Getting Started with Application Layer Protocol Inspection

The following topics describe how to configure application layer protocol inspection.

Application Layer Protocol Inspection

Inspection engines are required for services that embed IP addressing information in the user data packet or that open secondary channels on dynamically assigned ports. These protocols require the ASA to do a deep packet inspection instead of passing the packet through the fast path. As a result, inspection engines can affect overall throughput. Several common inspection engines are enabled on the ASA by default, but you might need to enable others depending on your network.

The following topics explain application inspection in more detail.

When to Use Application Protocol Inspection

When a user establishes a connection, the ASA checks the packet against ACLs, creates an address translation, and creates an entry for the session in the fast path, so that further packets can bypass time-consuming checks. However, the fast path relies on predictable port numbers and does not perform address translations inside a packet.

Many protocols open secondary TCP or UDP ports. The initial session on a well-known port is used to negotiate dynamically assigned port numbers.

Other applications embed an IP address in the packet that needs to match the source address that is normally translated when it goes through the ASA.

If you use applications like these, then you need to enable application inspection.

When you enable application inspection for a service that embeds IP addresses, the ASA translates embedded addresses and updates any checksum or other fields that are affected by the translation.

When you enable application inspection for a service that uses dynamically assigned ports, the ASA monitors sessions to identify the dynamic port assignments, and permits data exchange on these ports for the duration of the specific session.

Inspection Policy Maps

You can configure special actions for many application inspections using an *inspection policy map*. These maps are optional: you can enable inspection for a protocol that supports inspection policy maps without configuring a map. These maps are needed only if you want something other than the default inspection actions.

An inspection policy map consists of one or more of the following elements. The exact options available for an inspection policy map depends on the application.

- Traffic matching criteria—You match application traffic to criteria specific to the application, such as a URL string, for which you then enable actions.

  For some traffic matching criteria, you use regular expressions to match text inside a packet. Be sure to create and test the regular expressions before you configure the policy map, either singly or grouped together in a regular expression class map.

- Inspection class map—Some inspection policy maps let you use an inspection class map to include multiple traffic matching criteria. You then identify the inspection class map in the inspection policy map and enable actions for the class as a whole. The difference between creating a class map and defining the traffic match directly in the inspection policy map is that you can create more complex match criteria and you can reuse class maps. However, you cannot set different actions for different matches.

- Parameters—Parameters affect the behavior of the inspection engine.

The following topics provide more details.

Replacing an In-Use Inspection Policy Map

If you have an inspection enabled with a policy map in a service policy, replacing the policy map is a two-step process. First, you must remove the inspection from the service policy and apply changes. Then, you add it back, select the new policy map name, and again apply changes.

How Multiple Traffic Classes are Handled

You can specify multiple inspection class maps or direct matches in the inspection policy map.

If a packet matches multiple different classes or direct matches, then the order in which the ASA applies the actions is determined by internal ASA rules, and not by the order they are added to the inspection policy map. The internal rules are determined by the application type and the logical progression of parsing a packet, and are not user-configurable. For example for HTTP traffic, parsing a Request Method field precedes parsing the Header Host Length field; an action for the Request Method field occurs before the action for the Header Host Length field.

If an action drops a packet, then no further actions are performed in the inspection policy map. For example, if the first action is to reset the connection, then it will never match any further match criteria. If the first action is to log the packet, then a second action, such as resetting the connection, can occur.

If a packet matches multiple match criteria that are the same, then they are matched in the order they appear in the policy map.

A class map is determined to be the same type as another class map or direct match based on the lowest priority match option in the class map (the priority is based on the internal rules). If a class map has the same type of lowest priority match option as another class map, then the class maps are matched according to the order they are added to the policy map. If the lowest priority match for each class map is different, then the class map with the higher priority match option is matched first.

Guidelines for Application Inspection

Failover

State information for multimedia sessions that require inspection are not passed over the state link for stateful failover. The exceptions are GTP and SIP, which are replicated over the state link.

Clustering

The following inspections are not supported in clustering:

- CTIQBE
- H323, H225, and RAS
- IPsec passthrough
- MGCP
- MMP
- RTSP
- SCCP (Skinny)
- WAAS

IPv6

Supports IPv6 for the following inspections:

- DNS over UDP
- FTP
- HTTP
- ICMP
- IPsec pass-through
- IPv6
- SCCP (Skinny)
- SIP
- SMTP
- VXLAN

Supports NAT64 for the following inspections:

- DNS over UDP
- FTP
- HTTP
- ICMP

Additional Guidelines

- Some inspection engines do not support PAT, NAT, outside NAT, or NAT between same security interfaces. For more information about NAT support, see Default Inspections and NAT Limitations, on page 4.
- For all the application inspections, the ASA limits the number of simultaneous, active data connections to 200 connections. For example, if an FTP client opens multiple secondary connections, the FTP inspection engine allows only 200 active connections and the 201 connection is dropped and the adaptive security appliance generates a system error message.
- Inspected protocols are subject to advanced TCP-state tracking, and the TCP state of these connections is not automatically replicated. While these connections are replicated to the standby unit, there is a best-effort attempt to re-establish a TCP state.
- If the system determines that a TCP connection requires inspection, the system clears all TCP options except for the MSS and selective-acknowledgment (SACK) options on the packets before inspecting them. Other options are cleared even if you allow them in a TCP map applied to the connections.
- TCP/UDP Traffic directed to the ASA (to an interface) is inspected by default. However, ICMP traffic directed to an interface is never inspected, even if you enable ICMP inspection. Thus, a ping (echo request) to an interface can fail under specific circumstances, such as when the echo request comes from a source that the ASA can reach through a backup default route.

Defaults for Application Inspection

The following topics explain the default operations for application inspection.

Default Inspections and NAT Limitations

By default, the configuration includes a policy that matches all default application inspection traffic and applies inspection to the traffic on all interfaces (a global policy). Default application inspection traffic includes traffic to the default ports for each protocol. You can only apply one global policy, so if you want to alter the global policy, for example, to apply inspection to non-standard ports, or to add inspections that are not enabled by default, you need to either edit the default policy or disable it and apply a new one.

The following table lists all inspections supported, the default ports used in the default class map, and the inspection engines that are on by default, shown in bold. This table also notes any NAT limitations. In this table:

- Inspection engines that are enabled by default for the default port are in bold.
- The ASA is in compliance with the indicated 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 ASA does not enforce the order.

Table 1: Supported Application Inspection Engines

Application	Default Protocol, Port	NAT Limitations	Standards	Comments
CTIQBE	TCP/2748	No extended PAT. No NAT64. (Clustering) No static PAT.	—	—
DCERPC	TCP/135	No NAT64.	—	—
DNS over UDP	UDP/53	No NAT support is available for name resolution through WINS.	RFC 1123	—
FTP	TCP/21	(Clustering) No static PAT.	RFC 959	—
GTP	UDP/3386 (GTPv0) UDP/2123 (GTPv1+)	No extended PAT. No NAT.	—	Requires the GTP/GPRS license.
H.323 H.225 and **RAS**	TCP/1720 UDP/1718 UDP (RAS) 1718-1719	(Clustering) No static PAT. No extended PAT. No NAT on same security interfaces. No NAT64.	ITU-T H.323, H.245, H225.0, Q.931, Q.932	—
HTTP	TCP/80	—	RFC 2616	Beware of MTU limitations stripping ActiveX and Java. If the MTU is too small to allow the Java or ActiveX tag to be included in one packet, stripping may not occur.
ICMP	ICMP	—	—	ICMP traffic directed to an ASA interface is never inspected.
ICMP ERROR	ICMP	—	—	—
ILS (LDAP)	TCP/389	No extended PAT. No NAT64.	—	—
Instant Messaging (IM)	Varies by client	No extended PAT. No NAT64.	RFC 3860	—
IP Options	RSVP	No NAT64.	RFC 791, RFC 2113	—
IPsec Pass Through	UDP/500	No PAT. No NAT64.	—	—
IPv6	—	No NAT64.	RFC 2460	—

Application	Default Protocol, Port	NAT Limitations	Standards	Comments
MGCP	UDP/2427, 2727	No extended PAT. No NAT64. (Clustering) No static PAT.	RFC 2705bis-05	—
MMP	TCP/5443	No extended PAT. No NAT64.	—	—
NetBIOS Name Server over IP	UDP/137, 138 (Source ports)	No extended PAT. No NAT64.	—	NetBIOS is supported by performing NAT of the packets for NBNS UDP port 137 and NBDS UDP port 138.
PPTP	TCP/1723	No NAT64. (Clustering) No static PAT.	RFC 2637	—
RADIUS Accounting	UDP/1646	No NAT64.	RFC 2865	—
RSH	TCP/514	No PAT. No NAT64. (Clustering) No static PAT.	Berkeley UNIX	—
RTSP	TCP/554	No extended PAT. No NAT64. (Clustering) No static PAT.	RFC 2326, 2327, 1889	No handling for HTTP cloaking.
SIP	TCP/5060 UDP/5060	No NAT/PAT on interfaces with the same, or lower to higher, security levels. No extended PAT. No NAT64 or NAT46. (Clustering) No static PAT.	RFC 2543	Does not handle TFTP uploaded Cisco IP Phone configurations under certain circumstances.
SKINNY (SCCP)	TCP/2000	No NAT on same security interfaces. No extended PAT. No NAT64, NAT46, or NAT66. (Clustering) No static PAT.	—	Does not handle TFTP uploaded Cisco IP Phone configurations under certain circumstances.
SMTP and **ESMTP**	TCP/25	No NAT64.	RFC 821, 1123	—
SNMP	UDP/161, 162	No NAT or PAT.	RFC 1155, 1157, 1212, 1213, 1215	v.2 RFC 1902-1908; v.3 RFC 2570-2580.

Application	Default Protocol, Port	NAT Limitations	Standards	Comments
SQL*Net	TCP/1521	No extended PAT. No NAT64. (Clustering) No static PAT.	—	v.1 and v.2.
Sun RPC	TCP/111 UDP/111	No extended PAT. No NAT64.	—	—
TFTP	UDP/69	No NAT64. (Clustering) No static PAT.	RFC 1350	Payload IP addresses are not translated.
WAAS	TCP/1- 65535	No extended PAT. No NAT64.	—	—
XDMCP	UDP/177	No extended PAT. No NAT64. (Clustering) No static PAT.	—	—
VXLAN	UDP/4789	Not applicable	RFC 7348	Virtual Extensible Local Area Network.

Default Inspection Policy Maps

Some inspection types use hidden default policy maps. For example, if you enable ESMTP inspection without specifying a map, _default_esmtp_map is used.

The default inspection is described in the sections that explain each inspection type. You can view these default maps using the **show running-config all policy-map** command; use **Tools > Command Line Interface**.

DNS inspection is the only one that uses an explicitly-configured default map, preset_dns_map.

Configure Application Layer Protocol Inspection

You configure application inspection in service policies.

Inspection is enabled by default globally on all interfaces for some applications on their standard ports and protocols. See Default Inspections and NAT Limitations, on page 4 for more information on default inspections. A common method for customizing the inspection configuration is to customize the default global policy. You can alternatively create a new service policy as desired, for example, an interface-specific policy.

Before you begin

For some applications, you can perform special actions when you enable inspection by configuring inspection policy maps. The table later in this procedure shows which protocols allow inspection policy maps, with pointers to the instructions on configuring them. If you want to configure these advanced features, create the map before configuring inspection.

Procedure

Step 1 Choose **Configuration** > **Firewall** > **Service Policy Rules**.

Step 2 Open a rule.

- To edit the default global policy, select the "inspection_default" rule in the Global folder and click **Edit**.
- To create a new rule, click **Add > Add Service Policy Rule**. Proceed through the wizard to the Rules page.
- If you have another inspection rule, or a rule to which you are adding an inspection, select it and click **Edit**.

If you want to match non-standard ports, then create a new rule for the non-standard ports. See Default Inspections and NAT Limitations, on page 4 for the standard ports for each inspection engine.

You can combine multiple rules in the same service policy if desired, so you can create one rule to match certain traffic, and another to match different traffic. However, if traffic matches a rule that contains an inspection action, and then matches another rule that also has an inspection action, only the first matching rule is used.

If you are implementing RADIUS accounting inspection, create a management service policy rule instead. See Configure RADIUS Accounting Inspection.

Step 3 On the Rule Actions wizard page or tab, select the **Protocol Inspection** tab.

Step 4 (To change an in-use policy) If you are editing any in-use policy to use a different inspection policy map, you must disable the inspection, and then re-enable it with the new inspection policy map name:

a) Uncheck the protocol's check box.
b) Click **OK**.
c) Click **Apply**.
d) Repeat these steps to return to the **Protocol Inspections** tab.

Step 5 Select the inspection type that you want to apply.

You can select multiple options on the default inspection traffic class only.

Some inspection engines let you control additional parameters when you apply the inspection to the traffic. Click **Configure** for the inspection type to configure an inspection policy map and other options. You can either choose an existing map, or create a new one. You can predefine inspection policy maps from the **Configuration > Firewall > Objects > Inspect Maps** list.

The following table lists the protocols you can inspect, whether they allow inspection policy maps or inspection class maps, and a pointer to detailed information about the inspection.

Table 2: Inspection Protocols

Protocol	Supports Inspection Policy Maps	Supports Inspection Class Maps	Notes
CTIQBE	No	No	See CTIQBE Inspection.
DCERPC	Yes	No	See DCERPC Inspection.

Protocol	Supports Inspection Policy Maps	Supports Inspection Class Maps	Notes
DNS	Yes	Yes	See DNS Inspection. If you are using the Botnet Traffic Filter, choose **Enable DNS snooping**. We suggest that you enable DNS snooping only on interfaces where external DNS requests are going. Enabling DNS snooping on all UDP DNS traffic, including that going to an internal DNS server, creates unnecessary load on the ASA. For example, if the DNS server is on the outside interface, you should enable DNS inspection with snooping for all UDP DNS traffic on the outside interface.
ESMTP	Yes	No	See SMTP and Extended SMTP Inspection.
FTP	Yes	Yes	See FTP Inspection. Select **Use Strict FTP** to select an inspection policy map. Strict FTP increases the security of protected networks by preventing web browsers from sending embedded commands in FTP requests.
GTP	Yes	No	See GTP Inspection Overview.
H.323 H.225	Yes	Yes	See H.323 Inspection.
H.323 RAS	Yes	Yes	See H.323 Inspection.
HTTP	Yes	Yes	See HTTP Inspection.
ICMP	No	No	See ICMP Inspection.
ICMP Error	No	No	See ICMP Error Inspection.
ILS	No	No	See ILS Inspection.
IM	Yes	Yes	See Instant Messaging Inspection.
IP-Options	Yes	No	See IP Options Inspection.
IPSec Pass Thru	Yes	No	See IPsec Pass Through Inspection.
IPv6	Yes	No	See IPv6 Inspection.
MGCP	Yes	No	See MGCP Inspection.
NetBIOS	Yes	No	See NetBIOS Inspection.
PPTP	No	No	See PPTP Inspection.

Protocol	Supports Inspection Policy Maps	Supports Inspection Class Maps	Notes
RADIUS Accounting	Yes	No	See RADIUS Accounting Inspection Overview. RADIUS accounting inspection is available for a management service policy only. You must select a policy map to implement this inspection.
RSH	No	No	See RSH Inspection.
RTSP	Yes	No	See RTSP Inspection.
SCCP (Skinny)	Yes	No	See Skinny (SCCP) Inspection. If you want to inspect encrypted SCCP traffic, choose **Enable encrypted traffic inspection** and select a TLS proxy (click **Manage** to create one if necessary).
SIP	Yes	Yes	See SIP Inspection. If you want to inspect encrypted SIP traffic, choose **Enable encrypted traffic inspection** and select a TLS proxy (click **Manage** to create one if necessary).
SNMP	Yes	No	See SNMP Inspection.
SQLNET	No	No	See SQL*Net Inspection.
SUNRPC	No	No	See Sun RPC Inspection. The default class map includes UDP port 111; if you want to enable Sun RPC inspection for TCP port 111, you need to create a new class map that matches TCP port 111, add the class to the policy, and then apply SUNRPC inspection to that class.
TFTP	No	No	See TFTP Inspection.
WAAS	No	No	Enables TCP option 33 parsing. Use when deploying Cisco Wide Area Application Services products.
XDMCP	No	No	See XDMCP Inspection.
VXLAN	No	No	See VXLAN Inspection.

Step 6 Click **OK** or **Finish** to save the service policy rule.

Configure Regular Expressions

Regular expressions define pattern matching for text strings. You can use these expressions in some protocol inspection maps to match packets based on strings such as URLs or the contents of particular header fields.

Create a Regular Expression

A regular expression matches text strings either literally as an exact string, or by using *metacharacters* so that you can match multiple variants of a text string. You can use a regular expression to match the content of certain application traffic; for example, you can match a URL string inside an HTTP packet.

Before you begin

See the **regex** command in the command reference for performance impact information when matching a regular expression to packets. In general, matching against long input strings, or trying to match a large number of regular expressions, will reduce system performance.

Note As an optimization, the ASA searches on the deobfuscated URL. Deobfuscation compresses multiple forward slashes (/) into a single slash. For strings that commonly use double slashes, like "http://", be sure to search for "http:/" instead.

The following table lists the metacharacters that have special meanings.

Table 3: Regular Expression Metacharacters

Character	Description	Notes
.	Dot	Matches any single character. For example, **d.g** matches dog, dag, dtg, and any word that contains those characters, such as doggonnit.
(*exp*)	Subexpression	A subexpression segregates characters from surrounding characters, so that you can use other metacharacters on the subexpression. For example, **d(o|a)g** matches dog and dag, but **do|ag** matches do and ag. A subexpression can also be used with repeat quantifiers to differentiate the characters meant for repetition. For example, **ab(xy){3}z** matches abxyxyxyz.
|	Alternation	Matches either expression it separates. For example, **dog|cat** matches dog or cat.
?	Question mark	A quantifier that indicates that there are 0 or 1 of the previous expression. For example, **lo?se** matches lse or lose.

Character	Description	Notes
*	Asterisk	A quantifier that indicates that there are 0, 1 or any number of the previous expression. For example, **lo*se** matches lse, lose, loose, and so on.
+	Plus	A quantifier that indicates that there is at least 1 of the previous expression. For example, **lo+se** matches lose and loose, but not lse.
{*x*} or {*x*,}	Minimum repeat quantifier	Repeat at least *x* times. For example, **ab(xy){2,}z** matches abxyxyz, abxyxyxyz, and so on.
[*abc*]	Character class	Matches any character in the brackets. For example, **[abc]** matches a, b, or c.
[^*abc*]	Negated character class	Matches a single character that is not contained within the brackets. For example, **[^abc]** matches any character other than a, b, or c. **[^A-Z]** matches any single character that is not an uppercase letter.
[*a-c*]	Character range class	Matches any character in the range. **[a-z]** matches any lowercase letter. You can mix characters and ranges: **[abcq-z]** matches a, b, c, q, r, s, t, u, v, w, x, y, z, and so does [**a-cq-z]**. The dash (-) character is literal only if it is the last or the first character within the brackets: **[abc-]** or **[-abc]**.
“”	Quotation marks	Preserves trailing or leading spaces in the string. For example, **“ test”** preserves the leading space when it looks for a match.
^	Caret	Specifies the beginning of a line.
\	Escape character	When used with a metacharacter, matches a literal character. For example, **\[** matches the left square bracket.
char	Character	When character is not a metacharacter, matches the literal character.
\r	Carriage return	Matches a carriage return 0x0d.
\n	Newline	Matches a new line 0x0a.
\t	Tab	Matches a tab 0x09.
\f	Formfeed	Matches a form feed 0x0c.
\x*NN*	Escaped hexadecimal number	Matches an ASCII character using hexadecimal (exactly two digits).

Character	Description	Notes
NNN	Escaped octal number	Matches an ASCII character as octal (exactly three digits). For example, the character 040 represents a space.

Procedure

Step 1 Choose **Configuration** > **Firewall** > **Objects** > **Regular Expressions**.

Step 2 In the Regular Expressions area, do one of the following:

- Choose **Add** to add a new object. Enter a name and optionally, a description.
- Choose an existing object and click **Edit**.

Step 3 Either enter the regular expression in the **Value** field, or click **Build** to get help creating the expression.

The regular expression is limited to 100 characters in length.

If you click **Build**, use the following process to create the expression:

a) In the Build Snippet area, create a component of the expression using the following options. Look at the Snippet Preview area at the end of this section to see the expression you are building.

- Starts at the beginning of the line (^)—Indicates that the snippet should start at the beginning of a line, using the caret (^) metacharacter. Be sure to insert any snippet with this option at the beginning of the regular expression.
- Specify Character String—If you are trying to match a specific string, such as a word or phrase, enter the string.

  If there are any metacharacters in your text string that you want to be used literally, choose **Escape Special Characters** to add the backslash (\) escape character before them. for example, if you enter "example.com," this option converts it to "example\.com".

  If you want to match upper and lower case characters, choose **Ignore Case**. For example, "cats" is converted to "[cC][aA][tT][sS]".
- Specify Character—If you are trying to match a specific type of character or set of characters, rather than a particular phrase, select this option and identify the characters using these options:
  - **Negate the character**—Specifies not to match the character you identify.
  - **Any character (.)**—Inserts the period (.) metacharacter to match any character. For example, **d.g** matches dog, dag, dtg, and any word that contains those characters, such as doggonnit.
  - **Character set**—Inserts a character set. Text can match any character in the set. For example, if you specify [0-9A-Za-z], then this snippet will match any character from A to Z (upper or lower case) or any digit 0 through 9. The [\n\f\r\t] set matches a new line, form feed, carriage return, or a tab.
  - **Special character**—Inserts a character that requires an escape, including \, ?, *, +, |, ., [, (, or ^. The escape character is the backslash (\), which is automatically entered when you choose this option.

- **Whitespace character**—Whitespace characters include \n (new line), \f (form feed), \r (carriage return), or \t (tab).
- **Three digit octal number**—Matches an ASCII character as octal (up to three digits). For example, the character \040 represents a space. The backslash (\) is entered automatically.
- **Two digit hexadecimal number**—Matches an ASCII character using hexadecimal (exactly two digits). The backslash (\) is entered automatically.
- **Specified character**—Enter any single character.

b) Add the snippet to the regular expression box using one of the following buttons. Note that you can also type directly in the regular expression.
   - **Append Snippet**—Adds the snippet to the end of the regular expression.
   - **Append Snippet as Alternate**—Adds the snippet to the end of the regular expression separated by a pipe (|), which matches either expression it separates. For example, **dog|cat** matches dog or cat.
   - **Insert Snippet at Cursor**—Inserts the snippet at the cursor.

c) Repeat the process to add snippets until the expression is complete.

d) (Optional.) In **Selection Occurrences**, select how often the expression or parts of it must match text to be considered a match. Select text in the Regular Expression field, click one of the following options, and then click **Apply to Selection**. For example, if the regular expression is "test me," and you select "me" and apply **One or more times**, then the regular expression changes to "test (me)+".
   - **Zero or one times (?)**—There are 0 or 1 of the previous expression. For example, **lo?se** matches lse or lose.
   - **One or more times (+)**—There is at least 1 of the previous expression. For example, **lo+se** matches lose and loose, but not lse.
   - **Any number of times (*)**—There are 0, 1 or any number of the previous expression. For example, **lo*se** matches lse, lose, loose, and so on.
   - **At least**—Repeat at least *x* times. For example, **ab(xy){2,}z** matches abxyxyz, abxyxyxyz, and so on.
   - **Exactly**—Repeat exactly *x* times. For example, **ab(xy){3}z** matches abxyxyxyz.

e) Click **Test** to verify your expression will match the intended text. If the test is unsuccessful, you can try editing it in the test dialog, or return to the expression builder. If you edit the expression in the text dialog and click **OK**, the edits are saved and reflected in the expression builder.

f) Click **OK**.

Create a Regular Expression Class Map

A regular expression class map identifies one or more regular expression. It is simply a collection of regular expression objects. You can use a regular expression class map in many cases in replace of a regular expression object.

Procedure

Step 1 Choose **Configuration** > **Firewall** > **Objects** > **Regular Expressions**.

Step 2 In the Regular Expressions Classes area, do one of the following:

- Choose **Add** to add a new class map. Enter a name and optionally, a description.
- Choose an existing class map and click **Edit**.

Step 3 Select the expressions you want in the map and click **Add**. Remove any you do not want.

Step 4 Click **OK**.

Monitoring Inspection Policies

To monitor inspection service policies, enter the following commands. Select **Tools** > **Command Line Interface** to enter these commands. See the command reference on Cisco.com for detailed syntax and examples.

- **show service-policy inspect** *protocol*

  Displays statistics for inspection service policies. The *protocol* is the protocol from the inspect command, for example **dns**. However, not all inspection protocols show statistics with this command. For example:

```
asa# show service-policy inspect dns

Global policy:
  Service-policy: global_policy
    Class-map: inspection_default
      Inspect: dns preset_dns_map, packet 0, lock fail 0, drop 0, reset-drop 0,
5-min-pkt-rate 0 pkts/sec, v6-fail-close 0
        message-length maximum client auto, drop 0
        message-length maximum 512, drop 0
        dns-guard, count 0
        protocol-enforcement, drop 0
        nat-rewrite, count 0
asa#
```

- **show conn**

  Shows current connections for traffic passing through the device. This command has a wide range of keywords so that you can get information about various protocols.

- Additional commands for specific inspected protocols:
  - **show ctiqbe**

    Displays information about the media connections allocated by the CTIQBE inspection engine

  - **show h225**

    Displays information for H.225 sessions.

  - **show h245**

    Displays information for H.245 sessions established by endpoints using slow start.

- **show h323 ras**

  Displays connection information for H.323 RAS sessions established between a gatekeeper and its H.323 endpoint.

- **show mgcp** {**commands** | **sessions** }

  Displays the number of MGCP commands in the command queue or the number of existing MGCP sessions.

- **show sip**

  Displays information for SIP sessions.

- **show skinny**

  Displays information for Skinny (SCCP) sessions.

- **show sunrpc-server active**

  Displays the pinholes opened for Sun RPC services.

History for Application Inspection

Feature Name	Releases	Description
Inspection policy maps	7.2(1)	The inspection policy map was introduced. The following command was introduced: **class-map type inspect**.
Regular expressions and policy maps	7.2(1)	Regular expressions and policy maps were introduced to be used under inspection policy maps. The following commands were introduced: **class-map type regex**, **regex**, **match regex**.
Match any for inspection policy maps	8.0(2)	The **match any** keyword was introduced for use with inspection policy maps: traffic can match one or more criteria to match the class map. Formerly, only **match all** was available.