

Inspection for Voice and Video Protocols

The following topics explain application inspection for voice and video protocols. For basic information on why you need to use inspection for certain protocols, and the overall methods for applying inspection, see Getting Started with Application Layer Protocol Inspection.

- CTIQBE Inspection, on page 1
- H.323 Inspection, on page 2
- MGCP Inspection, on page 7
- RTSP Inspection, on page 10
- SIP Inspection, on page 13
- Skinny (SCCP) Inspection, on page 19
- STUN Inspection, on page 22
- History for Voice and Video Protocol Inspection, on page 22

CTIQBE Inspection

CTIQBE protocol inspection supports NAT, PAT, and bidirectional NAT. This enables Cisco IP SoftPhone and other Cisco TAPI/JTAPI applications to work successfully with Cisco CallManager for call setup across the ASA.

TAPI and JTAPI are used by many Cisco VoIP applications. CTIQBE is used by Cisco TSP to communicate with Cisco CallManager.

For information on enabling CTIQBE inspection, see Configure Application Layer Protocol Inspection.

Limitations for CTIQBE Inspection

Stateful failover of CTIQBE calls is not supported.

The following summarizes special considerations when using CTIQBE application inspection in specific scenarios:

- If two Cisco IP SoftPhones are registered with different Cisco CallManagers, which are connected to different interfaces of the ASA, calls between these two phones fail.
- When Cisco CallManager is located on the higher security interface compared to Cisco IP SoftPhones, if NAT or outside NAT is required for the Cisco CallManager IP address, the mapping must be static as Cisco IP SoftPhone requires the Cisco CallManager IP address to be specified explicitly in its Cisco TSP configuration on the PC.

- When using PAT or Outside PAT, if the Cisco CallManager IP address is to be translated, its TCP port 2748 must be statically mapped to the same port of the PAT (interface) address for Cisco IP SoftPhone registrations to succeed. The CTIQBE listening port (TCP 2748) is fixed and is not user-configurable on Cisco CallManager, Cisco IP SoftPhone, or Cisco TSP.

H.323 Inspection

H.323 inspection supports RAS, H.225, and H.245, and its functionality translates all embedded IP addresses and ports. It performs state tracking and filtering and can do a cascade of inspect function activation. H.323 inspection supports phone number filtering, dynamic T.120 control, H.245 tunneling control, HSI groups, protocol state tracking, H.323 call duration enforcement, T.38 Fax, and audio/video control.

H.323 inspection is enabled by default. You need to configure it only if you want non-default processing.

The following sections describe the H.323 application inspection.

H.323 Inspection Overview

H.323 inspection provides support for H.323 compliant applications such as Cisco CallManager. H.323 is a suite of protocols defined by the International Telecommunication Union for multimedia conferences over LANs. The ASA supports H.323 through Version 6, including H.323 v3 feature Multiple Calls on One Call Signaling Channel.

With H.323 inspection enabled, the ASA supports multiple calls on the same call signaling channel, a feature introduced with H.323 Version 3. This feature reduces call setup time and reduces the use of ports on the ASA.

The two major functions of H.323 inspection are as follows:

- NAT the necessary embedded IPv4 addresses in the H.225 and H.245 messages. Because H.323 messages are encoded in PER encoding format, the ASA uses an ASN.1 decoder to decode the H.323 messages.
- Dynamically allocate the negotiated H.245 and RTP/RTCP connections. The H.225 connection can also be dynamically allocated when using RAS.

How H.323 Works

The H.323 collection of protocols collectively may use up to two TCP connection and four to eight UDP connections. FastConnect uses only one TCP connection, and RAS uses a single UDP connection for registration, admissions, and status.

An H.323 client can initially establish a TCP connection to an H.323 server using TCP port 1720 to request Q.931 call setup. As part of the call setup process, the H.323 terminal supplies a port number to the client to use for an H.245 TCP connection. In environments where H.323 gatekeeper is in use, the initial packet is transmitted using UDP.

H.323 inspection monitors the Q.931 TCP connection to determine the H.245 port number. If the H.323 terminals are not using FastConnect, the ASA dynamically allocates the H.245 connection based on the inspection of the H.225 messages. The H.225 connection can also be dynamically allocated when using RAS.

Within each H.245 message, the H.323 endpoints exchange port numbers that are used for subsequent UDP data streams. H.323 inspection inspects the H.245 messages to identify these ports and dynamically creates

connections for the media exchange. RTP uses the negotiated port number, while RTCP uses the next higher port number.

The H.323 control channel handles H.225 and H.245 and H.323 RAS. H.323 inspection uses the following ports.

- 1718—Gate Keeper Discovery UDP port
- 1719—RAS UDP port
- 1720—TCP Control Port

You must permit traffic for the well-known H.323 port 1719 for RAS signaling. Additionally, you must permit traffic for the well-known H.323 port 1720 for the H.225 call signaling; however, the H.245 signaling ports are negotiated between the endpoints in the H.225 signaling. When an H.323 gatekeeper is used, the ASA opens an H.225 connection based on inspection of the ACF and RCF messages.

After inspecting the H.225 messages, the ASA opens the H.245 channel and then inspects traffic sent over the H.245 channel as well. All H.245 messages passing through the ASA undergo H.245 application inspection, which translates embedded IP addresses and opens the media channels negotiated in H.245 messages.

Each UDP connection with a packet going through H.323 inspection is marked as an H.323 connection and times out with the H.323 timeout as configured with the **timeout** command.



Note You can enable call setup between H.323 endpoints when the Gatekeeper is inside the network. The ASA includes options to open pinholes for calls based on the RegistrationRequest/RegistrationConfirm (RRQ/RCF) messages. Because these RRQ/RCF messages are sent to and from the Gatekeeper, the calling endpoint's IP address is unknown and the ASA opens a pinhole through source IP address/port 0/0. By default, this option is disabled. To enable call setup between H.323 endpoint, enter the **ras-rcf-pinholes enable** command during parameter configuration mode while creating an H.323 Inspection policy map.

H.239 Support in H.245 Messages

The ASA sits between two H.323 endpoints. When the two H.323 endpoints set up a telepresentation session so that the endpoints can send and receive a data presentation, such as spreadsheet data, the ASA ensure successful H.239 negotiation between the endpoints.

H.239 is a standard that provides the ability for H.300 series endpoints to open an additional video channel in a single call. In a call, an endpoint (such as a video phone), sends a channel for video and a channel for data presentation. The H.239 negotiation occurs on the H.245 channel.

The ASA opens pinholes for the additional media channel and the media control channel. The endpoints use open logical channel message (OLC) to signal a new channel creation. The message extension is part of H.245 version 13.

The decoding and encoding of the telepresentation session is enabled by default. H.239 encoding and decoding is preformed by ASN.1 coder.

Limitations for H.323 Inspection

H.323 inspection is tested and supported for Cisco Unified Communications Manager (CUCM) 7.0. It is not supported for CUCM 8.0 and higher. H.323 inspection might work with other releases and products.

The following are some of the known issues and limitations when using H.323 application inspection:

- PAT is supported except for extended PAT or per-session PAT.
- Static PAT may not properly translate IP addresses embedded in optional fields within H.323 messages. If you experience this kind of problem, do not use static PAT with H.323.
- Not supported with NAT between same-security-level interfaces.
- Not supported with NAT64.
- NAT with H.323 inspection is not compatible with NAT when done directly on the endpoints. If you perform NAT on the endpoints, disable H.323 inspection.

Configure H.323 Inspection Policy Map

You can create an H.323 inspection policy map to customize H.323 inspection actions if the default inspection behavior is not sufficient for your network.

Before you begin

Some traffic matching options use regular expressions for matching purposes. If you intend to use one of those techniques, first create the regular expression or regular expression class map.

Procedure

Step 1 (Optional) Create an H.323 inspection class map by performing the following steps.

A class map groups multiple traffic matches. You can alternatively identify **match** commands directly in the policy map. The difference between creating a class map and defining the traffic match directly in the inspection policy map is that the class map lets you create more complex match criteria, and you can reuse class maps.

To specify traffic that should not match the class map, use the **match not** command. For example, if the **match not** command specifies the string “example.com,” then any traffic that includes “example.com” does not match the class map.

For the traffic that you identify in this class map, you specify actions to take on the traffic in the inspection policy map.

If you want to perform different actions for each **match** command, you should identify the traffic directly in the policy map.

a) Create the class map: **class-map type inspect h323** [**match-all** | **match-any**] *class_map_name*

Where *class_map_name* is the name of the class map. The **match-all** keyword is the default, and specifies that traffic must match all criteria to match the class map. The **match-any** keyword specifies that the traffic matches the class map if it matches at least one of the criteria. The CLI enters class-map configuration mode, where you can enter one or more **match** commands.

b) (Optional) Add a description to the class map: **description** *string*

Where *string* is the description of the class map (up to 200 characters).

c) Specify the traffic on which you want to perform actions using one of the following **match** commands. If you use a **match not** command, then any traffic that does not match the criterion in the **match not** command has the action applied.

- **match** [**not**] **called-party regex** {*regex_name* | **class** *class_name*}—Matches the called party against the specified regular expression or regular expression class.
- **match** [**not**] **calling-party regex** {*regex_name* | **class** *class_name*}—Matches the calling party against the specified regular expression or regular expression class.
- **match** [**not**] **media-type** {**audio** | **data** | **video**}—Matches the media type.

Step 2 Create an H.323 inspection policy map: **policy-map type inspect h323** *policy_map_name*

Where the *policy_map_name* is the name of the policy map. The CLI enters policy-map configuration mode.

Step 3 (Optional) Add a description to the policy map: **description** *string*

Step 4 To apply actions to matching traffic, perform the following steps.

You can specify multiple **class** or **match** commands in the policy map. For information about the order of **class** and **match** commands, see How Multiple Traffic Classes are Handled.

a) Specify the traffic on which you want to perform actions using one of the following methods:

- If you created an H.323 class map, specify it by entering the following command: **class** *class_map_name*
- Specify traffic directly in the policy map using one of the **match** commands described for H.323 class maps. If you use a **match not** command, then any traffic that does not match the criterion in the **match not** command has the action applied.

b) Specify the action you want to perform on the matching traffic by entering one of the following commands:

- **drop** [**log**]—Drop the packet. For media type matches, you can include the **log** keyword to send a system log message.
- **drop-connection**—Drop the packet and close the connection. This option is available for called or calling party matching.
- **reset**—Drop the packet, close the connection, and send a TCP reset to the server and/or client. This option is available for called or calling party matching.

Step 5 To configure parameters that affect the inspection engine, perform the following steps:

a) Enter parameters configuration mode:

```
hostname(config-pmap)# parameters
hostname(config-pmap-p)#
```

b) Set one or more parameters. You can set the following options; use the **no** form of the command to disable the option:

- **ras-rcf-pinholes enable**—Enables call setup between H.323 endpoints. You can enable call setup between H.323 endpoints when the Gatekeeper is inside the network. Use this option to open pinholes for calls based on the RegistrationRequest/RegistrationConfirm (RRQ/RCF) messages. Because these RRQ/RCF messages are sent to and from the Gatekeeper, the calling endpoint's IP address is

unknown and the ASA opens a pinhole through source IP address/port 0/0. By default, this option is disabled.

- **timeout users** *time*—Sets the H.323 call duration limit (in hh:mm:ss format). To have no timeout, specify 00:00:00. Range is from 0:0:0 to 1193:0;0.
- **call-party-number**—Enforces sending call party number during call setup.
- **h245-tunnel-block action** {**drop-connection** | **log**}—Enforces H.245 tunnel blocking. Specify whether you want to drop the connection or simply log it.
- **rtp-conformance** [**enforce-payloadtype**]—Checks RTP packets flowing on the pinholes for protocol conformance. The optional enforce-payloadtype keyword enforces the payload type to be audio or video based on the signaling exchange.
- **state-checking** {**h225** | **ras**}—Enables state checking validation. You can enter the command separately to enable state checking for H.225 and RAS.
- **early-message** *message_type*—Whether to allow the specified type of H.225 messages before the H.225 SETUP message. You can allow the **facility** message to arrive early, in compliance with H.460.18.

  If you encounter call setup issues, where connections are being closed before being completed when using H.323/H.225, use this command to allow early messages. Also, ensure that you enable inspection for both H.323 RAS and H.225 (they are both enabled by default).

Step 6 While still in parameter configuration mode, you can configure HSI groups.

a) Define an HSI group and enter HSI group configuration mode: **hsi-group** *id*

   Where *id* is the HSI group ID. Range is from 0 to 2147483647.

b) Add an HSI to the HSI group using the IP address: **hsi** *ip_address*

   You can add a maximum of five hosts per HSI group.

c) Add an endpoint to the HSI group: **endpoint** *ip_address if_name*

   Where *ip_address* is the endpoint to add and *if_name* is the interface through which the endpoint is connected to the ASA. You can add a maximum of ten endpoints per HSI group.

Example

The following example shows how to configure phone number filtering:

```
hostname(config)# regex caller 1 "5551234567"
hostname(config)# regex caller 2 "5552345678"
hostname(config)# regex caller 3 "5553456789"

hostname(config)# class-map type inspect h323 match-all h323_traffic
hostname(config-pmap-c)# match called-party regex caller1
hostname(config-pmap-c)# match calling-party regex caller2

hostname(config)# policy-map type inspect h323 h323_map
hostname(config-pmap)# parameters
hostname(config-pmap-p)# class h323_traffic
```

```
hostname(config-pmap-c)# drop
```

What to do next

You can now configure an inspection policy to use the map. See Configure Application Layer Protocol Inspection.

MGCP Inspection

MGCP inspection is not enabled in the default inspection policy, so you must enable it if you need this inspection. However, the default inspect class does include the default MGCP ports, so you can simply edit the default global inspection policy to add MGCP inspection. You can alternatively create a new service policy as desired, for example, an interface-specific policy.

The following sections describe MGCP application inspection.

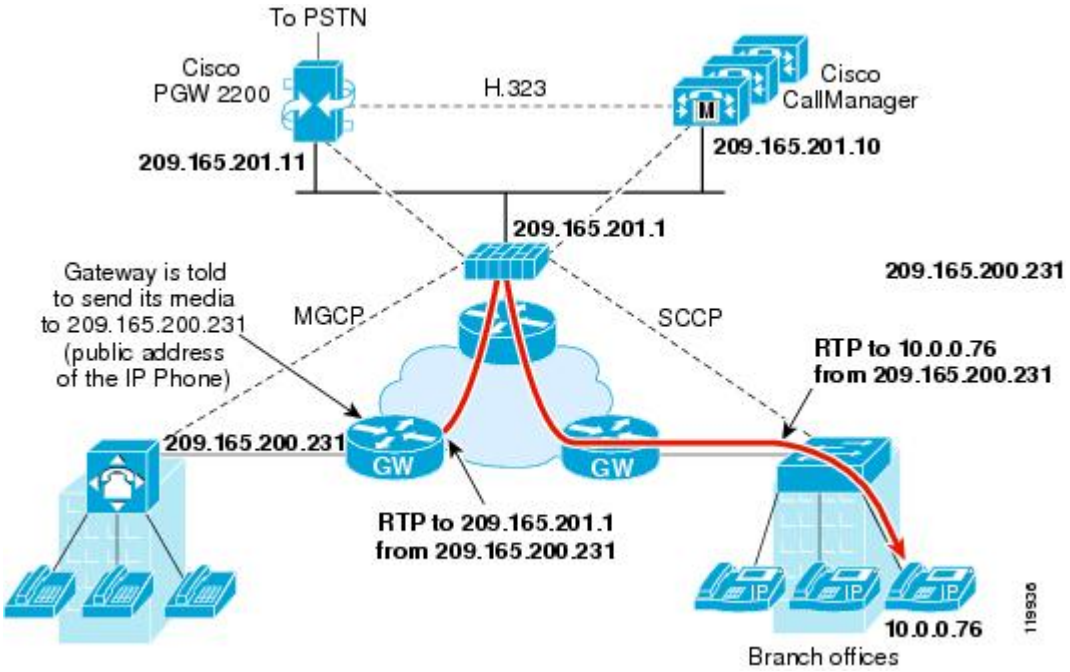
MGCP Inspection Overview

MGCP is used to control media gateways from external call control elements called media gateway controllers or call agents. A media gateway is typically a network element that provides conversion between the audio signals carried on telephone circuits and data packets carried over the Internet or over other packet networks. Using NAT and PAT with MGCP lets you support a large number of devices on an internal network with a limited set of external (global) addresses. Examples of media gateways are:

- Trunking gateways, that interface between the telephone network and a Voice over IP network. Such gateways typically manage a large number of digital circuits.
- Residential gateways, that provide a traditional analog (RJ11) interface to a Voice over IP network. Examples of residential gateways include cable modem/cable set-top boxes, xDSL devices, broad-band wireless devices.
- Business gateways, that provide a traditional digital PBX interface or an integrated soft PBX interface to a Voice over IP network.

MGCP messages are transmitted over UDP. A response is sent back to the source address (IP address and UDP port number) of the command, but the response may not arrive from the same address as the command was sent to. This can happen when multiple call agents are being used in a failover configuration and the call agent that received the command has passed control to a backup call agent, which then sends the response. The following figure illustrates how you can use NAT with MGCP.

Figure 1: Using NAT with MGCP



MGCP endpoints are physical or virtual sources and destinations for data. Media gateways contain endpoints on which the call agent can create, modify and delete connections to establish and control media sessions with other multimedia endpoints. Also, the call agent can instruct the endpoints to detect certain events and generate signals. The endpoints automatically communicate changes in service state to the call agent.

- Gateways usually listen to UDP port 2427 to receive commands from the call agent.
- The port on which the call agent receives commands from the gateway. Call agents usually listen to UDP port 2727 to receive commands from the gateway.

Note MGCP inspection does not support the use of different IP addresses for MGCP signaling and RTP data. A common and recommended practice is to send RTP data from a resilient IP address, such as a loopback or virtual IP address; however, the ASA requires the RTP data to come from the same address as MGCP signaling.

Configure an MGCP Inspection Policy Map

If the network has multiple call agents and gateways for which the ASA has to open pinholes, create an MGCP map. You can then apply the MGCP map when you enable MGCP inspection.

Procedure

Step 1 To create an MGCP inspection policy map: **policy-map type inspect mgcp** *policy_map_name*

Where the *policy_map_name* is the name of the policy map. The CLI enters policy-map configuration mode.

Step 2 (Optional) Add a description to the policy map: **description** *string*

Step 3 Enter parameters configuration mode.

```
hostname(config-pmap)# parameters
hostname(config-pmap-p)#
```

Step 4 Set one or more parameters. You can set the following options; use the **no** form of the command to disable the option.

- **call-agent** *ip_address group_id*—Configures the call agent groups that can manage one or more gateways. The call agent group information is used to open connections for the call agents in the group (other than the one a gateway sends a command to) so that any of the call agents can send the response. Call agents with the same group_id belong to the same group. A call agent may belong to more than one group. The group_id option is a number from 0 to 4294967295. The ip_address option specifies the IP address of the call agent.

  Note
  MGCP call agents send AUEP messages to determine if MGCP end points are present. This establishes a flow through the ASA and allows MGCP end points to register with the call agent.

- **gateway** *ip_address group_id*—Identifies which group of call agents is managing a particular gateway. The IP address of the gateway is specified with the ip_address option. The group_id option is a number from 0 to 4294967295 that must correspond with the group_id of the call agents that are managing the gateway. A gateway may only belong to one group.
- **command-queue** *command_limit*—Sets the maximum number of commands allowed in the MGCP command queue, from 1 to 2147483647. The default is 200.

Example

The following example shows how to define an MGCP map:

```
hostname(config)# policy-map type inspect mgcp sample_map
hostname(config-pmap)# parameters
hostname(config-pmap-p)# call-agent 10.10.11.5 101
hostname(config-pmap-p)# call-agent 10.10.11.6 101
hostname(config-pmap-p)# call-agent 10.10.11.7 102
hostname(config-pmap-p)# call-agent 10.10.11.8 102
hostname(config-pmap-p)# gateway 10.10.10.115 101
hostname(config-pmap-p)# gateway 10.10.10.116 102
hostname(config-pmap-p)# gateway 10.10.10.117 102
hostname(config-pmap-p)# command-queue 150
```

What to do next

You can now configure an inspection policy to use the map. See Configure Application Layer Protocol Inspection.

RTSP Inspection

RTSP inspection is enabled by default. You need to configure it only if you want non-default processing. The following sections describe RTSP application inspection.

RTSP Inspection Overview

The RTSP inspection engine lets the ASA pass RTSP packets. RTSP is used by RealAudio, RealNetworks, Apple QuickTime 4, RealPlayer, and Cisco IP/TV connections.



Note For Cisco IP/TV, use RTSP TCP ports 554 and 8554.

RTSP applications use the well-known port 554 with TCP (rarely UDP) as a control channel. The ASA only supports TCP, in conformity with RFC 2326. This TCP control channel is used to negotiate the data channels that are used to transmit audio/video traffic, depending on the transport mode that is configured on the client.

The supported RDT transports are: rtp/avp, rtp/avp/udp, x-real-rdt, x-real-rdt/udp, and x-pn-tng/udp.

The ASA parses Setup response messages with a status code of 200. If the response message is traveling inbound, the server is outside relative to the ASA and dynamic channels need to be opened for connections coming inbound from the server. If the response message is outbound, then the ASA does not need to open dynamic channels.

RTSP inspection does not support PAT or dual-NAT. Also, the ASA cannot recognize HTTP cloaking where RTSP messages are hidden in the HTTP messages.

RealPlayer Configuration Requirements

When using RealPlayer, it is important to properly configure transport mode. For the ASA, add an **access-list** command from the server to the client or vice versa. For RealPlayer, change transport mode by clicking **Options**>**Preferences**>**Transport**>**RTSP Settings**.

If using TCP mode on the RealPlayer, select the **Use TCP to Connect to Server** and **Attempt to use TCP for all content** check boxes. On the ASA, there is no need to configure the inspection engine.

If using UDP mode on the RealPlayer, select the **Use TCP to Connect to Server** and **Attempt to use UDP for static content** check boxes, and for live content not available via multicast. On the ASA, add an **inspect rtsp** command.

Limitations for RSTP Inspection

The following restrictions apply to the RSTP inspection.

- The ASA does not support multicast RTSP or RTSP messages over UDP.
- The ASA does not have the ability to recognize HTTP cloaking where RTSP messages are hidden in the HTTP messages.

- With Cisco IP/TV, the number of translates the ASA 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).
- You can configure NAT for Apple QuickTime 4 or RealPlayer. Cisco IP/TV only works with NAT if the Viewer and Content Manager are on the outside network and the server is on the inside network.

Configure RTSP Inspection Policy Map

You can create an RTSP inspection policy map to customize RTSP inspection actions if the default inspection behavior is not sufficient for your network.

Before you begin

Some traffic matching options use regular expressions for matching purposes. If you intend to use one of those techniques, first create the regular expression or regular expression class map.

Procedure

Step 1 (Optional) Create an RTSP inspection class map by performing the following steps.

A class map groups multiple traffic matches. You can alternatively identify **match** commands directly in the policy map. The difference between creating a class map and defining the traffic match directly in the inspection policy map is that the class map lets you create more complex match criteria, and you can reuse class maps.

To specify traffic that should not match the class map, use the **match not** command. For example, if the **match not** command specifies the string “example.com,” then any traffic that includes “example.com” does not match the class map.

For the traffic that you identify in this class map, you specify actions to take on the traffic in the inspection policy map.

If you want to perform different actions for each **match** command, you should identify the traffic directly in the policy map.

a) Create the class map: **class-map type inspect rtsp** [**match-all** | **match-any**] *class_map_name*

Where *class_map_name* is the name of the class map. The **match-all** keyword is the default, and specifies that traffic must match all criteria to match the class map. The **match-any** keyword specifies that the traffic matches the class map if it matches at least one of the criteria. The CLI enters class-map configuration mode, where you can enter one or more **match** commands.

b) (Optional) Add a description to the class map: **description** *string*

Where *string* is the description of the class map (up to 200 characters).

c) Specify the traffic on which you want to perform actions using one of the following **match** commands. If you use a **match not** command, then any traffic that does not match the criterion in the **match not** command has the action applied.

- **match** [**not**] **request-method** *method*—Matches an RTSP request method. The methods are: announce, describe, get_parameter, options, pause, play, record, redirect, setup, set_parameter, teardown.

- **match** [**not**] **url-filter regex** {*regex_name* | **class** *class_name*}—Matches the URL against the specified regular expression or regular expression class.

Step 2 To create an RTSP inspection policy map: **policy-map type inspect rtsp** *policy_map_name*

Where the *policy_map_name* is the name of the policy map. The CLI enters policy-map configuration mode.

Step 3 (Optional) Add a description to the policy map: **description** *string*

Step 4 To apply actions to matching traffic, perform the following steps.

a) Specify the traffic on which you want to perform actions using one of the following methods:

- If you created an RTSP class map, specify it by entering the following command: **class** *class_map_name*
- Specify traffic directly in the policy map using one of the **match** commands described for RTSP class maps. If you use a **match not** command, then any traffic that does not match the criterion in the **match not** command has the action applied.

b) Specify the action you want to perform on the matching traffic by entering one of the following commands:

- **drop-connection** [**log**]—Drop the packet, close the connection, and optionally send a system log message. This option is available for URL matching.
- **log**—Send a system log message.
- **rate-limit** *message_rate*—Limit the rate of messages per second. This option is available for request method matching.

You can specify multiple **class** or **match** commands in the policy map. For information about the order of **class** and **match** commands, see How Multiple Traffic Classes are Handled.

Step 5 To configure parameters that affect the inspection engine, perform the following steps:

a) Enter parameters configuration mode:

```
hostname(config-pmap)# parameters
hostname(config-pmap-p)#
```

b) Set one or more parameters. You can set the following options; use the **no** form of the command to disable the option:

- **reserve-port-protect**—Restricts the use of reserve ports during media negotiation.
- **url-length-limit** *bytes*—Sets a limit on the URL length allowed in the message, from 0 to 6000 bytes.

Example

The following example shows a how to define an RTSP inspection policy map.

```
hostname(config)# regex badurl1 www.url1.com/rtsp.avi
hostname(config)# regex badurl2 www.url2.com/rtsp.rm
```

```
hostname(config)# regex badurl3 www.url3.com/rtsp.asp

hostname(config)# class-map type regex match-any badurl-list
hostname(config-cmap)# match regex badurl1
hostname(config-cmap)# match regex badurl2
hostname(config-cmap)# match regex badurl3

hostname(config)# policy-map type inspect rtsp rtsp-filter-map
hostname(config-pmap)# match url-filter regex class badurl-list
hostname(config-pmap-p)# drop-connection

hostname(config)# class-map rtsp-traffic-class
hostname(config-cmap)# match default-inspection-traffic

hostname(config)# policy-map rtsp-traffic-policy
hostname(config-pmap)# class rtsp-traffic-class
hostname(config-pmap-c)# inspect rtsp rtsp-filter-map

hostname(config)# service-policy rtsp-traffic-policy global
```

What to do next

You can now configure an inspection policy to use the map. See Configure Application Layer Protocol Inspection.

SIP Inspection

SIP is a widely used protocol for Internet conferencing, telephony, presence, events notification, and instant messaging. Partially because of its text-based nature and partially because of its flexibility, SIP networks are subject to a large number of security threats.

SIP application inspection provides address translation in message header and body, dynamic opening of ports and basic sanity checks. It also supports application security and protocol conformance, which enforce the sanity of the SIP messages, as well as detect SIP-based attacks.

SIP inspection is enabled by default. You need to configure it only if you want non-default processing, or if you want to identify a TLS proxy to enable encrypted traffic inspection. The following topics explain SIP inspection in more detail.

SIP Inspection Overview

SIP, as defined by the IETF, enables call handling sessions, particularly two-party audio conferences, or “calls.” SIP works with SDP for call signaling. SDP specifies the ports for the media stream. Using SIP, the ASA can support any SIP VoIP gateways and VoIP proxy servers. SIP and SDP are defined in the following RFCs:

- SIP: Session Initiation Protocol, RFC 3261
- SDP: Session Description Protocol, RFC 2327

To support SIP calls through the ASA, signaling messages for the media connection addresses, media ports, and embryonic connections for the media must be inspected, because while the signaling is sent over a well-known destination port (UDP/TCP 5060), the media streams are dynamically allocated. Also, SIP embeds

IP addresses in the user-data portion of the IP packet. Note that the maximum length of the SIP Request URI that the ASA supports is 255.

Instant Messaging (IM) applications also use SIP extensions (defined in RFC 3428) and SIP-specific event notifications (RFC 3265). After users initiate a chat session (registration/subscription), the IM applications use the MESSAGE/INFO methods and 202 Accept responses when users chat with each other. For example, two users can be online at any time, but not chat for hours. Therefore, the SIP inspection engine opens pinholes that time out according to the configured SIP timeout value. This value must be configured at least five minutes longer than the subscription duration. The subscription duration is defined in the Contact Expires value and is typically 30 minutes.

Because MESSAGE/INFO requests are typically sent using a dynamically allocated port other than port 5060, they are required to go through the SIP inspection engine.



Note SIP inspection supports the Chat feature only. Whiteboard, File Transfer, and Application Sharing are not supported. RTC Client 5.0 is not supported.

Limitations for SIP Inspection

SIP inspection is tested and supported for Cisco Unified Communications Manager (CUCM) 7.0, 8.0, 8.6, and 10.5. It is not supported for CUCM 8.5, or 9.x. SIP inspection might work with other releases and products.

SIP inspection does not support the T.38 MIME Internet Facsimile Protocol(IFP). SIP inspection drops SIP invitations that use the T.38 MIME audio sub-type. If you need to allow this type, disable SIP inspection and write an access control rule that allows the RTP streams.

NAT Limitations for SIP Inspection

- SIP inspection applies NAT for embedded IP addresses. However, if you configure NAT to translate both source and destination addresses, the external address ("from" in the SIP header for the "trying" response message) is not rewritten. Thus, you should use object NAT when working with SIP traffic so that you avoid translating the destination address.
- Do not configure NAT or PAT for interfaces with the same, or lower (source) to higher (destination), security levels. This configuration is not supported.
- If you configure SIP inspection for a targeted traffic class (that is, not the inspection_default traffic class), ensure that you use a bi-directional ACL and that you specify the 5060 destination port only. Otherwise, you might have NAT problems where the IP address in the SIP header is not translated even though the IP packet is correctly translated.
- If you hard-code the mapped address in the VIA header of the SIP invitation, do not enable SIP inspection. You can run into problems if you use static NAT to translate the source client address, and the interface for the default route is different than the interface for the connected route used by the client.

PAT Limitations for SIP Inspection

The following limitations and restrictions apply when using PAT with SIP:

- If a remote endpoint tries to register with a SIP proxy on a network protected by the ASA, the registration fails under very specific conditions, as follows:

- PAT is configured for the remote endpoint.
- The SIP registrar server is on the outside network.
- The port is missing in the contact field in the REGISTER message sent by the endpoint to the proxy server.

- If a SIP device transmits a packet in which the SDP portion has an IP address in the owner/creator field (o=) that is different than the IP address in the connection field (c=), the IP address in the o= field may not be properly translated. This is due to a limitation in the SIP protocol, which does not provide a port value in the o= field. Because PAT needs a port to translate, the translation fails.
- When using PAT, any SIP header field which contains an internal IP address without a port might not be translated and hence the internal IP address will be leaked outside. If you want to avoid this leakage, configure NAT instead of PAT.

Default SIP Inspection

SIP inspection is enabled by default using the default inspection map, which includes the following:

- SIP instant messaging (IM) extensions: Enabled.
- Non-SIP traffic on SIP port: Dropped.
- Hide server's and endpoint's IP addresses: Disabled.
- Mask software version and non-SIP URIs: Disabled.
- Ensure that the number of hops to destination is greater than 0: Enabled.
- RTP conformance: Not enforced.
- SIP conformance: Do not perform state checking and header validation.

Also note that inspection of encrypted traffic is not enabled. You must configure a TLS proxy to inspect encrypted traffic.

Configure SIP Inspection Policy Map

You can create a SIP inspection policy map to customize SIP inspection actions if the default inspection behavior is not sufficient for your network.

Before you begin

Some traffic matching options use regular expressions for matching purposes. If you intend to use one of those techniques, first create the regular expression or regular expression class map.

Procedure

Step 1 (Optional) Create a SIP inspection class map by performing the following steps.

A class map groups multiple traffic matches. You can alternatively identify **match** commands directly in the policy map. The difference between creating a class map and defining the traffic match directly in the inspection policy map is that the class map lets you create more complex match criteria, and you can reuse class maps.

To specify traffic that should not match the class map, use the **match not** command. For example, if the **match not** command specifies the string "example.com," then any traffic that includes "example.com" does not match the class map.

For the traffic that you identify in this class map, you specify actions to take on the traffic in the inspection policy map.

If you want to perform different actions for each **match** command, you should identify the traffic directly in the policy map.

a) Create the class map: **class-map type inspect sip** [**match-all** | **match-any**] *class_map_name*

   Where *the class_map_name* is the name of the class map. The **match-all** keyword is the default, and specifies that traffic must match all criteria to match the class map. The match-any keyword specifies that the traffic matches the class map if it matches at least one **match** statement. The CLI enters class-map configuration mode, where you can enter one or more **match** commands.

b) (Optional) Add a description to the class map: **description** *string*

   Where *string* is the description of the class map (up to 200 characters).

c) Specify the traffic on which you want to perform actions using one of the following **match** commands. If you use a **match not** command, then any traffic that does not match the criterion in the **match not** command has the action applied.

   - **match** [**not**] **called-party regex** {*regex_name* | **class** *class_name*}—Matches the called party, as specified in the To header, against the specified regular expression or regular expression class.
   - **match** [**not**] **calling-party regex** {*regex_name* | **class** *class_name*}—Matches the calling party, as specified in the From header, against the specified regular expression or regular expression class.
   - **match** [**not**] **content length gt** *bytes*—Matches messages where the content length in the SIP header is greater than the specified number of bytes, from 0 to 65536.
   - **match** [**not**] **content type** {**sdp** | **regex** {*regex_name* | **class** *class_name*}—Matches the content type as SDP or against the specified regular expression or regular expression class.
   - **match** [**not**] **im-subscriber regex** {*regex_name* | **class** *class_name*}—Matches the SIP IM subscriber against the specified regular expression or regular expression class.
   - **match** [**not**] **message-path regex** {*regex_name* | **class** *class_name*}—Matches the SIP via header against the specified regular expression or regular expression class.
   - **match** [**not**] **request-method** *method*—Matches a SIP request method: ack, bye, cancel, info, invite, message, notify, options, prack, refer, register, subscribe, unknown, update.
   - **match** [**not**] **third-party-registration regex** {*regex_name* | **class** *class_name*}—Matches the requester of a third-party registration against the specified regular expression or regular expression class.
   - **match** [**not**] **uri {sip | tel} length gt** *bytes*—Matches a URI in the SIP headers of the selected type (SIP or TEL) that is greater than the length specified, between 0 and 65536 bytes.

d) Enter **exit** to leave class map configuration mode.

Step 2 Create a SIP inspection policy map: **policy-map type inspect sip** *policy_map_name*

Where the *policy_map_name* is the name of the policy map. The CLI enters policy-map configuration mode.

Step 3 (Optional) Add a description to the policy map: **description** *string*

Step 4 To apply actions to matching traffic, perform the following steps.

a) Specify the traffic on which you want to perform actions using one of the following methods:

- If you created a SIP class map, specify it by entering the following command: **class** *class_map_name*
- Specify traffic directly in the policy map using one of the **match** commands described for SIP class maps. If you use a **match not** command, then any traffic that does not match the criterion in the **match not** command has the action applied.

b) Specify the action you want to perform on the matching traffic by entering one of the following commands:

- **drop**—Drop all packets that match.
- **drop-connection**—Drop the packet and close the connection.
- **reset**—Drop the packet, close the connection, and send a TCP reset to the server and/or client.
- **log**—Send a system log message. You can use this option alone or with one of the other actions.
- **rate-limit** *message_rate*—Limits the rate of messages. Rate limiting is available for request method matches to "invite" and "register" only.

You can specify multiple **class** or **match** commands in the policy map. For information about the order of **class** and **match** commands, see How Multiple Traffic Classes are Handled.

Step 5 To configure parameters that affect the inspection engine, perform the following steps:

a) Enter parameters configuration mode:

```
hostname(config-pmap)# parameters
hostname(config-pmap-p)#
```

b) Set one or more parameters. You can set the following options; use the **no** form of the command to disable the option:

- **im**—Enables instant messaging.
- **ip-address-privacy**—Enables IP address privacy, which hides the server and endpoint IP addresses.
- **max-forwards-validation action** {**drop** | **drop-connection** | **reset** | **log**} [**log**]—Checks the value of the Max-Forwards header, which cannot be zero before reaching the destination. You must also choose the action to take for non-conforming traffic (drop packet, drop connection, reset, or log) and whether to enable or disable logging.
- **rtp-conformance** [**enforce-payloadtype**]—Checks RTP packets flowing on the pinholes for protocol conformance. The optional enforce-payloadtype keyword enforces the payload type to be audio or video based on the signaling exchange.
- **software-version action** {**mask [log]** | **log**}—Identifies the software version using the Server and User-Agent (endpoint) header fields. You can mask the software version in the SIP messages and optionally log it, or simply log it.

- **state-checking action** {**drop** | **drop-connection** | **reset** | **log**} [**log**]—Enables state transition checking. You must also choose the action to take for non-conforming traffic (drop packet, drop connection, reset, or log) and whether to enable or disable logging.
- **strict-header-validation action** {**drop** | **drop-connection** | **reset** | **log**} [**log**]—Enables strict verification of the header fields in the SIP messages according to RFC 3261. You must also choose the action to take for non-conforming traffic (drop packet, drop connection, reset, or log) and whether to enable or disable logging.
- **traffic-non-sip**—Allows non-SIP traffic on the well-known SIP signaling port.
- **trust-verification-server ip** *ip_address*—Identifies Trust Verification Services servers, which enable Cisco Unified IP Phones to authenticate application servers during HTTPS establishment. You can enter the command up to four times to identify four servers. SIP inspection opens pinholes to each server for each registered phone, and the phone decides which to use. Configure the Trust Verification Services server on the CUCM server.
- **trust-verification-server port** *number*—Identifies the Trust Verification Services port. The default port is 2445, so use this command only if the server uses a different port. The allowed port range is 1026 to 32768.
- **uri-non-sip action** {**mask [log]** | **log**}—Identifies the non-SIP URIs present in the Alert-Info and Call-Info header fields. You can mask the information in the SIP messages and optionally log it, or simply log it.

Examples

The following example shows how to disable instant messaging over SIP:

```
hostname(config)# policy-map type inspect sip mymap
hostname(config-pmap)# parameters
hostname(config-pmap-p)# no im

hostname(config)# policy-map global_policy
hostname(config-pmap)# class inspection_default
hostname(config-pmap-c)# inspect sip mymap

hostname(config)# service-policy global_policy global
```

The following example shows how to identify four Trust Verification Services servers.

```
hostname(config)# policy-map type inspect sip sample_sip_map
hostname(config-pmap)# parameters
hostname(config-pmap-p)# trust-verification-server ip 10.1.1.1
hostname(config-pmap-p)# trust-verification-server ip 10.1.1.2
hostname(config-pmap-p)# trust-verification-server ip 10.1.1.3
hostname(config-pmap-p)# trust-verification-server ip 10.1.1.4
hostname(config-pmap-p)# trust-verification-server port 2445
```

What to do next

You can now configure an inspection policy to use the map. See Configure Application Layer Protocol Inspection.

Skinny (SCCP) Inspection

SCCP (Skinny) application inspection performs translation of embedded IP address and port numbers within the packet data, and dynamic opening of pinholes. It also performs additional protocol conformance checks and basic state tracking.

SCCP inspection is enabled by default. You need to configure it only if you want non-default processing, or if you want to identify a TLS proxy to enable encrypted traffic inspection.

The following sections describe SCCP application inspection.

SCCP Inspection Overview

Skinny (SCCP) is a simplified protocol used in VoIP networks. Cisco IP Phones using SCCP can coexist in an H.323 environment. When used with Cisco CallManager, the SCCP client can interoperate with H.323 compliant terminals.

The ASA supports PAT and NAT for SCCP. PAT is necessary if you have more IP phones than global IP addresses for the IP phones to use. By supporting NAT and PAT of SCCP Signaling packets, Skinny application inspection ensures that all SCCP signaling and media packets can traverse the ASA.

Normal traffic between Cisco CallManager and Cisco IP Phones uses SCCP and is handled by SCCP inspection without any special configuration. The ASA also supports DHCP options 150 and 66, which it accomplishes by sending the location of a TFTP server to Cisco IP Phones and other DHCP clients. Cisco IP Phones might also include DHCP option 3 in their requests, which sets the default route.



Note The ASA supports inspection of traffic from Cisco IP Phones running SCCP protocol version 22 and earlier.

Supporting Cisco IP Phones

In topologies where Cisco CallManager is located on the higher security interface with respect to the Cisco IP Phones, if NAT is required for the Cisco CallManager IP address, the mapping must be static as a Cisco IP Phone requires the Cisco CallManager IP address to be specified explicitly in its configuration. A static identity entry allows the Cisco CallManager on the higher security interface to accept registrations from the Cisco IP Phones.

Cisco IP Phones require access to a TFTP server to download the configuration information they need to connect to the Cisco CallManager server.

When the Cisco IP Phones are on a lower security interface compared to the TFTP server, you must use an ACL to connect to the protected TFTP server on UDP port 69. While you do need a static entry for the TFTP server, this does not have to be an identity static entry. When using NAT, an identity static entry maps to the same IP address. When using PAT, it maps to the same IP address and port.

When the Cisco IP Phones are on a higher security interface compared to the TFTP server and Cisco CallManager, no ACL or static entry is required to allow the Cisco IP Phones to initiate the connection.

Limitations for SCCP Inspection

SCCP inspection is tested and supported for Cisco Unified Communications Manager (CUCM) 7.0, 8.0, 8.6, and 10.5. It is not supported for CUCM 8.5, or 9.x. SCCP inspection might work with other releases and products.

If the address of an internal Cisco CallManager is configured for NAT or PAT to a different IP address or port, registrations for external Cisco IP Phones fail because the ASA does not support NAT or PAT for the file content transferred over TFTP. Although the ASA supports NAT of TFTP messages and opens a pinhole for the TFTP file, the ASA cannot translate the Cisco CallManager IP address and port embedded in the Cisco IP Phone configuration files that are transferred by TFTP during phone registration.



Note The ASA supports stateful failover of SCCP calls except for calls that are in the middle of call setup.

Default SCCP Inspection

SCCP inspection is enabled by default using these defaults:

- Registration: Not enforced.
- Maximum message ID: 0x181.
- Minimum prefix length: 4
- Media timeout: 00:05:00
- Signaling timeout: 01:00:00.
- RTP conformance: Not enforced.

Configure a Skinny (SCCP) Inspection Policy Map

To specify actions when a message violates a parameter, create an SCCP inspection policy map. You can then apply the inspection policy map when you enable SCCP inspection.

Procedure

Step 1 Create an SCCP inspection policy map: **policy-map type inspect skinny** *policy_map_name*

Where the *policy_map_name* is the name of the policy map. The CLI enters policy-map configuration mode.

Step 2 (Optional) Add a description to the policy map: **description** *string*

Step 3 (Optional) Drop traffic based on the station message ID field in SCCP messages.

a) Identify the traffic based on the station message ID value in hexadecimal, from 0x0 to 0xffff. You can either specify a single ID, or a range of IDs, using the **match** [**not**] **message-id** command. If you use a **match not** command, then any traffic that does not match the criterion in the **match not** command has the action applied.

match message-id {*value* | **range** *start_value end_value*}

Example:

```
hostname(config-pmap)# match message-id 0x181
hostname(config-pmap)# match message-id range 0x200 0xffff
```

b) Specify the action to perform on matching packets. You can drop the packet and optionally log it: **drop** [**log**]

c) Repeat the process until you identify all message IDs that you want to drop.

Step 4 Configure parameters that affect the inspection engine.

a) Enter parameters configuration mode.

```
hostname(config-pmap)# parameters
hostname(config-pmap-p)#
```

b) Set one or more parameters. You can set the following options; use the **no** form of the command to disable the option:

- **enforce-registration**—Enforces registration before calls can be placed.
- **message-ID max** *hex_value*—Sets the maximum SCCP station message ID allowed. The message ID is in hex, and the default maximum is 0x181.
- **rtp-conformance** [**enforce-payloadtype**]—Checks RTP packets flowing on the pinholes for protocol conformance. The optional enforce-payloadtype keyword enforces the payload type to be audio or video based on the signaling exchange.
- **sccp-prefix-len** {**max** | **min**} *length*—Sets the maximum or minimum SCCP prefix length value allowed. Enter the command twice to set both a minimum and maximum value. The default minimum is 4, there is no default maximum.
- **timeout** {**media** | **signaling**} *time*—Sets the timeouts for media and signaling connections (in hh:mm:ss format). To have no timeout, specify 0 for the number. The default media timeout is 5 minutes, the default signaling timeout is one hour.

Example

The following example shows how to define an SCCP inspection policy map.

```
hostname(config)# policy-map type inspect skinny skinny-map
hostname(config-pmap)# parameters
hostname(config-pmap-p)# enforce-registration
hostname(config-pmap-p)# match message-id range 200 300
hostname(config-pmap-p)# drop log
hostname(config)# class-map inspection_default
hostname(config-cmap)# match default-inspection-traffic
hostname(config)# policy-map global_policy
hostname(config-pmap)# class inspection_default
hostname(config-pmap-c)# inspect skinny skinny-map
hostname(config)# service-policy global_policy global
```

What to do next

You can now configure an inspection policy to use the map. See Configure Application Layer Protocol Inspection.

STUN Inspection

Session Traversal Utilities for NAT (STUN), defined in RFC 5389, is used by WebRTC clients for browser-based real-time communications so that plug-ins are not necessary. WebRTC clients often use cloud STUN servers to learn their public IP addresses and ports. WebRTC uses Interactive Connectivity Establishment (ICE, RFC 5245) to verify connectivity between clients. These clients typically use UDP, although they can also use TCP or other protocols.

Because firewalls often block outgoing UDP traffic, WebRTC products such as Cisco Spark can have problems completing connections. STUN inspection opens pinholes for STUN endpoints, and enforces basic STUN and ICE compliance, to allow communications for clients if the connectivity check is acknowledged by both sides. Thus, you can avoid opening new ports in your access rules to enable these applications.

When you enable STUN inspection on the default inspection class, TCP/UDP port 3478 is watched for STUN traffic. The inspection supports IPv4 addresses and TCP/UDP only.

There are some NAT limitations for STUN inspection. For WebRTC traffic, static NAT/PAT44 are supported. Cisco Spark can support additional types of NAT, because Spark does not require pinholes. You can also use NAT/PAT64, including dynamic NAT/PAT, with Cisco Spark.

STUN inspection is supported in failover and cluster modes, as pinholes are replicated. However, the transaction ID is not replicated among nodes. In the case where a node fails after receiving a STUN Request and another node received the STUN Response, the STUN Response will be dropped.



Note STUN inspection uses transaction IDs to match requests and responses. If you use debug to troubleshoot connection drops, note that the system changes the format (endianness) of the IDs for the debug output, so they do not compare directly to those you might see in a pcap.

For information on enabling STUN inspection, see Configure Application Layer Protocol Inspection.

History for Voice and Video Protocol Inspection

Feature Name	Releases	Feature Information
SIP, SCCP, and TLS Proxy support for IPv6	9.3(1)	You can now inspect IPv6 traffic when using SIP, SCCP, and TLS Proxy (using SIP or SCCP). We did not modify any commands.
SIP support for Trust Verification Services, NAT66, CUCM 10.5, and model 8831 phones.	9.3(2)	You can now configure Trust Verification Services servers in SIP inspection. You can also use NAT66. SIP inspection has been tested with CUCM 10.5. We added the **trust-verification-server** parameter command.

Feature Name	Releases	Feature Information
Improved SIP inspection performance on multiple core ASA.	9.4(1)	If you have multiple SIP signaling flows going through an ASA with multiple cores, SIP inspection performance has been improved. However, you will not see improved performance if you are using a TLS, phone, or IME proxy. We did not modify any commands.
SIP inspection support in ASA clustering	9.4(1)	You can now configure SIP inspection on the ASA cluster. A control flow can be created on any unit (due to load balancing), but its child data flows must reside on the same unit. TLS Proxy configuration is not supported. We introduced the following command: **show cluster service-policy**.
SIP inspection support for Phone Proxy and UC-IME Proxy was removed.	9.4(1)	You can no longer use Phone Proxy or UC-IME Proxy when configuring SIP inspection. Use TLS Proxy to inspect encrypted traffic. We removed the following commands: **phone-proxy**, **uc-ime**. We removed the **phone-proxy** and **uc-ime** keywords from the **inspect sip** command.
H.323 inspection support for the H.255 FACILITY message coming before the H.225 SETUP message for H.460.18 compatibility.	9.6(1)	You can now configure an H.323 inspection policy map to allow for H.225 FACILITY messages to come before the H.225 SETUP message, which can happen when endpoints comply with H.460.18. We introduced the following command: **early-message**.
Session Traversal Utilities for NAT (STUN) inspection.	9.6(2)	You can now inspect STUN traffic for WebRTC applications including Cisco Spark. Inspection opens pinholes required for return traffic. We added or modified the following commands: **inspect stun**, **show asp drop**, **show conn detail**, **show service-policy inspect stun**.
Support for TLSv1.2 in TLS proxy and Cisco Unified Communications Manager 10.5.2.	9.7(1)	You can now use TLSv1.2 with TLS proxy for encrypted SIP or SCCP inspection with the Cisco Unified Communications Manager 10.5.2. The TLS proxy supports the additional TLSv1.2 cipher suites added as part of the **client cipher-suite** command. We modified the following commands: **client cipher-suite**.
TLS proxy deprecated for SCCP (Skinny) inspection.	9.13(1)	The **tls-proxy** keyword, and support for SCCP/Skinny encrypted inspection, was deprecated. The keyword will be removed from the **inspect skinny** command in a future release.
TLS proxy support eliminated for SCCP (Skinny) inspection.	9.14(1)	The **tls-proxy** keyword, and support for SCCP/Skinny encrypted inspection, was removed.

Feature Name	Releases	Feature Information
The default SIP inspection policy map drops non-SIP traffic.	9.16(1)	For SIP-inspected traffic, the default is now to drop non-SIP traffic. The previous default was to allow non-SIP traffic on ports inspected for SIP. We changed the default SIP policy map to include the **no traffic-non-sip** command.