



# Firewall Support for SIP

The Firewall Support for SIP feature integrates Cisco IOS firewalls, Voice over IP (VoIP) protocol, and Session Initiation Protocol (SIP) within a Cisco IOS-based platform, enabling better network convergence.



**Note**

Some Cisco IOS versions earlier than 12.2(11)YU and 12.2(15)T may accept the configuration commands for SIP that are shown in this document; however, those earlier versions will not function properly.

## Feature Specifications for Firewall Support for SIP

### Feature History

Release	Modification
12.2(11)YU	This feature was introduced.
12.2(15)T	This feature was integrated into Cisco IOS Release 12.2(15)T.

### Supported Platforms

For platforms supported in Cisco IOS Releases 12.2(11)YU and 12.2(15)T, consult Cisco Feature Navigator.

## Finding Support Information for Platforms and Cisco IOS Software Images

Use Cisco Feature Navigator to find information about platform support and Cisco IOS software image support. Access Cisco Feature Navigator at <http://www.cisco.com/go/fn>. You must have an account on Cisco.com. If you do not have an account or have forgotten your username or password, click **Cancel** at the login dialog box and follow the instructions that appear.

## Contents

- [Restrictions for Firewall Support for SIP, page 2](#)
- [Information About Firewall Support for SIP, page 2](#)
- [How to Configure Your Firewall for SIP, page 8](#)
- [Configuration Examples for Firewall SIP Support, page 11](#)
- [Additional References, page 11](#)
- [Command Reference, page 13](#)

# Restrictions for Firewall Support for SIP

## DNS Name Resolution

Although SIP methods can have Domain Name System (DNS) names instead of raw IP addresses, this feature currently does not support DNS names.

## SIP UDP Support Only

This feature supports only the SIP User Datagram Protocol (UDP) format for signaling; the TCP format is not supported.

## SIP Abbreviated Header

This feature does not support the compact form of SIP header fields.

## Earlier Versions of Cisco IOS

Some Cisco IOS versions earlier than 12.2(11)YU and 12.2(15)T may accept the configuration commands for SIP that are shown in this document; however, those earlier versions will not function properly.

# Information About Firewall Support for SIP

To configure the Cisco IOS Firewall Support for SIP feature, you must understand the following concepts:

- [Firewall and SIP Overviews, page 2](#)
- [Firewall for SIP Functionality Description, page 5](#)
- [SIP Message Treatment by the Firewall, page 6](#)
- [Call Database, page 7](#)

## Firewall and SIP Overviews

This section contains the following concepts:

- [Cisco IOS Firewall, page 2](#)
- [SIP \(Session Initiation Protocol\), page 3](#)

## Cisco IOS Firewall

The Cisco IOS firewall extends the concept of static access control lists (ACLs) by introducing dynamic ACL entries that open on the basis of the necessary application ports on a specific application and close these ports at the end of the application session. The Cisco IOS firewall achieves this functionality by inspecting the application data, checking for conformance of the application protocol, extracting the relevant port information to create the dynamic ACL entries, and closing these ports at the end of the session. The Cisco IOS firewall is designed to easily allow a new application inspection whenever support is needed.

## SIP (Session Initiation Protocol)

SIP is an ASCII-based, application-layer control protocol that can be used to establish, maintain, and terminate calls between two or more endpoints. Like other VoIP protocols, SIP is designed to address the functions of signaling and session management within a packet telephony network. Signaling allows call information to be carried across network boundaries. Session management provides the ability to control the attributes of an end-to-end call.

### SIP Messages

SIP has two types of messages—requests and responses—that have the following generic structure:

```
generic-message = Request-Line | Status-Line
                  * ( general-header | request-header
                    | response-header | entity-header )
                  CRLF
                  [ message-body]
```



#### Note

Any of these message components may contain embedded IP addresses.

[Table 1](#) identifies the six available SIP request messages.

**Table 1** SIP Request Messages

SIP Message	Purpose
ACK	Confirms receipt of a final response to INVITE
BYE	Is sent by either side to end the call
CANCEL	Is sent to end a call that has not yet been connected
INVITE	Is a request from a User Agent Client (UAC) to initiate a session
OPTIONS	Are sent to query capabilities of the user agents and network servers
REGISTER	Is sent by the client to register the address with a SIP proxy

[Table 2](#) identifies the available SIP response methods.

**Table 2** SIP Response Messages

SIP Message	Purpose
1xx Informational	<ul style="list-style-type: none"> <li>• 100 = Trying</li> <li>• 180 = Ringing</li> <li>• 181 = Call Is Being Forwarded</li> <li>• 182 = Queued</li> <li>• 183 = Session Progress</li> </ul>
2xx Successful	<ul style="list-style-type: none"> <li>• 200 = OK</li> </ul>

**Table 2** *SIP Response Messages (continued)*

<b>SIP Message</b>	<b>Purpose</b>
3xx Redirection	<ul style="list-style-type: none"> <li>• 300 = Multiple Choices</li> <li>• 301 = Moved Permanently</li> <li>• 302 = Moved Temporarily</li> <li>• 303 = See Other</li> <li>• 305 = Use Proxy</li> <li>• 380 = Alternative Service</li> </ul>
4xx Request Failure	<ul style="list-style-type: none"> <li>• 400 = Bad Request</li> <li>• 401 = Unauthorized</li> <li>• 402 = Payment Required</li> <li>• 403 = Forbidden</li> <li>• 404 = Not Found</li> <li>• 405 = Method Not Allowed</li> <li>• 406 = Not Acceptable</li> <li>• 407 = Proxy Authentication Required</li> <li>• 408 = Request Timeout</li> <li>• 409 = Conflict</li> <li>• 410 = Gone</li> <li>• 411 = Length Required</li> <li>• 413 = Request Entity Too Large</li> <li>• 414 = Request URI Too Large</li> <li>• 415 = Unsupported Media Type</li> <li>• 420 = Bad Extension</li> <li>• 480 = Temporarily Not Available</li> <li>• 481 = Call Leg/Transaction Does Not Exist</li> </ul>
4xx Request Failure (continued)	<ul style="list-style-type: none"> <li>• 482 = Loop Detected</li> <li>• 483 = Too Many Hops</li> <li>• 484 = Address Incomplete</li> <li>• 485 = Ambiguous</li> <li>• 486 - Busy Here</li> </ul>

**Table 2** SIP Response Messages (continued)

SIP Message	Purpose
5xx Server Failure	<ul style="list-style-type: none"><li>• 500 = Internal Server Error</li><li>• 501 = Not Implemented</li><li>• 502 = Bad Gateway</li><li>• 503 = Service Unavailable</li><li>• 504 = Gateway Timeout</li><li>• 505 = SIP Version Not Supported</li></ul>
6xx Global Failure	<ul style="list-style-type: none"><li>• 600 = Busy Anywhere</li><li>• 603 = Decline</li><li>• 604 = Does Not Exist Anywhere</li><li>• 606 = Not Acceptable</li></ul>

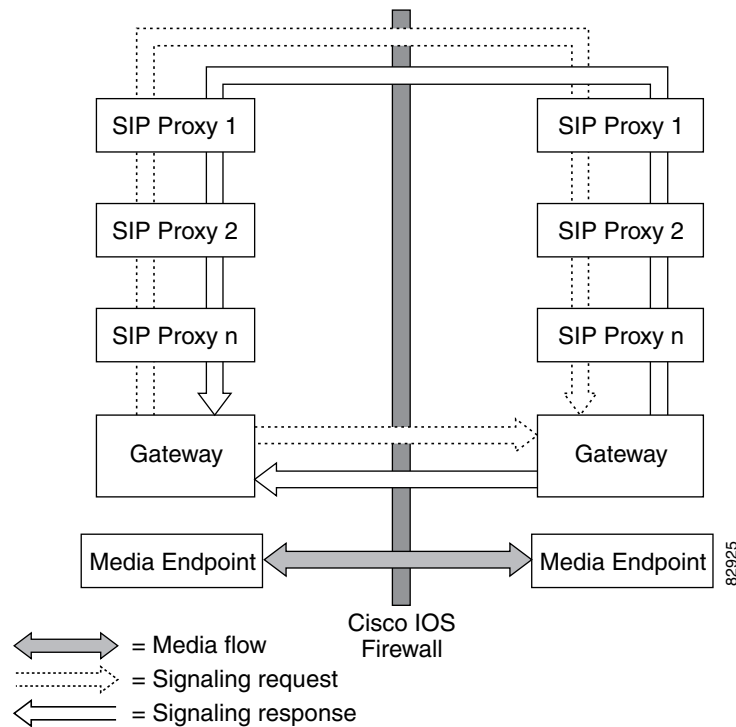
## Firewall for SIP Functionality Description

The Firewall for SIP Support feature allows SIP signaling requests to traverse directly between gateways or through a series of proxies to the destination gateway or phone. After the initial request, if the Record-Route header field is not used, subsequent requests can traverse directly to the destination gateway address as specified in the Contact header field. Thus, the Cisco IOS firewall is aware of all surrounding proxies and gateways and allows the following functionality:

- SIP signaling responses can travel the same path as SIP signaling requests.
- Subsequent signaling requests can travel directly to the endpoint (destination gateway).
- Media endpoints can exchange data between each other.

See [Figure 1](#) for a sample topology that displays these functionalities.

Figure 1 Cisco IOS Firewall for SIP Awareness Sample Topology



## SIP Message Treatment by the Firewall

See [Table 3](#) for information on the treatment of SIP methods by the Cisco IOS firewall.

**Table 3** Treatment of SIP Methods by the Cisco IOS Firewall

SIP Message	Purpose
200 OK	Signifies the end of the call creation phase. The packet is checked for validity against the call database, and the contact information of the server is taken from it. Temporary call-flow-based openings in the firewall are created for allowing the BYE message, which can be initiated from the inside or outside.
200 OK for BYE	Signifies the graceful termination of the call and is in response to the BYE message. The same action as the CANCEL message is taken.
ACK	Signifies that the message is passed after checking for validity.
BYE	Signifies the intent to terminate the call. The database state is updated and temporary openings in the firewall are created for response to the BYE message.
CANCEL	Signifies abnormal data termination. The signaling sessions, media sessions, pregenerated temporary openings in the firewall, and the call database entry for the call are removed.

**Table 3** Treatment of SIP Methods by the Cisco IOS Firewall (continued)

SIP Message	Purpose
INVITE	Occurs typically at the start of the call. The firewall will create a database entry upon receipt of this method and fill the database with relevant information extracted from this message. Temporary openings in the firewall will allow for a series of responses to the INVITE request. The temporary openings will be call-flow sensitive and will allow for responses for a fixed amount of time (t = 30 secs).
NO MATCH	Signifies a signaling message that is not present in the database.
Other Methods	Signifies that the message is passed if the call ID is present in the call database.
REGISTER	Results in the creation of an entry in the call database. Time-based, flow-control ACL firewall openings will allow for the response to the REGISTER and subsequent INVITE messages.
SESSION PROGRESS	Contains a response to the INVITE message, and it is a packet during the call creation phase. The packet is checked against the call database for validity of call ID and the media ports; the server proxy information is gathered from the packet. Media channels should be created in this phase.

## Call Database

A call database, which contains the details of a call leg, is maintained for all call flows. A call database is created and maintained because there can be numerous signaling sessions for each call. [Table 4](#) identifies the information available in the call database.

**Table 4** Call Database Information

Type	Purpose
call_int_over	Checks to see whether or not call initialization is over, and if so, checks to see if the call is in the teardown phase
C con ip & C con port	Signifies the IP address and port in the contact field of the initiator; for example, "Contact:<sip:1111@172.16.0.3:5060;user=phone>"
C media ip & C media port	Signifies the IP address in the media field of the initiator; for example, "c=IN IP4 172.16.0.3"
C media port	Signifies the port in the media field of the initiator; for example, "m=audio 20758 RTP/AVP 0"
C src ip & C src port	Signifies the actual IP address and port of the initiator
C via ip & C via port	Signifies the IP address and port in the via field of the initiator (the first via line); for example, "Via: SIP/2.0/UDP 172.16.0.3:5060"
current sip state	Is the current state of the call (which helps to avoid retransmission)
from/to/callid	Is extracted from the "INVITE" SIP request message to identify the call
media header	Keeps the list of media sessions for the call

**Table 4 Call Database Information (continued)**

Type	Purpose
media opened	Signifies multiple messages that may have media information, so you need to check to see whether or not the media has been opened for the call
prev sip state	Signifies the previous state of the call (which helps to avoid retransmission)
S con ip & S con port	Signifies the IP address and port in the contact field for the responder
S media ip	Signifies the IP address in the media field for the responder
S media port	Signifies the port in the media field for the responder
S src ip & S src port	Signifies the actual IP address and port of the responder
S via ip & S via port	Signifies the IP address and port in the via field for the responder
signal header	Keeps the list of signaling sessions for the call
sip_proxy_traversed	Makes the firewall topologically aware of whether the call has traversed through proxies

## How to Configure Your Firewall for SIP

To configure a Cisco IOS Firewall for SIP support, perform the following tasks:

- [Configuring Firewall for SIP Support, page 8](#) (required)
- [Verifying Firewall for SIP Support, page 9](#) (optional)
- [Monitoring Firewall for SIP Support, page 10](#) (optional)

## Configuring Firewall for SIP Support

To enable a firewall to support SIP, use the following commands.

### Prerequisite

Before you configure Cisco IOS firewall support for SIP on your router, you first need to configure access lists, whose purpose normally is to block SIP traffic from unprotected networks for which the firewall will create temporary openings for specific traffic. For information about configuring access lists and the **access-list** command, see the chapter “[Configuring IPsec Network Security](#)” in the *Cisco IOS Security Configuration Guide*, Release 12.2, and the *Cisco IOS Command Reference*, Release 12.2 T, respectively.

### SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip inspect name** *inspection-name* **sip** [**alert** {**on** | **off**}] [**audit-trail** {**on** | **off**}] [**timeout** *seconds*]
4. **interface** *type number*
5. **ip inspect** *inspection-name* {**in** | **out**}

## 6. Repeat Steps 3 through 5 (Optional)

## DETAILED STEPS

	Command or Action	Purpose
Step 1	<b>enable</b>  <b>Example:</b> Router> enable	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	<b>configure terminal</b>  <b>Example:</b> Router# configure terminal	Enters global configuration mode.
Step 3	<b>ip inspect name inspection-name sip [alert {on   off}] [audit-trail {on   off}] [timeout seconds]</b>  <b>Example:</b> Router(config)# ip inspect name voip sip	Turns on inspection for SIP. <ul style="list-style-type: none"> <li>• <b>alert</b>—Alert messages are generated. This function is <b>on</b> by default.</li> <li>• <b>audit-trail</b>—Audit trail messages are generated. This function is <b>off</b> by default.</li> <li>• <b>timeout</b>—Overrides the global channel inactivity timeout value.</li> </ul>
Step 4	<b>interface type number</b>  <b>Example:</b> Router(config)# interface FastEthernet 0/0	Configures an interface type and enters interface configuration mode.
Step 5	<b>ip inspect inspection-name {in   out}</b>  <b>Example:</b> Router(config-if)# ip inspect voip in	Applies inspection configurations to an interface and for a particular traffic direction.
Step 6	If SIP calls are coming from other interfaces, repeat Steps 3 through 5 and apply SIP inspections for the calls that are coming from those interfaces.	<b>Note</b> The inspection of protocols other than SIP may not be desirable for traffic that comes from external networks, so it may be necessary to configure an additional inspection rule specifying only SIP.

## Verifying Firewall for SIP Support

To verify Cisco IOS firewall session information, perform the following optional steps:

## SUMMARY STEPS

1. **enable**
2. **show ip inspect name inspection-name**
3. **show ip inspect session [detail]**
4. **show ip access-list**

## DETAILED STEPS

	Command or Action	Purpose
Step 1	<code>enable</code>  <b>Example:</b> Router> enable	Enables privileged EXEC mode. <ul style="list-style-type: none"> <li>Enter your password if prompted.</li> </ul>
Step 2	<code>show ip inspect name inspection-name</code>  <b>Example:</b> Router# show ip inspect name voip	(Optional) Displays the configured inspection rule.
Step 3	<code>show ip inspect session [detail]</code>  <b>Example:</b> Router# show ip inspect session	(Optional) Displays existing sessions that are currently being tracked and inspected by the Cisco IOS firewall. <ul style="list-style-type: none"> <li>The optional <b>detail</b> keyword causes additional details about these sessions to be shown.</li> </ul>
Step 4	<code>show ip access-list</code>  <b>Example:</b> Router# show ip access-list	(Optional) Displays the contents of all current IP access lists.

## Monitoring Firewall for SIP Support

To monitor firewall events, perform the following optional steps:

## SUMMARY STEPS

- enable
- debug ip inspect sip

## DETAILED STEPS

	Command or Action	Purpose
Step 1	<code>enable</code>  <b>Example:</b> Router> enable	Enables privileged EXEC mode. <ul style="list-style-type: none"> <li>Enter your password if prompted.</li> </ul>
Step 2	<code>debug ip inspect sip</code>  <b>Example:</b> Router# debug ip inspect sip	(Optional) Displays the operations of the SIP inspection engine for debugging purposes.

# Configuration Examples for Firewall SIP Support

This section provides the following configuration example:

- [Firewall and SIP Configuration Example, page 11](#)

## Firewall and SIP Configuration Example

The following example shows how to allow outside initiated calls and internal calls. For outside initiated calls, an ACL needs to be punched to allow for the traffic from the initial signaling packet from outside. Subsequent signaling and media channels will be allowed by the inspection module.

```
ip inspect name voip sip
interface FastEthernet0/0
 ip inspect voip in
!
!
interface FastEthernet0/1
 ip inspect voip in
 ip access-group 100 in
!
!
access-list 100 permit udp host <gw ip> any eq 5060
access-list 100 permit udp host <proxy ip> any eq 5060
access-list deny ip any any
```

## Additional References

For additional information related to Firewall Support for SIP, refer to the following references:

- [Related Documents, page 11](#)
- [Standards, page 12](#)
- [MIBs, page 12](#)
- [RFCs, page 12](#)
- [Technical Assistance, page 13](#)

## Related Documents

Related Topic	Document Title
Cisco IOS firewall information and configuration tasks	The chapter “ <a href="#">Configuring Context-Based Access Control</a> ” in the <i>Cisco IOS Security Configuration Guide</i> , Release 12.2
Cisco IOS firewall commands	The chapter “ <a href="#">Context-Based Access Control Commands</a> ” in the <i>Cisco IOS Security Command Reference</i> , Release 12.2
SIP information and configuration tasks	The chapter “ <a href="#">Configuring Session Initiation Protocol for Voice over IP</a> ” in the <i>Cisco IOS Voice, Video, and Fax Configuration Guide</i> , Release 12.2 and

Related Topic	Document Title
Additional SIP Information	<i>Guide to Cisco Systems' VoIP Infrastructure Solution for SIP</i>
Access lists and the <b>access-list</b> command	The chapter “ <a href="#">Configuring IPsec Network Security</a> ” in the <i>Cisco IOS Security Configuration Guide</i> , Release 12.2, and the <i>Cisco IOS Command Reference</i> , Release 12.2, respectively.

## Standards

Standards	Title
None	—

## MIBs

MIBs	MIBs Link
None	To obtain lists of supported MIBs by platform and Cisco IOS release, and to download MIB modules, go to the Cisco MIB website on Cisco.com at the following URL: <a href="http://www.cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml">http://www.cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml</a>

To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL:

<http://tools.cisco.com/ITDIT/MIBS/servlet/index>

If Cisco MIB Locator does not support the MIB information that you need, you can also obtain a list of supported MIBs and download MIBs from the Cisco MIBs page at the following URL:

<http://www.cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml>

To access Cisco MIB Locator, you must have an account on Cisco.com. If you have forgotten or lost your account information, send a blank e-mail to [cco-locksmith@cisco.com](mailto:cco-locksmith@cisco.com). An automatic check will verify that your e-mail address is registered with Cisco.com. If the check is successful, account details with a new random password will be e-mailed to you. Qualified users can establish an account on Cisco.com by following the directions found at this URL:

<http://www.cisco.com/register>

## RFCs

RFCs <sup>1</sup>	Title
RFC 2543	<i>SIP: Session Initiation Protocol</i>

1. Not all supported RFCs are listed.

## Technical Assistance

Description	Link
Technical Assistance Center (TAC) home page, containing 30,000 pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	<a href="http://www.cisco.com/public/support/tac/home.shtml">http://www.cisco.com/public/support/tac/home.shtml</a>

## Command Reference

This section documents modified commands. All other commands used with this feature are documented in the Cisco IOS Release 12.2 T command reference publications.

- [debug ip inspect](#)
- [ip inspect name](#)

# debug ip inspect

To display messages about Cisco IOS firewall events, use the **debug ip inspect** privileged EXEC command. To disable debugging output, use the **no** form of this command.

```
debug ip inspect { function-trace | object-creation | object-deletion | events | timers | protocol | detailed }
```

```
no debug ip inspect detailed
```

## Syntax Description

<b>function-trace</b>	Displays messages about software functions called by the Cisco IOS firewall.
<b>object-creation</b>	Displays messages about software objects being created by the Cisco IOS firewall. Object creation corresponds to the beginning of Cisco IOS firewall-inspected sessions.
<b>object-deletion</b>	Displays messages about software objects being deleted by the Cisco IOS firewall. Object deletion corresponds to the closing of Cisco IOS firewall-inspected sessions.
<b>events</b>	Displays messages about Cisco IOS firewall software events, including information about Cisco IOS firewall packet processing.
<b>timers</b>	Displays messages about Cisco IOS firewall timer events such as when a Cisco IOS firewall idle timeout is reached.
<i>protocol</i>	Displays messages about Cisco IOS firewall-inspected protocol events, including details about the packets of the protocol. <a href="#">Table 5</a> provides a list of <i>protocol</i> keywords.
<b>detailed</b>	Causes detailed information to be displayed for all the other enabled Cisco IOS firewall debugging. Use this form of the command in conjunction with other Cisco IOS firewall debugging commands.

**Table 5 Protocol Keywords for the debug ip inspect Command**

Application Protocol	protocol keyword
Transport-layer protocols	
TCP	tcp
User Datagram Protocol (UDP)	udp
Application-layer protocols	
CU-SeeMe	cuseeme
FTP commands and responses	ftp-cmd
FTP tokens (enables tracing of the FTP tokens parsed)	ftp-tokens

**Table 5** Protocol Keywords for the debug ip inspect Command (continued)

Application Protocol	protocol keyword
H.323 (version 1 and version 2)	h323
HTTP	http
Microsoft NetShow	netshow
RealAudio	realaudio
Remote Procedure Call (RPC)	rpc
Real Time Streaming Protocol (RTSP)	rtsp
Session Initiation Protocol (SIP)	sip
Simple Mail Transfer Protocol (SMTP)	smtp
Structured Query Language*Net (SQL*Net)	sqlnet
StreamWorks	streamworks
Trivial File Transfer Protocol (TFTP)	tftp
UNIX r-commands (rlogin, rexec, rsh)	rcmd
VDOLive	vdolive

**Command Modes**

Privileged EXEC

**Command History**

Release	Modification
11.2 P	This command was introduced.
12.0(5)T	NetShow support was added.
12.0(7)T	H.323 V2 and RTSP protocol support were added.
12.2(11)YU	SIP protocol support was added.
12.2(15)T	This command was integrated into Cisco IOS Release 12.2(15)T.

**Examples**

The following is sample output from the **debug ip inspect function-trace** command:

```
*Mar 2 01:16:16: CBAC FUNC: insp_inspection
*Mar 2 01:16:16: CBAC FUNC: insp_pre_process_sync
*Mar 2 01:16:16: CBAC FUNC: insp_find_tcp_host_entry addr 40.0.0.1 bucket 41
*Mar 2 01:16:16: CBAC FUNC: insp_find_pregen_session
*Mar 2 01:16:16: CBAC FUNC: insp_get_idbsb
*Mar 2 01:16:16: CBAC FUNC: insp_get_idbsb
*Mar 2 01:16:16: CBAC FUNC: insp_get_irc_of_idb
*Mar 2 01:16:16: CBAC FUNC: insp_get_idbsb
*Mar 2 01:16:16: CBAC FUNC: insp_create_sis
*Mar 2 01:16:16: CBAC FUNC: insp_inc_halfopen_sis
*Mar 2 01:16:16: CBAC FUNC: insp_link_session_to_hash_table
*Mar 2 01:16:16: CBAC FUNC: insp_inspect_pak
*Mar 2 01:16:16: CBAC FUNC: insp_l4_inspection
*Mar 2 01:16:16: CBAC FUNC: insp_process_tcp_seg
```

```

*Mar 2 01:16:16: CBAC FUNC: insp_listen_state
*Mar 2 01:16:16: CBAC FUNC: insp_ensure_return_traffic
*Mar 2 01:16:16: CBAC FUNC: insp_add_acl_item
*Mar 2 01:16:16: CBAC FUNC: insp_ensure_return_traffic
*Mar 2 01:16:16: CBAC FUNC: insp_add_acl_item
*Mar 2 01:16:16: CBAC FUNC: insp_process_syn_packet
*Mar 2 01:16:16: CBAC FUNC: insp_find_tcp_host_entry addr 40.0.0.1 bucket 41
*Mar 2 01:16:16: CBAC FUNC: insp_create_tcp_host_entry
*Mar 2 01:16:16: CBAC* FUNC: insp_fast_inspection
*Mar 2 01:16:16: CBAC* FUNC: insp_inspect_pak
*Mar 2 01:16:16: CBAC* FUNC: insp_l4_inspection
*Mar 2 01:16:16: CBAC* FUNC: insp_process_tcp_seg
*Mar 2 01:16:16: CBAC* FUNC: insp_synrcvd_state
*Mar 2 01:16:16: CBAC* FUNC: insp_fast_inspection
*Mar 2 01:16:16: CBAC* FUNC: insp_inspect_pak
*Mar 2 01:16:16: CBAC* FUNC: insp_l4_inspection
*Mar 2 01:16:16: CBAC* FUNC: insp_process_tcp_seg
*Mar 2 01:16:16: CBAC* FUNC: insp_synrcvd_state
*Mar 2 01:16:16: CBAC FUNC: insp_dec_halfopen_sis
*Mar 2 01:16:16: CBAC FUNC: insp_remove_sis_from_host_entry
*Mar 2 01:16:16: CBAC FUNC: insp_find_tcp_host_entry addr 40.0.0.1 bucket 41

```

This output shows the functions called by the Cisco IOS firewall as a session is inspected. Entries with an asterisk (\*) after the word “CBAC” are entries when the fast path is used; otherwise, the process path is used.

The following is sample output from the **debug ip inspect object-creation** and **debug ip inspect object-deletion** command:

```

*Mar 2 01:18:30: CBAC OBJ_CREATE: create pre-gen sis 25A3574
*Mar 2 01:18:30: CBAC OBJ_CREATE: create acl wrapper 25A36FC -- acl item 25A3634
*Mar 2 01:18:30: CBAC OBJ_CREATE: create sis 25C1CC4
*Mar 2 01:18:30: CBAC OBJ_DELETE: delete pre-gen sis 25A3574
*Mar 2 01:18:30: CBAC OBJ_CREATE: create host entry 25A3574 addr 10.0.0.1 bucket 31
*Mar 2 01:18:30: CBAC OBJ_DELETE: delete sis 25C1CC4
*Mar 2 01:18:30: CBAC OBJ_DELETE: delete create acl wrapper 25A36FC -- acl item 25A3634
*Mar 2 01:18:31: CBAC OBJ_DELETE: delete host entry 25A3574 addr 10.0.0.1

```

The following is sample output from the **debug ip inspect object-creation**, **debug ip inspect object-deletion**, and **debug ip inspect events** commands:

```

*Mar 2 01:18:51: CBAC OBJ_CREATE: create pre-gen sis 25A3574
*Mar 2 01:18:51: CBAC OBJ_CREATE: create acl wrapper 25A36FC -- acl item 25A3634
*Mar 2 01:18:51: CBAC Src 10.1.0.1 Port [1:65535]
*Mar 2 01:18:51: CBAC Dst 10.0.0.1 Port [46406:46406]
*Mar 2 01:18:51: CBAC Pre-gen sis 25A3574 created: 10.1.0.1[1:65535]
30.0.0.1[46406:46406]
*Mar 2 01:18:51: CBAC OBJ_CREATE: create sis 25C1CC4
*Mar 2 01:18:51: CBAC sis 25C1CC4 initiator_addr (10.1.0.1:20) responder_addr
(30.0.0.1:46406) initiator_alt_addr (40.0.0.1:20) responder_alt_addr (10.0.0.1:46406)
*Mar 2 01:18:51: CBAC OBJ_DELETE: delete pre-gen sis 25A3574
*Mar 2 01:18:51: CBAC OBJ_CREATE: create host entry 25A3574 addr 10.0.0.1 bucket 31
*Mar 2 01:18:51: CBAC OBJ_DELETE: delete sis 25C1CC4
*Mar 2 01:18:51: CBAC OBJ_DELETE: delete create acl wrapper 25A36FC -- acl item 25A3634
*Mar 2 01:18:51: CBAC OBJ_DELETE: delete host entry 25A3574 addr 10.0.0.1

```

The following is sample output from the **debug ip inspect timers** command:

```

*Mar 2 01:19:15: CBAC Timer Init Leaf: Pre-gen sis 25A3574
*Mar 2 01:19:15: CBAC Timer Start: Pre-gen sis 25A3574 Timer: 25A35D8 Time: 30000
milisecs
*Mar 2 01:19:15: CBAC Timer Init Leaf: sis 25C1CC4
*Mar 2 01:19:15: CBAC Timer Stop: Pre-gen sis 25A3574 Timer: 25A35D8
*Mar 2 01:19:15: CBAC Timer Start: sis 25C1CC4 Timer: 25C1D5C Time: 30000 milisecs

```

```
*Mar 2 01:19:15: CBAC Timer Start: sis 25C1CC4 Timer: 25C1D5C Time: 3600000 milisecs
*Mar 2 01:19:15: CBAC Timer Start: sis 25C1CC4 Timer: 25C1D5C Time: 5000 milisecs
*Mar 2 01:19:15: CBAC Timer Stop: sis 25C1CC4 Timer: 25C1D5C
```

The following is sample output from the **debug ip inspect tcp** command:

```
*Mar 2 01:20:43: CBAC* sis 25A3604 pak 2541C58 TCP P ack 4223720032 seq 4200176225(22)
(10.0.0.1:46409) => (10.1.0.1:21)
*Mar 2 01:20:43: CBAC* sis 25A3604 ftp L7 inspect result: PROCESS-SWITCH packet
*Mar 2 01:20:43: CBAC sis 25A3604 pak 2541C58 TCP P ack 4223720032 seq 4200176225(22)
(10.0.0.1:46409) => (10.1.0.1:21)
*Mar 2 01:20:43: CBAC sis 25A3604 ftp L7 inspect result: PASS packet
*Mar 2 01:20:43: CBAC* sis 25A3604 pak 2544374 TCP P ack 4200176247 seq 4223720032(30)
(10.0.0.1:46409) <= (10.1.0.1:21)
*Mar 2 01:20:43: CBAC* sis 25A3604 ftp L7 inspect result: PASS packet
*Mar 2 01:20:43: CBAC* sis 25A3604 pak 25412F8 TCP P ack 4223720062 seq 4200176247(15)
(10.0.0.1:46409) => (10.1.0.1:21)
*Mar 2 01:20:43: CBAC* sis 25A3604 ftp L7 inspect result: PASS packet
*Mar 2 01:20:43: CBAC sis 25C1CC4 pak 2544734 TCP S seq 4226992037(0) (10.1.0.1:20) =>
(10.0.0.1:46411)
*Mar 2 01:20:43: CBAC* sis 25C1CC4 pak 2541E38 TCP S ack 4226992038 seq 4203405054(0)
(10.1.0.1:20) <= (10.0.0.1:46411)
```

This sample shows TCP packets being processed, and lists the corresponding acknowledge (ACK) packet numbers and sequence (SEQ) numbers. The number of data bytes in the TCP packet is shown in parentheses—for example, (22). For each packet shown, the addresses and port numbers are shown separated by a colon. For example, (10.1.0.1:21) indicates an IP address of 10.1.0.1 and a TCP port number of 21.

Entries with an asterisk (\*) after the word “CBAC” are entries when the fast path is used; otherwise, the process path is used.

The following is sample output from the **debug ip inspect tcp** and **debug ip inspect detailed** commands:

```
05:08:06: CBAC SIP INVITE
0x816F3908
05:08:06: CBAC SIP client media ip 192.168.1.3CBAC SIP: insp_sip_get_nat_data() Before NAT
API call
packet dst ip/port = [192.168.101.3:5060], pre-NAT IP = 192.168.1.3

05:08:06: CBAC SIP: insp_sip_get_nat_data() After NAT API call
packet dst ip/port = [192.168.101.3:5060], post-natted IP = 192.168.1.3

05:08:06: CBAC SIP client contact ip 192.168.1.3:5060CBAC SIP: insp_sip_get_nat_data()
Before NAT API call
packet dst ip/port = [192.168.101.3:5060], pre-NAT IP = 192.168.1.3

05:08:06: CBAC SIP: insp_sip_get_nat_data() After NAT API call
packet dst ip/port = [192.168.101.3:5060], post-natted IP = 192.168.1.3

05:08:06: CBAC SIP client media port 18476
05:08:06: CBAC SIP client CallId : 3FDB097E-151911CC-800BA11C-1814763A@192.168.1.3
05:08:06: CBAC SIP client via ip 192.168.1.116:5060
05:08:06: CBAC SIP SESSION PROGRESS
0x816F3908
05:08:06: CBAC SIP client media ip 192.168.101.3CBAC SIP: insp_sip_get_nat_data() Before
NAT API call
packet dst ip/port = [192.168.1.116:5060], pre-NAT IP = 192.168.101.3

05:08:06: CBAC SIP: insp_sip_get_nat_data() After NAT API call
packet dst ip/port = [192.168.1.116:5060], post-natted IP = 192.168.101.3

05:08:06: CBAC SIP client media port 17292
05:08:06: CBAC SIP client CallId : 3FDB097E-151911CC-800BA11C-1814763A@192.168.1.3
```

```

05:08:06: CBAC SIP client via ip 192.168.1.116:5060
05:08:08: CBAC SIP OK
0x816F3908
05:08:08: CBAC SIP client media ip 192.168.101.3CBAC SIP: insp_sip_get_nat_data() Before
NAT API call
  packet dst ip/port = [192.168.1.116:5060], pre-NAT IP = 192.168.101.3

05:08:08: CBAC SIP: insp_sip_get_nat_data() After NAT API call
packet dst ip/port = [192.168.1.116:5060], post-natted IP = 192.168.101.3

05:08:08: CBAC SIP client contact ip 192.168.101.3:5060CBAC SIP: insp_sip_get_nat_data()
Before NAT API call
  packet dst ip/port = [192.168.1.116:5060], pre-NAT IP = 192.168.101.3

05:08:08: CBAC SIP: insp_sip_get_nat_data() After NAT API call
packet dst ip/port = [192.168.1.116:5060], post-natted IP = 192.168.101.3

05:08:08: CBAC SIP client media port 17292
05:08:08: CBAC SIP client CallId : 3FDB097E-151911CC-800BA11C-1814763A@192.168.1.3
05:08:08: CBAC SIP client via ip 192.168.1.116:5060
05:08:28: CBAC SIP BYE
0x816F3908
05:08:28: CBAC SIP client CallId : 3FDB097E-151911CC-800BA11C-1814763A@192.168.1.3
05:08:28: CBAC SIP client via ip 192.168.101.3:5060

*Mar  2 01:20:58: CBAC* Pak 2541E38 Find session for (30.0.0.1:46409) (40.0.0.1:21) tcp
*Mar  2 01:20:58:  P ack 4223720160 seq 4200176262(22)
*Mar  2 01:20:58: CBAC* Pak 2541E38 Addr:port pairs to match: (30.0.0.1:46409)
(40.0.0.1:21)
*Mar  2 01:20:58: CBAC* sis 25A3604 SIS_OPEN
*Mar  2 01:20:58: CBAC* Pak 2541E38 IP: s=30.0.0.1 (Ethernet0), d=40.0.0.1 (Ethernet1),
len 76,proto=6
*Mar  2 01:20:58: CBAC sis 25A3604 Saving State: SIS_OPEN/ESTAB iisn 4200176160 i_rcvnxt
4223720160 i_sndnxt 4200176262 i_rcvwnd 8760 risn 4223719771 r_rcvnxt 4200176262 r_sndnxt
4223720160 r_rcvwnd 8760
*Mar  2 01:20:58: CBAC* sis 25A3604 pak 2541E38 TCP P ack 4223720160 seq 4200176262(22)
(30.0.0.1:46409) => (40.0.0.1:21)
*Mar  2 01:20:58: CBAC* sis 25A3604 pak 2541E38 SIS_OPEN/ESTAB TCP seq 4200176262(22)
Flags: ACK 4223720160 PSH
*Mar  2 01:20:58: CBAC* sis 25A3604 pak 2541E38 --> SIS_OPEN/ESTAB iisn 4200176160
i_rcvnxt 4223720160 i_sndnxt 4200176284 i_rcvwnd 8760 risn 4223719771 r_rcvnxt 4200176262
r_sndnxt 4223720160 r_rcvwnd 8760
*Mar  2 01:20:58: CBAC* sis 25A3604 L4 inspect result: PASS packet 2541E38
(30.0.0.1:46409) (40.0.0.1:21) bytes 22 ftp
*Mar  2 01:20:58: CBAC sis 25A3604 Restoring State: SIS_OPEN/ESTAB iisn 4200176160
i_rcvnxt 4223
720160 i_sndnxt 4200176262 i_rcvwnd 8760 risn 4223719771 r_rcvnxt 4200176262 r_sndnxt
4223720160 r_rcvwnd 8760
*Mar  2 01:20:58: CBAC* sis 25A3604 ftp L7 inspect result: PROCESS-SWITCH packet
*Mar  2 01:20:58: CBAC* sis 25A3604 ftp L7 inspect result: PROCESS-SWITCH packet
*Mar  2 01:20:58: CBAC* Bump up: inspection requires the packet in the process
path(30.0.0.1) (40.0.0.1)
*Mar  2 01:20:58: CBAC Pak 2541E38 Find session for (30.0.0.1:46409) (40.0.0.1:21) tcp
*Mar  2 01:20:58:  P ack 4223720160 seq 4200176262(22)
*Mar  2 01:20:58: CBAC Pak 2541E38 Addr:port pairs to match: (30.0.0.1:46409)
(40.0.0.1:21)
*Mar  2 01:20:58: CBAC sis 25A3604 SIS_OPEN
*Mar  2 01:20:58: CBAC Pak 2541E38 IP: s=30.0.0.1 (Ethernet0), d=40.0.0.1 (Ethernet1), len
76, proto=6

```

# ip inspect name

To define a set of inspection rules, use the **ip inspect name** command in global configuration mode. To remove the inspection rule for a protocol or to remove the entire set of inspection rules, use the **no** form of this command.

```
ip inspect name inspection-name protocol [alert {on | off}] [audit-trail {on | off}]
[timeout seconds]
```

```
no ip inspect name [inspection-name protocol]
```

## Syntax Description

<i>inspection-name</i>	Names the set of inspection rules. If you want to add a protocol to an existing set of rules, use the same <i>inspection-name</i> as the existing set of rules.  <b>Note</b> The <i>inspection-name</i> cannot exceed 16 characters; otherwise, the name will be truncated to the 16-character limit.
<i>protocol</i>	A protocol keyword listed in <a href="#">Table 6</a> or <a href="#">Table 7</a> .
<b>alert {on   off}</b>	(Optional) For each inspected protocol, the generation of alert messages can be set be <b>on</b> or <b>off</b> . If no option is selected, alerts are generated on the basis of the setting of the <b>ip inspect alert-off</b> command.
<b>audit-trail {on   off}</b>	(Optional) For each inspected protocol, <b>audit trail</b> can be set <b>on</b> or <b>off</b> . If no option is selected, an audit trail message is generated on the basis of the setting of the <b>ip inspect audit-trail</b> command.
<b>timeout seconds</b>	(Optional) To override the global TCP or User Datagram Protocol (UDP) idle timeouts for the specified protocol, specify the number of seconds for a different idle timeout.  This timeout overrides the global TCP and UPD timeouts but will not override the global Domain Name System (DNS) timeout.

**Table 6 Protocol Keywords—Transport-Layer Protocols**

Protocol	Keyword
TCP	<b>tcp</b>
User Datagram Protocol (UDP)	<b>udp</b>

**Table 7 Protocol Keywords—Application-Layer Protocols**

Protocol	Keyword
CU-SeeMe	<b>cuseeme</b>
FTP	<b>ftp</b>
Java	<b>http</b>
H.323	<b>h323</b>

**Table 7 Protocol Keywords—Application-Layer Protocols (continued)**

Protocol	Keyword
Microsoft NetShow	<b>netshow</b>
RealAudio	<b>realaudio</b>
Remote Procedure Call (RPC)	<b>rpc</b>
Session Initiation Protocol (SIP)	<b>sip</b>
Simple Mail Transfer Protocol (SMTP)	<b>smtp</b>
Structured Query Language*Net (SQL*Net)	<b>sqlnet</b>
StreamWorks	<b>streamworks</b>
Trivial File Transfer Protocol (TFTP)	<b>tftp</b>
UNIX R commands (rlogin, rexec, rsh)	<b>rcmd</b>
VDOLive	<b>vdolive</b>

**Defaults**

No inspection rules are defined until you define them using this command.

**Command Modes**

Global configuration

**Command History**

Release	Modification
11.2P	This command was introduced.
12.0(5)T	Introduced configurable alert and audit trail, IP fragmentation checking, and NetShow protocol support.
12.2(11)YU	SIP protocol support was added.
12.2(15)T	This command was integrated into Cisco IOS Release 12.2(15)T

**Usage Guidelines**

To define a set of inspection rules, enter this command for each protocol that you want the Cisco IOS firewall to inspect, using the same *inspection-name*. Give each set of inspection rules a unique *inspection-name*, which should not exceed the 16-character limit. Define either one or two sets of rules per interface—you can define one set to examine both inbound and outbound traffic, or you can define two sets: one for outbound traffic and one for inbound traffic.

To define a single set of inspection rules, configure inspection for all the desired application-layer protocols, and for TCP or UDP as desired. This combination of TCP, UDP, and application-layer protocols join together to form a single set of inspection rules with a unique name.

To remove the inspection rule for a protocol, use the **no** form of this command with the specified inspection name and protocol; to remove the entire set of inspection rules, use the **no** form of this command only; that is, do not list any inspection names or protocols.

In general, when inspection is configured for a protocol, return traffic entering the internal network will be permitted only if the packets are part of a valid, existing session for which state information is being maintained.

### TCP and UDP Inspection

You can configure TCP and UDP inspection to permit TCP and UDP packets to enter the internal network through the firewall, even if the application-layer protocol is not configured to be inspected. However, TCP and UDP inspection do not recognize application-specific commands, and therefore might not permit all return packets for an application, particularly if the return packets have a different port number from the previous exiting packet.

Any application-layer protocol that is inspected will take precedence over the TCP or UDP packet inspection. For example, if inspection is configured for FTP, all control channel information will be recorded in the state table, and all FTP traffic will be permitted back through the firewall if the control channel information is valid for the state of the FTP session. The fact that TCP inspection is configured is irrelevant.

With TCP and UDP inspection, packets entering the network must exactly match an existing session: the entering packets must have the same source and destination addresses and the same source and destination port numbers as the exiting packet (but reversed). Otherwise, the entering packets will be blocked at the interface.

### Use of the timeout Keyword

If you specify a timeout for any of the transport-layer or application-layer protocols, the timeout will override the global idle timeout for the interface that the set of inspection rules is applied to.

If the protocol is TCP or a TCP application-layer protocol, the timeout will override the global TCP idle timeout. If the protocol is UDP or a UDP application-layer protocol, the timeout will override the global UDP idle timeout.

If you do not specify a timeout for a protocol, the timeout value applied to a new session of that protocol will be taken from the corresponding TCP or UDP global timeout value valid at the time of session creation.

### SIP Inspection

You can configure SIP inspection to permit media sessions associated with SIP-signaled calls to traverse the firewall. Because SIP is frequently used to signal both incoming and outgoing calls, it is often necessary to configure SIP inspection in both directions on a firewall (both from the protected internal network and from the external network). Because inspection of traffic from the external network is not done with most protocols, it may be necessary to create an additional inspection rule to cause only SIP inspection to be performed on traffic coming from the external network.

---

### Examples

The following example causes the software to inspect TCP sessions and UDP sessions and to specifically allow CU-SeeMe, FTP, and RPC traffic back through the firewall for existing sessions only. For UDP traffic, audit-trail is on. For FTP traffic, the idle timeout is set to override the global TCP idle timeout. For RPC traffic, program numbers 100003, 100005, and 100021 are permitted.

```
ip inspect name myrules tcp
ip inspect name myrules udp audit-trail on
ip inspect name myrules cuseeme
ip inspect name myrules ftp timeout 120
ip inspect name myrules rpc program-number 100003
ip inspect name myrules rpc program-number 100005
ip inspect name myrules rpc program-number 100021
```

The following example adds fragment checking to software inspection of TCP and UDP sessions for the rule named “myrules.” In this example, the firewall software will allocate 100 state structures, and the timeout value for dropping unassembled packets is set to 4 seconds. If 100 initial fragments for 100 different packets are sent through the router, all of the state structures will be used up. The initial fragment for packet 101 will be dropped. Additionally, if the number of free state structures (structures available for use by unassembled packets) drops below the threshold values, 32 or 16, the timeout value is automatically reduced to 2 or 1, respectively. Changing the timeout value frees up packet state structures more quickly.

```
ip inspect name myrules tcp
ip inspect name myrules udp audit-trail on
ip inspect name myrules cuseeme
ip inspect name myrules ftp timeout 120
ip inspect name myrules rpc program-number 100003
ip inspect name myrules rpc program-number 100005
ip inspect name myrules rpc program-number 100021
ip inspect name myrules fragment max 100 timeout 4
```

The following firewall and SIP example shows how to allow outside-initiated calls and internal calls. For outside-initiated calls, an access control list (ACL) needs to be punched to allow for the traffic from the initial signaling packet from outside. Subsequent signaling and media channels will be allowed by the inspection module.

```
ip inspect name voip sip
interface FastEthernet0/0
 ip inspect voip in
!
!
interface FastEthernet0/1
 ip inspect voip in
 ip access-group 100 in
!
!
access-list 100 permit udp host <gw ip> any eq 5060
access-list 100 permit udp host <proxy ip> any eq 5060
access-list deny ip any any
```

## Related Commands

Command	Description
<b>ip inspect</b>	Applies a set of inspection rules to an interface.
<b>ip inspect alert-off</b>	Disables Cisco IOS firewall alert messages.
<b>ip inspect audit trail</b>	Turns on Cisco IOS firewall audit trail messages, which will be displayed on the console after each Cisco IOS firewall session close.