match L4_VIP_POLICY
  class L4_FTP-VIP_CLASS
    loadbalance vip inservice
    loadbalance policy L7_FTP-LB-SF-FTP_POLICY
    inspect ftp strict policy L7_FTP-INSPSF-FTP_POLICY

interface vlan 29
  ip address 172.16.0.1 255.255.255.0
  fragment chain 20
  fragment min-mtu 68
  nat-pool 1 192.168.120.71 192.168.120.71 netmask 255.255.255.0 pat
  no shutdown
interface vlan 120
  description Upstream VLAN_120 - Clients and VIPs
  ip address 192.168.120.1 255.255.255.0
  fragment chain 20
  fragment min-mtu 68
  access-group input ACL1
  nat-pool 1 192.168.120.70 192.168.120.70 netmask 255.255.255.0 pat
  service-policy input L4_VIP_POLICY
  no shutdown
ip route 10.1.0.0 255.255.255.0 192.168.120.254
ip route 172.16.0.0 255.252.0.0 172.16.0.253
```

Layer 3 and Layer 4 Application Protocol Inspection for DNS Inspection

In the following application protocol inspection configuration, the ACE performs DNS query inspection using a Layer 3 and Layer 4 policy map. DNS requires application inspection so that DNS queries will not be subject to the generic UDP handling based on activity timeouts. The ACE performs the reassembly of DNS packets to verify that the packet length is less than the configured maximum length of a DNS reply.

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

class-map match-any L4_DNS-INSPECT_CLASS
  description DNS application protocol inspection of incoming traffic
  match port udp eq domain
policy-map multi-match L4_DNS-INSPECT_POLICY
  class L4_DNS-INSPECT_CLASS
    inspect dns maximum length 1000
```

```
interface vlan 70
  ip address 192.168.2.1 255.255.255.0
  access-group input ACL1
  service-policy input L4_DNS-INSPECT_POLICY
  no shutdown
```

Viewing Application Protocol Inspection Statistics and Service Policy Information

The ACE CLI provides a comprehensive set of **show** commands that display application protocol inspection statistics and service policy configuration information. This section contains the following topics:

- Displaying HTTP Protocol Inspection Statistics
- Displaying Service Policy Configuration Information

Displaying HTTP Protocol Inspection Statistics

You can display FTP, HTTP, or RTSP protocol inspection statistics by using the **show stats inspect** command. The syntax of this command is as follows:

show stats inspect {**ftp** | **http** | **rtsp**}

For example, enter:

```
host1/Admin# show stats inspect http
+----------------------------------------+
+--------- HTTP Inspect statistics --------+
+----------------------------------------+
 Total request/response   : 0
 Total allow decisions    : 0
 Total drop decisions     : 0
 Total logging decisions  : 0
```

Use the **clear stats inspect** command to clear the HTTP protocol inspection statistics.

Table 3-9 describes the fields in the **show stats inspect** command output.

Table 3-9 Field Descriptions for show stats inspect http Command

Field	Description
Total Request/Response	Total number of FTP, HTTP, or RTSP packet requests or responses processed by the ACE.
Total Allow Decisions	Total number of FTP, HTTP, or RTSP packets inspected and allowed by the ACE.
Total Drop Decisions	Total number of FTP, HTTP, or RTSP okpackets inspected and denied by the ACE.
Total Logging Decisions	Total number of syslog messages generated to track the action taken by the ACE on the matching HTTP traffic. Logging is enabled as an action in the associated HTTP inspection policy map. This field is displayed only for the **http** keyword.

Displaying Service Policy Configuration Information

You can display service policy statistics by using the **show service-policy** command in Exec mode. The statistics that appear in the output are dependent on the configuration of the associated Layer 3 and Layer 4 policy map. The **show service-policy** command displays the following information:

- VLAN to which the policy is applied
- Class map associated with the policy
- Status of any load-balancing operations

The syntax of this command is as follows:

show service-policy *policy_name* [**detail**]

The keywords, arguments, and options are as follows:

- *policy_name*—Identifier of an existing policy map that is currently in service (applied to an interface) as an unquoted text string with a maximum of 64 alphanumeric characters.
- **detail**—(Optional) Displays a more detailed listing of policy map statistics and status information.

Note The ACE updates the counters that the **show service-policy** command displays after the applicable connections are closed.

For example, to display service policy statistics for the HTTP_INSPECT_L4POLICY policy map, enter:

```
host1/Admin# show service-policy HTTP_INSPECT_L4POLICY
Status     : ACTIVE
Description: HTTP protocol deep inspection of incoming traffic
----------------------------------------
Interface: vlan 40
  service-policy: HTTP_INSPECT_L4POLICY
    class: HTTP_INSPECT_L4CLASS
      inspect http:
        curr conns       : 0         , hit count        : 0
        dropped conns    : 0
       client pkt count : 0        , client byte count: 0
       server pkt count : 0        , server byte count: 0
        L4 policy stats:
          TotalReq/Resp: 0          TotalAllowed: 0
          TotalDropped : 0          TotalLogged : 0
        L7 policy: HTTP_INSPECT_L7POLICY, url logging: disabled
        L7 policy stats: Total number of L7 rules 1
          L7 class/match HTTP_INSPECT_L7CLASS: reset
            TotalInspected     : 0          TotalMatched: 0
            TotalDroppedOnError: 0          TotalLogged : 0
```

For example, to display service policy statistics for the FTP_INSPECT_L4POLICY policy map, enter:

```
host1/Admin# show service-policy FTP_INSPECT_L4POLICY
Status     : ACTIVE
Description: FTP command inspection of incoming traffic
----------------------------------------
Context Global Policy:
  service-policy: FTP_INSPECT_L4POLICY
    class: class-default
      inspect ftp:
        strict ftp: ENABLED
        curr conns       : 0         , hit count        : 0
        dropped conns    : 0
        client pkt count : 0         , client byte count: 0
        server pkt count : 0         , server byte count: 0
        L7 policy: FTP_INSPECT_L4POLICY
            TotalReplyMasked : 0          TotalDropped: 0
```

For example, to display service policy statistics for the APP_INSPECT_L4POLICY policy map, enter:

```
host1/Admin# show service-policy APP_INSPECT_L4POLICY
Status     : ACTIVE
-----------------------------------------
Context Global Policy:
  service-policy: APP_INSPECT_L4POLICY
    class: APP_INSPECT_L4CLASS
      inspect dns:
        max length: 0
        curr conns       : 0         , hit count        : 0
        dropped conns    : 0
        client pkt count : 0         , client byte count: 0
        server pkt count : 0         , server byte count: 0
```

To clear the service policy statistics, use the **clear service-policy** command. The syntax of this command is as follows:

clear service-policy *policy_name*

The *policy_name* argument, is the identifier of an existing policy map that is currently in service (applied to an interface).

For example, to clear the statistics for the policy map HTTP_INSPECT_L4POLICY that is currently in service, enter:

```
host1/Admin# clear service-policy HTTP_INSPECT_L4POLICY
```

Table 3-10 describes the fields in the **show service-policy detail** command output for an application protocol inspection policy map.

Table 3-10 Field Descriptions for the show service-policy detail Command Output

Field	Description
Status	Status of the policy map as applied in a service policy to a VLAN interface: Active or Inactive.
Description	Optional description about the policy map.
Context Global Policy	Indicates whether the service policy has been applied globally in configuration mode to all VLAN interfaces for the context.

Table 3-10 Field Descriptions for the show service-policy detail Command Output (continued)

Field	Description
Interface	VLAN identifier of the interface associated with the service policy.
Service-Policy	Identifier of the policy map.
Class	Identifier of the class map associated with the policy map.
Inspect DNS	DNS application protocol inspection statistics.
Inspect HTTP	HTTP application protocol inspection statistics.
Inspect FTP	FTP application protocol inspection statistics.
Inspect ICMP	ICMP application protocol inspection statistics.
Inspect ILS	ILS application protocol inspection statistics.
Inspect RTSP	RTSP application protocol inspection statistics.
Inspect SIP	SIP application protocol inspection statistics.
Inspect Skinny	SCCP application protocol inspection statistics.
Max Length	Maximum length of a DNS reply.
Strict FTP	Status of the strict FTP function for FTP application protocol inspection: Enabled or Disabled.
URL Logging	Status of the URL logging function for HTTP application protocol inspection: Enabled or Disabled.
ICMP Error	Status of the ICMP error function for ICMP application protocol inspection: Enabled or Disabled.
Curr Conns	Number of active connections.
Hit Count	Number of connections that the ACE.
Dropped Conns	Number of connections that the ACE discarded.
Client Pkt Count	Number of packets received from clients.
Client Byte Count	Number of bytes received from clients.
Server Pkt Count	Number of packets received from servers.
Server Byte Count	Number of bytes received from servers.

Table 3-10 Field Descriptions for the show service-policy detail Command Output (continued)

Field	Description
L4 Policy Stats	
TotalReq/ Resp	Total number of requests and responses for the policy map.
Total Allowed	Total number of packets received and allowed.
Total Dropped	Total number of packets received and discarded.
Total Logged	Total number of errors logged.
L7 Policy	Identifier of the policy map associated with the service policy.
L7 Policy Stats	Current status of the Layer 7 policy map including the total number of Layer 7 rules.
L7 Class/ Match	Identifier of the Layer 7 HTTP deep packet inspection class map and the associated policy map match actions.
Total Inspected	Total number of packets inspected.
Total Matched	Total number of packets matched.
Total Reply Masked	Total number of masked system replies to the FTP SYST command. (Applicable only to the FTP SYST command and its associated reply.)
Total Dropped On Error	Total number of packets dropped due to an error in the match.
TotalLogged	Total number of errors logged.