

Command Reference

The Multimedia Conference Manager command descriptions are classified as either gatekeeper or proxy commands in the following sections:

- Gatekeeper Commands
- Proxy Commands
- Debug Commands

All other commands used with the Multimedia Conference Manager feature are documented in the Cisco IOS Release 11.3 command references.

Gatekeeper Commands

This section describes the following gatekeeper commands:

- **accounting**
- **alias static**
- **arq reject-unknown-prefix**
- **gatekeeper**
- **gw-type-prefix**
- **lrq reject-unknown-prefix**
- **security**
- **show gatekeeper calls**
- **show gatekeeper endpoints**
- **show gatekeeper gw-type-prefix**
- **show gatekeeper status**
- **show gatekeeper zone prefix**
- **show gatekeeper zone status**
- **shutdown**
- **zone access**
- **zone bw**
- **zone local**
- **zone prefix**
- **zone remote**
- **zone subnet**

accounting

To enable the accounting security feature on the gatekeeper, use the **accounting** gatekeeper configuration command. To disable accounting, use the **no** form of this command.

accounting
no accounting

Syntax Description

This command has no arguments or keywords.

Default

Disabled

Command Mode

Gatekeeper configuration

Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(2)NA.

Specify a RADIUS/TACACS+ server before using the **accounting** command.

Related Commands

radius-server host

radius-server key

alias static

To create a static entry in the local alias table, use the **alias static** gatekeeper configuration command. To remove a static entry, use the **no** form of this command.

```
alias static ip-signaling-addr [port] gkid gatekeeper-name [ras ip-ras-addr port] [terminal |
mcu | gateway {h320 | h323-proxy | voip}] [e164 e164-address] [h323id h323-id]
no alias static ip-signaling-addr [port] gkid gatekeeper-name [ras ip-ras-addr port] [terminal |
mcu | gateway {h320 | h323-proxy | voip}] [e164 e164-address] [h323id h323-id]
```

Syntax Description

<i>ip-signaling-addr</i>	IP address of the H.323 node, used as the address to signal when establishing a call.
[<i>port</i>]	(Optional) Port number other than the endpoint's Call Signaling well-known port number (1720).
gkid <i>gatekeeper-name</i>	Name of the local gatekeeper whose zone this node is a member of.
ras <i>ip-ras-addr</i>	(Optional) Node's RAS signaling address. If omitted, the <i>ip-signaling-addr</i> parameter is used in conjunction with the RAS well-known port.
<i>port</i>	(Optional) Port number other than the RAS well-known port number (1719).
terminal	(Optional) Indicates that the alias refers to a terminal.
mcu	Indicates that the alias refers to an MCU.
gateway	Indicates that the alias refers to a gateway.
h320	Indicates that the alias refers to an H.320 node.
h323-proxy	Indicates that the alias refers to an H.323 proxy.
voip	Indicates that the alias refers to VoIP.
e164 <i>e164-address</i>	(Optional) Specifies the node's e164 address. This keyword and argument can be used more than once to specify as many E.164 addresses as needed. Note that there is a maximum number of 128 characters that can be entered for this address. To avoid exceeding this limit, you can enter multiple alias static commands with the same call signaling address and different aliases.
h323id <i>h323-id</i>	(Optional) Specifies the node's H.323 alias. This keyword and argument can be used more than once to specify as many H.323 ID aliases as needed. Note that there is a maximum number of 256 characters that can be entered for this address. To avoid exceeding this limit, you can enter multiple alias static commands with the same call signaling address and different aliases.

Default

No static aliases exist.

Command Mode

Gatekeeper configuration

Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(2)NA.

The local alias table can be used to load static entries by performing as many of the commands as necessary. Aliases for the same IP address can be added in different commands, if required.

Typically, static aliases are needed to access endpoints that do not belong to a zone, (that is, they are not registered with any gatekeeper), or whose gatekeeper is inaccessible for some reason.

Example

The following example creates a static terminal alias in the local zone.

```
zone local gk.zone1.com zone1.com
alias static 191.7.8.5 gkid gk.zone1.com terminal e164 14085551212 h323id bobs_terminal
```

arq reject-unknown-prefix

To enable the gatekeeper to reject Admission Requests (ARQs) for zone prefixes that are not configured, use the **arq reject-unknown-prefix** gatekeeper command. To re-enable the gatekeeper to accept and process all incoming ARQs, use the **no** form of this command.

```
arq reject-unknown-prefix  
no arq reject-unknown-prefix
```

Syntax Description

This command has no arguments or keywords.

Default

The gatekeeper accepts and processes all incoming ARQs.

Command Mode

Gatekeeper configuration

Usage Guidelines

This command first appeared in Cisco IOS Releases 11.3(6)Q and 11.3(7)NA.

Use the **arq reject-unknown-prefix** command to configure the gatekeeper to reject any incoming ARQs for a destination E.164 address that does not match any of the configured zone prefixes.

When an endpoint or gateway initiates an H.323 call, it sends an ARQ to its gatekeeper. The gatekeeper uses the configured list of zone prefixes to determine where to direct the call. If the called address does not match any of the known zone prefixes, the gatekeeper attempts to *hairpin* the call out through a local gateway. If you do not want your gateway to do this, then use the **arq reject-unknown-prefix** command. (*hairpin* is a term used in telephony that means to send a call back in the direction that it came from. For example, if a call cannot be routed over IP to a gateway that is closer to the target phone, the call is typically sent back out the local zone, back the way it came from.)

This command is typically used to either restrict local gateway calls to a known set of prefixes or deliberately fail such calls so that an alternate choice on a gateway's rotary dial-peer is selected.

Example

Consider a gatekeeper configured as follows:

```
zone local gk408 cisco.com  
zone remote gk415 cisco.com 172.21.139.91  
zone prefix gk408 1408.....  
zone prefix gk415 1415.....
```

In this example configuration, the gatekeeper manages a zone containing gateways to the 408 area code, and it knows about a peer gatekeeper with gateways to the 415 area code. Using the **zone prefix** command, the gatekeeper is then configured with the appropriate prefixes so that calls to those area codes hop off in the optimal zone.

If the **arq request-unknown-prefix** command is not configured, the gatekeeper handles calls in the following way:

- A call to the 408 area code is routed out through a local gateway.
- A call to the 415 area code is routed to the gk415 zone where it hops off on a local gateway there.
- A call to the 212 area code is routed to a local gateway in the gk408 zone.

If the **arq reject-unknown-prefix** command is configured, the gatekeeper handles calls in the following way:

- A call to the 408 area code is routed out through a local gateway.
- A call to the 415 area code is routed to the gk415 zone where it hops off on a local gateway there.
- A call to the 212 area code is rejected, because the destination address does not match any configured prefixes.

gatekeeper

To enter gatekeeper configuration mode, use the **gatekeeper** global configuration command.

gatekeeper

Syntax Description

This command has no arguments or keywords.

Default

Disabled

Command Mode

Global configuration

Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(2)NA.

Press Ctrl-Z or use the **exit** command to exit gatekeeper configuration mode.

Example

The following example brings the gatekeeper online:

```
configure terminal
gatekeeper
no shutdown
```

gw-type-prefix

To configure a technology prefix in the gatekeeper, use the **gw-type-prefix** command. To remove the technology prefix, use the **no** form of the command.

```
gw-type-prefix type-prefix [hopoff gkid] [default-technology] [[gw ipaddr ipaddr
[ port ]] ...]
no gw-type-prefix type-prefix [hopoff gkid] [default-technology] [[gw ipaddr ipaddr
[ port ]] ...]
```

Syntax Description

type-prefix

A technology prefix is recognized and is stripped before checking for the zone prefix. It is strongly recommended that you select technology prefixes that do not lead to ambiguity with zone prefixes. Do this by using the # character to terminate technology prefixes, for example, 3#.

hopoff *gkid*

(Optional) Use this option to specify the gatekeeper or zone where the call is to hop off, regardless of the zone prefix in this destination address. The *gkid* argument specifies the gatekeeper or zone name, and can be local or remote.

default-technology

(Optional) Gateways registering with this prefix are used as the default for routing any addresses that are otherwise unresolved.

gw ipaddr *ipaddr* [*port*]

(Optional) Use this option to indicate that the gateway is incapable of registering technology prefixes. When it registers (without indicating any technology prefixes), it is added to the pool of gateway for this *type-prefix*, just as if it had sent the technology prefix in its registration. This parameter can be repeated to associate more than one gateway with a technology prefix.

Default

No technology prefix is defined.

Command Mode

Gatekeeper configuration

Usage Guidelines

This command first appeared in Cisco IOS Releases 11.3(6)Q and 11.3(7)NA.

More than one gateway can register with the same technology prefix. In such cases, a random selection is made of one of them.

You do not have to define a technology prefix to a gatekeeper if there are gateways configured to register with that prefix, and if there are no special flags (**hopoff** *gkid* or **default-technology**) that you want to associate with that prefix.

The total number of different technology prefixes should not exceed 50 per zone.

Example

The following example specifies 4# as the default technology prefix:

```
gw-type-prefix 4# default-technology
```

Related Commands

zone prefix

lrq reject-unknown-prefix

To enable the gatekeeper to reject all Location Requests (LRQs) for zone prefixes that are not configured, use the **lrq reject-unknown-prefix** gatekeeper command. To re-enable the gatekeeper to accept and process all incoming LRQs, use the **no** form of this command.

lrq reject-unknown-prefix
no lrq reject-unknown-prefix

Syntax Description

This command has no arguments or keywords.

Default

The gatekeeper accepts and processes all incoming LRQs.

Command Mode

Gatekeeper configuration

Usage Guidelines

This command first appeared in Cisco IOS Releases 11.3(6)Q and 11.3(7)NA.

Use the **lrq reject-unknown-prefix** command to configure the gatekeeper to reject any incoming LRQs for a destination E.164 address that does not match any of the configured zone prefixes.

Whether or not you enable the **lrq reject-unknown-prefix** command, the following is true when the E.164 address matches a zone prefix:

- If the matching zone prefix is local (i.e. controlled by this gatekeeper), the LRQ is serviced.
- If the matching zone prefix is remote (i.e. controlled by some other gatekeeper), the LRQ is rejected.

If you do not enable the **lrq reject-unknown-prefix** command, and the target address does not match any known local or remote prefix, the default behavior is to attempt to service the call using one of the local zones. If this default behavior is not suitable for your site, configure the **lrq reject-unknown-prefix** command on your router to force the gatekeeper to reject such requests.

Example

Consider the following gatekeeper configuration:

```
zone local gk408 cisco.com
zone local gk415 cisco.com
zone prefix gk408 1408.....
zone prefix gk415 1415.....
lrq reject-unknown-prefix
```

In this example configuration, the gatekeeper is configured to manage two zones. One zone contains gateways with interfaces in the 408 area code, and the second zone contains gateways in the 415 area code. Then using the **zone prefix** command, the gatekeeper is configured with the appropriate prefixes so that calls to those area codes hop off in the optimal zone.

Now say some other zone has been erroneously configured to route calls to the 212 area code to this gatekeeper. When the LRQ for a number in the 212 area code arrives at this gatekeeper, the gatekeeper fails to match the area code, and the LRQ is rejected.

If this was your only site that had any gateways in it, and you wanted your other sites to route all calls requiring gateways to this gatekeeper, then you can undo the **lrq reject-unknown-prefix command** by simply using the **no lrq reject-unknown-prefix**. Now when the gatekeeper receives an LRQ for the address 12125551234, it will attempt to find an appropriate gateway in either one of the zones gk408 or gk415 to service the call.

security

To enable authentication and authorization on a gatekeeper, use the **security** gatekeeper configuration command. To disable security, use the **no** form of this command.

```

security { any | h323-id | e164 } { password default password | password separator
character }
no security { any | h323-id | e164 } { password default password | password separator
character }
    
```

Syntax Description

any	Uses the first alias of an incoming RAS registration, regardless of its type, as the means of identifying the user to RADIUS/TACACS+.
h323-id	Uses the first H.323 ID type alias as the means of identifying the user to RADIUS/TACACS+.
e164	Uses the first E.164 address type alias as the means of identifying the user to RADIUS/TACACS+.
password default <i>password</i>	Specifies the default password that the gatekeeper associates with endpoints when authenticating them with an authentication server. The <i>password</i> must be identical to the password on the authentication server.
password separator <i>character</i>	<p>Specifies the character that endpoints use to separate the H.323-ID from the piggybacked password in the registration. This allows each endpoint to supply a user-specific password. The separator character and password will be stripped from the string before it is treated as an H.323-ID alias to be registered.</p> <p>Note that passwords may only be piggybacked in the H.323-ID, not the E.164 address. This is because the E.164 address allows a limited set of mostly numeric characters. If the endpoint does not wish to register an H.323-ID, it can still supply an H.323-ID consisting of just the separator character and password. This will be understood to be a password mechanism and no H.323-ID will be registered.</p>

Default
Disabled

Command Mode
Gatekeeper configuration

Usage Guidelines
This command first appeared in Cisco IOS Release 11.3(2)NA.

Use the security command to enable identification of registered aliases by RADIUS/TACACS+. If the alias does not exist in RADIUS/TACACS+, the endpoint will not be allowed to register.

A RADIUS/TACACS+ server and encryption key must have been configured in Cisco IOS software for security to work.

Only the first alias of the proper type will be identified. If no alias of the proper type is found, the registration will be rejected.

This command does not allow you to define the password mechanism unless the security type (**h323-id** or **e164** or **any**) has been defined. While the **no security password** command undefines the password mechanism, it leaves the security type unchanged, so security is still enabled. However, the **no security {h323-id | e164 | any}** command disables security entirely, including removing any existing password definitions.

Examples

The following example enables identification of registrations using the first H.323 ID found in any registration:

```
security h323id
```

The following example enables security, authenticating all users by using their H.323-IDs and a password of qwerty2x:

```
security h323-id
security password qwerty2x
```

The next example enables security, authenticating all users by using their H.323-IDs and the password entered by the user in the H.323-ID alias he or she registers:

```
security h323-id
security password separator !
```

Now if a user registers with an H.323-ID of joe!024aqx, the gatekeeper authenticates user joe with password 024aqx, and if that is successful, registers the user with the H.323-ID of joe. If the exclamation mark is not found, the user is authenticated with the default password or a null password if no default has been configured.

The following example enables security, authenticating all users by using their E.164 IDs and the password entered by the user in the H.323-ID alias he or she registers:

```
security e164
security password separator !
```

Now if a user registers with an E.164 address of 5551212 and an H.323-ID of !hs8473q6, the gatekeeper authenticates user 5551212 and password hs8473q6. Because the H.323-ID string supplied by the user begins with the separator character, no H.323-ID is registered and the user is only known by the E.164 address.

Related Commands

accounting
radius-server host
radius-server key

show gatekeeper calls

To show the status of each ongoing call that a gatekeeper is aware of, use the **show gatekeeper calls EXEC** command.

show gatekeeper calls

Syntax Description

This command has no arguments or keywords.

Command Mode

EXEC

Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(2)NA.

Sample Display

The following is sample output from the **show gatekeeper calls** command:

```
router#show gatekeeper calls

ConferenceID                               Age(secs)   BW(kbps)
-----
0x277B87C0A283D111B63E00609704D8EA    94          768
Endpoint(s): CallSignalAddr Port RASignalAddr Port
  src EP:  REMOTE
  dst EP:  90.0.0.11  1720  90.0.0.11  1700
  dst PX:  45.0.0.11  1720  70.0.0.31  24999
```

Field	Description
ConferenceID	The conference ID.
Age(secs)	The age of the call in seconds.
BW(kbps)	The bandwidth in use in kbps.
Endpoint(s)	Lists each endpoints' (terminal, gateway, or proxy) role in the call (originator, target, or proxy), and the call signaling and RAS address.

show gatekeeper endpoints

To display the status of all registered endpoints for a gatekeeper, use the **show gatekeeper endpoints EXEC** command.

show gatekeeper endpoints

Syntax Description

This command has no arguments or keywords.

Command Mode

EXEC

Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(2)NA.

Sample Display

The following is sample output from the **show gatekeeper endpoints** command:

```
router#show gatekeeper endpoints

CallSignalAddr  Port  RASSignalAddr  Port  Zone Name          Type  F
-----
172.21.127.8    1720  172.21.127.8   24999  gk-px4.cisco.com  MCU
H323-ID: joe@cisco.com
```

Field	Description
CallSignalAddr	The endpoints' IP call signaling address and IP RAS address. The conference ID is listed below, along with a list of all aliases registered for that endpoint.
Port	Port number.
RASSignalAddr	RAS address.
Port	Number of ports.
Zone Name	Zone name.
Type	The endpoint type (terminal, gateway, MCU, and so forth).
F	Whether endpoints are static (indicated by an "S") or dynamic (indicated by a "D"). An "S" is displayed when this endpoint is registered by the alias command rather than through an RAS message from the endpoint.

show gatekeeper gw-type-prefix

To display the gateway technology prefix table, use the **show gatekeeper gw-type-prefix EXEC** command.

show gatekeeper gw-type-prefix

Command Mode

EXEC

Usage Guidelines

This command first appeared in Cisco IOS Releases 11.3(6)Q and 11.3(7)NA.

Sample Display

The following is sample output from the **show gatekeeper gw-type-prefix** command:

```
router#show gatekeeper gw-type-prefix

GATEWAY-TYPE PREFIX TABLE
=====
Prefix: 3#*      (Hopoff gk408)

Prefix: 4#*      (Default gateway-technology)

Prefix: 7#*      (Hopoff gk408)
  Static Configured Gateways:
    1.1.1.1:1720
    2.2.2.2:1720
```

Field	Description
Prefix:	The technology prefix defined with the gw-type-prefix command.
(Hopoff gk408)	The address associated with the technology prefix is the hopoff point, which is handled locally. (In this display, calls specifying technology prefix 3# or 7# are always routed to zone gk408, regardless of the actual zone prefix in the destination address.)
(Default gateway-technology)	The address associated with the technology prefix is a gateway used as the default for routing any addresses that are otherwise unresolveable.
Static Configured Gateways:	Lists all IP addresses and port numbers of statically configured gateways.

show gatekeeper status

To show overall gatekeeper status that includes authorization and authentication status, zone status, and so on, use the **show gatekeeper status EXEC** command.

show gatekeeper status

Syntax Description

This command has no arguments or keywords.

Command Mode

EXEC

Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(2)NA.

Sample Display

The following is sample output from the **show gatekeeper status** command:

```
router#show gatekeeper status

Gatekeeper State: UP
Zone Name: gk-px4.cisco.com
Accounting: DISABLED
Security: DISABLED
```

Field	Description
Gatekeeper State	The gatekeeper status: <ul style="list-style-type: none"> UP is operational DOWN is administratively shut down INACTIVE is administratively enabled, that is, the no shutdown command has been issued but no local zones have been configured HSRP STANDBY indicates the gatekeeper is on hot standby and will take over when the currently active gatekeeper fails.
Zone Name	Zone name.
Accounting	Authorization and accounting status.
Security	Security status.

show gatekeeper zone prefix

To display the zone prefix table, use the **show gatekeeper zone prefix EXEC** command.

show gatekeeper zone prefix

Command Mode

EXEC

Usage Guidelines

This command first appeared in Cisco IOS Releases 11.3(6)Q and 11.3(7)NA.

Sample Display

The following is sample output from the **show gatekeeper zone prefix** command:

```
router#show gatekeeper zone prefix

      ZONE PREFIX TABLE
      =====
GK-NAME          E164-PREFIX
-----          -
gk.zone13        212.....
gk.zone14        415.....
gk.zone14        408.....
```

Field	Description
GK-NAME	The gatekeeper name.
E164-PREFIX	The E.164 prefix and a dot that acts as a wildcard for matching each remaining number in the telephone number.

show gatekeeper zone status

To display the status of zones related to a gatekeeper, use the **show gatekeeper zone status EXEC** command.

show gatekeeper zone status

Syntax Description

This command has no arguments or keywords.

Command Mode

EXEC

Usage Guideline

This command first appeared in Cisco IOS Release 11.3(2)NA.

Sample Display

The following is sample output from the **show gatekeeper zone status** command:

```
router#show gatekeeper zone status

GK name          Domain Name      RAS Address      PORT  FLAGS  MAX-BW  CUR-BW
-----          -
gk-px4.cisco     cisco.com       172.21.127.8    1719  LS     (kbps)  (kbps)
SUBNET ATTRIBUTES :
 subnet 172.21.127.0/255.255.255.0 : (Enabled)
All Other Subnets : (Disabled)
Inbound accessibility: use proxies.
othergk         cisco.com       234.234.234.234 1719  RS     0
```

Field	Description
GK name	The gatekeeper name, which is truncated after 12 characters in the display. Also, a list of subnets controlled by the gatekeeper and inbound accessibility information.
Domain Name	The domain with which the gatekeeper is associated.
RAS Address	The RAS address of the gatekeeper.
FLAGS	Displays the following information: <ul style="list-style-type: none"> • S = Static (CLI-configured, not DNS-discovered) • L = Local • R = Remote
MAX-BW	The maximum bandwidth for the zone, in kbps.
CUR-BW	The current bandwidth in use, in kbps.

shutdown

To disable the gatekeeper, use the **shutdown** gatekeeper configuration command. To enable the gatekeeper, use the **no** form of this command.

shutdown
no shutdown

Syntax Description

This command has no arguments or keywords.

Default

Disabled (shut down)

Command Mode

Gatekeeper configuration

Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(2)NA.

The gatekeeper does not have to be enabled before you can use the other gatekeeper configuration commands. In fact, it is recommended that you complete the gatekeeper configuration before bringing up the gatekeeper because some characteristics may be difficult to alter while the gatekeeper is running, as there may be active registrations or calls.

While the no shutdown command enables the gatekeeper, it does not make it operational. The two exceptions to this are:

- If no local zones are configured, a **no shutdown** command places the gatekeeper in INACTIVE mode waiting for a local zone definition.
- If local zones are defined to use an HSRP virtual address, and the HSRP interface is in STANDBY mode, the gatekeeper goes into HSRP STANDBY mode. Only when the HSRP interface is ACTIVE will the gatekeeper go into the operational UP mode.

zone access

To configure the accessibility of your local zone, use the **zone access** gatekeeper configuration command. To remove any accessibility configurations, use the **no** form of this command.

```
zone access local-zone-name {default | remote-zone remote-zone-name} {direct | proxied}
no zone access local-zone-name remote-zone remote-zone-name
```

Syntax Description

<i>local-zone-name</i>	Name of local zone (synonymous with local gatekeeper).
default	Use with the direct or proxied keyword to define the mode of behavior for all remote zones that have not been specially named using the remote-zone remote-zone-name keyword and argument combination.
remote-zone remote-zone-name	Name of remote zone (synonymous with remote gatekeeper) for which a special mode of behavior is defined.
direct	Configures direct calls (without use of proxies) between endpoints. The local zone (or gatekeeper) offers the local endpoint's IP address instead of a local proxy's IP address.
proxied	Configures calls using proxies between endpoints. The local zone (or gatekeeper) offers the local proxy's IP address instead of the local endpoint's address.

Default

The local zone allows proxied access for all remote zones.

Command Mode

Gatekeeper configuration

Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(2)NA.

By default, a gatekeeper will offer a local proxy's IP address when queried by a remote gatekeeper about a target local endpoint. This is considered proxied access. By using the **zone access** command, you can configure the local gatekeeper to offer the local endpoint's address instead of the local proxy's address. This is considered direct access.

Note The **zone access** command, configured on your local gatekeeper, only affects the use of proxies for incoming calls (that is, it does not affect the use of local proxies for outbound calls). When originating a call, a gatekeeper will use a proxy only if the remote gatekeeper offers a proxy at the remote end. A call between two endpoints in the same zone will always be a direct (non-proxied) call.

You can define the accessibility behavior of a local zone relative to certain remote zones using the **remote-zone** *remote-zone-name* keyword and argument combination with the **direct** or **proxied** keyword. You can define the default behavior of a local zone relative to all other remote zones using the **default** keyword with the **direct** or **proxied** keywords. To remove an explicitly named remote zone so that it is governed by the default-behavior rule, use the **no zone access** command.

Example

The following example allows direct access to the local zone *eng.xyz.com* from remote zones within xyz corporation. All other remote locations will have proxied access to *eng.xzy.com*.

```
zone local eng.xyz.com xyz.com
zone access eng.xyz.com remote-zone mfg.xyz.com direct
zone access eng.xyz.com remote-zone mktg.xyz.com direct
zone access eng.xyz.com remote-zone sales.xyz.com direct
zone access eng.xyz.com default proxied
```

The following example supposes that only local gatekeepers within *xyz.com* have direct access to each other because your corporation has firewalls or you do not advertise your gatekeepers externally. You have excellent QoS within your corporate network, except for a couple of foreign offices. In this case, use proxies with the foreign offices (in Milan and Tokyo) and nowhere else.

```
zone local sanjose.xyz.com xyz.com
zone access sanjose.xyz.com default direct
zone access sanjose.xyz.com remote-zone milan.xyz.com proxied
zone access sanjose.xyz.com remote-zone tokyo.xyz.com proxied
```

Related Commands

show gatekeeper zone status
zone local

zone bw

To set the maximum bandwidth allowed in a gatekeeper zone at any one time, use the **zone bw** gatekeeper configuration command. To remove the maximum bandwidth setting and make the bandwidth unlimited, use the **no** form of this command.

```
zone bw gatekeeper-name max-bandwidth  
no zone bw gatekeeper-name max-bandwidth
```

Syntax Description

<i>gatekeeper-name</i>	Name of the gatekeeper controlling the zone.
<i>max-bandwidth</i>	Maximum bidirectional bandwidth in kilobits per second (kbps) allowed in the zone at any one time.

Default

Bandwidth is unlimited.

Command Mode

Gatekeeper configuration

Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(2)NA.

Example

The following example sets the maximum bandwidth to 1000 kbps for zone gk1:

```
zone bw gk1 1000
```

Related Commands

show gatekeeper zone status

zone local

To specify a zone controlled by a gatekeeper, use the **zone local** gatekeeper configuration command. To remove a zone controlled by a gatekeeper, use the **no** form of this command. This command can also be used to change the IP address used by the gatekeeper.

```
zone local gatekeeper-name domain-name [rasIPAddress]  
no zone local gatekeeper-name domain-name
```

Syntax Description

<i>gatekeeper-name</i>	The gatekeeper's name or zone name. This is usually the fully domain-qualified host name of the gatekeeper. For example, if the <i>domain-name</i> is cisco.com, the <i>gatekeeper-name</i> might be <i>gk1.cisco.com</i> . However, if the gatekeeper is controlling multiple zones, the <i>gatekeeper-name</i> for each zone should be some unique string that has a mnemonic value.
<i>domain-name</i>	The domain name served by this gatekeeper.
<i>rasIPAddress</i>	(Optional) The IP address of one of the interfaces on the gatekeeper. When the gatekeeper responds to gatekeeper discovery messages, it signals the endpoint or gateway to use this address in future communications. Note Setting this address for one local zone makes it the address used for all local zones.

Default

No local zone is defined.

Note The gatekeeper cannot operate without at least one local zone definition. Without local zones, the gatekeeper goes to an inactive state when the **no shutdown** command is issued.

Command Mode

Gatekeeper configuration

Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(2)NA.

Multiple local zones can be defined. The gatekeeper manages all configured local zones. Intra-zone and inter-zone behavior remains the same [zones are controlled by the same or different gatekeepers.]

Only one *rasIPAddress* argument can be defined for all local zones. You cannot configure each zone to use a different RAS IP address. If you define this in the first zone definition, you can omit it for all subsequent zones, which automatically pick up this address. If you set it in a subsequent **zone local** command, it changes the RAS address of all previously configured local zones as well. Once defined, you can change it by re-issuing any **zone local** command with a different *rasIPAddress* argument.

If the *rasIPaddress* argument is an HSRP virtual address, it automatically puts the gatekeeper into HSRP mode. In this mode, the gatekeeper assumes STANDBY or ACTIVE status according to whether the HSRP interface is on STANDBY or ACTIVE status.

You cannot remove a local zone if there are endpoints or gateways registered in it. To remove the local zone, shut down the gatekeeper first, which forces unregistration.

Multiple zones are controlled by multiple logical gatekeepers on the same Cisco IOS platform.

The maximum number of local zones defined in a gatekeeper should not exceed 100.

Example

The following example creates a zone controlled by a gatekeeper in the domain called *cisco.com*:

```
zone local gk1.cisco.com cisco.com
```

Related Commands

You can use the master indexes or search on-line to find documentation of related commands.

show gatekeeper zone status

zone remote

zone prefix

To configure the gatekeeper with knowledge of its own as well as any remote zone's prefixes, use the **zone prefix** gatekeeper configuration command. To remove knowledge of zone prefixes, use the **no** form of this command.

```
zone prefix gatekeeper-name e164-prefix  
no zone prefix gatekeeper-name e164-prefix
```

Syntax Description

<i>gatekeeper-name</i>	The name of a local or remote gatekeeper, which must have been defined using the zone local or zone remote command.
<i>e164-prefix</i>	An E.164 prefix in standard form followed by dots (.) that each represent a number in the E.164 address. For example, 212..... is matched by 212 and any seven numbers. Note Although a dot representing each digit in an E.164 address is the preferred configuration method, you may also enter an asterisk (*) to match any number of digits.

Default

No knowledge of its own or any other zone's prefix is defined.

Command Mode

Gatekeeper configuration

Usage Guidelines

This command first appeared in Cisco IOS Releases 11.3(6)Q and 11.3(7)NA.

A gatekeeper can handle more than one zone prefix but a zone prefix cannot be shared by more than one gatekeeper. If you have defined a zone prefix as being handled by a gatekeeper and now define it as being handled by a second gatekeeper, the second assignment cancels the first.

If you need to have a gatekeeper handle more than one prefix, but you want to be able to group gateways by prefix, define several local zones. When a zone handles several prefixes, all gateways in that zone constitute a common pool that can be used to hop off to any of those prefixes. You may want to partition your gateways by prefix. For example, you have a gateway that interfaces to the 408 area code, and another that interfaces to the 415 area code, and for cost reasons, you want each gateway to be used only for calls to its area code. In this case, define several local zones on the gatekeeper, each responsible for a prefix, and have the each gateway register to the zone handling its prefix. You can define local zone gk-408 to handle prefix 408..... and local zone gk-415 to handle 415..... and have the gateway interfacing to the 408 area code register with gk-408, and the gateway with the 415 interface register to gk-415.

The number of zone prefixes defined for a directory gatekeeper that is dedicated to forwarding LRQs and not handling local registrations and calls should not exceed 10,000; 4 MB of memory must be dedicated to describing zones and zone prefixes to support this maximum number of zone prefixes. The number of zone prefixes defined for a gatekeeper that handles local registrations and calls should not exceed 2000.

Example

The following example matches the 212 area code and any seven digits as the zone prefix for gk-ny:

```
zone prefix gk-ny 212.....
```

Related Commands

zone local

zone remote

zone remote

To statically specify a remote zone if DNS is unavailable or undesirable, use the **zone remote** gatekeeper configuration command. To remove the remote zone, use the **no** form of this command.

```
zone remote other-gatekeeper-name other-domain-name other-gatekeeper-ip-address  
[port-number]
```

```
no zone remote other-gatekeeper-name other-domain-name other-gatekeeper-ip-address  
[port-number]
```

Syntax Description

<i>other-gatekeeper-name</i>	Name of the remote gatekeeper.
<i>other-domain-name</i>	Domain name of the remote gatekeeper.
<i>other-gatekeeper-ip-adress</i>	IP address of the remote gatekeeper.
<i>port-number</i>	(Optional) RAS signaling port number for the remote zone. Value ranges from 1 to 65535. If this is not set, the default is the well-known RAS port number 1719.

Default

No remote zone is defined. DNS will locate the remote zone.

Command Mode

Gatekeeper configuration

Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(2)NA.

All gatekeepers do not have to be in DNS. For those that are not, use the **zone remote** command so that the local gatekeeper knows how to access them. In addition, you may wish to improve call response time slightly for frequently accessed zones. If the **zone remote** command is configured for a particular zone, you do not need to make a DNS lookup transaction.

The maximum number of zones defined on a gatekeeper varies depending on the mode or the call model or both. For example, a directory gatekeeper may be in the mode of being responsible for forwarding LRQs and not handling any local registrations and calls; The call model might be E.164 addressed calls instead of H.323-ID addressed calls.

For a directory gatekeeper that does not handle local registrations and calls, the maximum remote zones defined should not exceed 10,000; An additional 4 MB of memory is required to store this maximum number of remote zones.

For a gatekeeper that handles local registrations and only E.164 addressed calls, the number of remote zones defined should not exceed 2000.

For a gatekeeper that handles H.323-ID calls, the number of remote zones defined should not exceed 200.

Example

The following example configures the local gatekeeper to reach targets of the form *xxx.cisco.com* by sending queries to the gatekeeper named *sj3.cisco.com* at IP address 1.2.3.4:

```
zone remote sj3.cisco.com cisco.com 1.2.3.4
```

Related Commands

show gatekeeper zone status

zone local

zone subnet

To configure a gatekeeper to accept discovery and registration messages sent by endpoints in designated subnets, use the **zone subnet** gatekeeper configuration command. To disable the gatekeeper from acknowledging discovery and registration messages from subnets or remove subnets entirely, use the **no** form of this command.

```
zone subnet local-gatekeeper-name { default | subnet-address { /bits-in-mask | mask-address } }  
enable  
no zone subnet local-gatekeeper-name { default | subnet-address { /bits-in-mask  
| mask-address } } enable
```

Syntax Description

<i>local-gatekeeper-name</i>	Name of the local gatekeeper.
default	Applies to all other subnets that are not specifically defined by the zone subnet command.
<i>subnet-address</i>	Address of the subnet being defined.
<i>/bits-in-mask</i>	Number of bits of the mask to be applied to the subnet address.
<i>mask-address</i>	Mask (in dotted string format) to be applied to the subnet address.
enable	Gatekeeper accepts discovery and registration from the specified subnets.

Default

The local gatekeeper accepts discovery and registration requests from all subnets. If the request specifies a gatekeeper name, it must match the local gatekeeper name or the request will not be accepted.

Command Mode

Gatekeeper configuration

Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(2)NA.

You can use the **zone subnet** command more than once to create a list of subnets controlled by a gatekeeper. The subnet masks do not have to match actual subnets in use at your site. For example, to specify a particular endpoint, you can supply its address with a 32-bit netmask.

Example

The following example starts by disabling the gatekeeper, *gk1.cisco.com*, from accepting discovery and registration messages from all subnets. Next, *gk1.cisco.com* is configured to accept discovery and registration messages from all H.323 nodes on the subnet 172.21.127.0. In addition, *gk1.cisco.com* is configured to accept discovery and registration messages from a particular endpoint with the IP address 172.21.128.56.

```
no zone subnet gk1.cisco.com default enable
zone subnet gk1.cisco.com 172.21.127.0/24 enable
zone subnet gk1.cisco.com 172.21.128.56/32 enable
```

Related Commands

show gatekeeper zone status

zone local

Proxy Commands

This section documents the following proxy commands:

- **h323 asr**
- **h323 gatekeeper**
- **h323 h323-id**
- **h323 interface**
- **h323 qos**
- **proxy h323**
- **show proxy h323 calls**
- **show proxy h323 detail-call**
- **show proxy h323 status**

h323 asr

To enable application-specific routing (ASR) and specify the maximum bandwidth for a proxy, use the **h323 asr** interface configuration command. To disable ASR, use the **no h323 asr**. Use **no h323 asr bandwidth *max-bandwidth*** to remove a bandwidth setting but keep ASR enabled.

```
h323 asr [bandwidth max-bandwidth]  
no h323 asr [bandwidth max-bandwidth]
```

Syntax Description

bandwidth *max-bandwidth* (Optional) Maximum bandwidth on the interface. Value ranges from 1 to 10,000,000 kbps. If you do not specify the *max-bandwidth*, this value defaults to the bandwidth on the interface. If you specify *max-bandwidth* as a value greater than the interface bandwidth, the bandwidth will default to the interface bandwidth.

Default

ASR is disabled.

Command Mode

Interface configuration

Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(2)NA.

This command is independent of the **h323 interface** command.

Note If you specify **no h323 asr bandwidth** *max-bandwidth*, this removes the bandwidth setting and ASR is still enabled. You must enter **no h323 asr** to disable ASR.

Example

The following example enables ASR and specifies a maximum bandwidth of 10,000 kbps:

```
h323 asr bandwidth 10000
```

h323 gatekeeper

To specify the gatekeeper associated with a proxy and control how the gatekeeper is discovered, use the **h323 gatekeeper** interface configuration command. To disassociate the gatekeeper, use the **no** form of this command.

```
h323 gatekeeper [id gatekeeper-id] {ipaddr ipaddr [port] | multicast }  
no h323 gatekeeper [id gatekeeper-id] {ipaddr ipaddr [port] | multicast }
```

Syntax Description

id <i>gatekeeper-id</i>	(Optional) The <i>gatekeeper-id</i> argument specifies the gatekeeper name. Typically, this is a DNS name, but it can also be a raw IP address in dotted form. If this parameter is specified, gatekeepers that have either the default or explicit flags set for the proxy's subnet will respond. If this parameter is not specified, only those gatekeepers with the default subnet flag will respond.
ipaddr <i>ipaddr</i> [<i>port</i>]	If this parameter is specified, the gatekeeper discovery message will be unicast to this address and, optionally, the port specified.
multicast	If this parameter is specified, the gatekeeper discovery message will be multicast to the well-known RAS multicast address and port.

Default

No gatekeeper is configured for the proxy.

Command Mode

Interface configuration

Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(2)NA.

You must enter the **h323 interface** and **h323 h323-id** commands before using this command. The **h323 gatekeeper** command must be specified on your Cisco IOS platform or the proxy will not go online. The proxy will use the interface's address as its RAS signaling address.

Example

The following example sets up a unicast discovery to a gatekeeper whose name is unknown:

```
h323 gatekeeper ipaddr 191.7.5.2
```

The following example sets up a multicast discovery for a gatekeeper of a particular name:

```
h323 gatekeeper id gk.zone5.com multicast
```

Related Commands

h323 h323-id
h323 interface

h323 h323-id

To register an H.323 proxy alias with a gatekeeper, use the **h323 h323-id** interface configuration command. To remove an H.323 alias, use the **no** form of this command.

```
h323 h323-id h323-id  
no h323 h323-id h323-id
```

Syntax Description

h323-id Specifies the name of the proxy. It is recommended that this be a fully qualified e-mail ID, with the domain name being the same as that of its gatekeeper.

Default

No *h323-id* proxy alias is registered.

Command Mode

Interface configuration

Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(2)NA.

Each entry registers a specified H.323 ID proxy alias to a gatekeeper. Typically, these aliases are either simple text strings or legitimate e-mail IDs.

Note You must enter the **h323 interface** command before using this command. The **h323 h323-id** command must be entered on the same interface as the **h323 gatekeeper** command. The proxy will not go online without this command.

Example

The following example registers an H.323 proxy alias called *proxy1@zone5.com* with a gatekeeper:

```
h323 h323-id proxy1@zone5.com
```

Related Commands

h323 gatekeeper
h323 interface

h323 interface

To specify the interface from which the proxy will take its IP address, use the **h323 interface** interface configuration command. To disable the interface, use the **no** form of this command.

h323 interface
no h323 interface

Syntax Description

This command has no arguments or keywords.

Default

Disabled

Command Mode

Interface configuration

Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(2)NA.

For non-ASR configurations, any interface on the Cisco IOS platform will work well as the proxy interface. For ASR configurations, the proxy interface should be a loopback interface, so that routing updates and packet switching are appropriately isolated between the ASR and non-ASR interfaces.

h323 qos

To enable QoS on the proxy, use the **h323 qos** interface configuration command. To disable QoS, use the **no** form of this command.

```
h323 qos { ip-precedence value | rsvp { controlled-load | guaranteed-qos } }  
no h323 qos { ip-precedence value | rsvp { controlled-load | guaranteed-qos } }
```

Syntax Description

ip-precedence <i>value</i>	Specifies that RTP streams should set their IP precedence bits to the specified <i>value</i> .
rsvp controlled-load	Specifies controlled load class of service.
rsvp guaranteed-qos	Specifies guaranteed QoS class of service.

Default

No QoS is configured.

Command Mode

Interface configuration

Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(2)NA.

You must execute the **h323 interface** command before using this command.

Both IP precedence and RSVP QoS can be configured by invoking this command twice with the two different QoS forms.

Related Commands

h323 interface

proxy h323

To enable the proxy feature on your router, use the **proxy h323** global configuration command. To disable the proxy feature, use the **no** form of this command.

proxy h323
no proxy h323

Syntax Description

This command has no arguments or keywords.

Default

Disabled

Command Mode

Global configuration

Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(2)NA.

Note If the multimedia interface is not enabled using the **proxy h323** command, or if no gatekeeper is available, starting the proxy allows it to attempt to locate these resources. No calls will be accepted until the multimedia interface and the gatekeeper are found.

Example

The following example turns on the proxy feature:

```
proxy h323
```

show proxy h323 calls

To list each active call on the proxy, use the **show h323 calls** privileged EXEC command.

```
show proxy h323 calls
```

Syntax Description

This command has no arguments or keywords.

Command Mode

Privileged EXEC

Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(2)NA.

Sample Display

The following is sample output from the **show h323 calls** command:

```
router#show proxy h323 calls

Call unique key = 1
Conference ID = [277B87C0A283D111B63E00609704D8EA]
Calling endpoint call signalling address = 55.0.0.41
Calling endpoint aliases:
  H323_ID: ptell1@zone1.com
Call state = Media Streaming
Time call was initiated = 731146290 ms
```

show proxy h323 detail-call

To display the details of a particular call on a proxy, use the **show h323 detail-call** privileged EXEC command. The display can be shown with or without the proxy statistics enabled.

```
show proxy h323 detail-call call-key
```

Syntax Description

call-key Specifies the call you want to display. The *call-key* is derived from the **show proxy h323 calls** display.

Command Mode

Privileged EXEC

Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(2)NA.

Sample Display 1

The following is sample output from the **show proxy h323 detail-call** command without the proxy statistics enabled:

```
router#show proxy h323 detail-call 1

ConferenceID = [277B87C0A283D111B63E00609704D8EA]
Calling endpoint aliases:
  H323_ID: pte111@zone1.com
Called endpoint aliases:
  H323_ID: pte121@zone2.com
Peer proxy call signalling address = 55.0.0.41
Time call was initiated = 731146290 ms
Inbound CRV = 144
Outbound CRV = 70
Call state = Media Streaming
H245 logical channels for call leg pte111@zone1.com<->px1@zone.com
  Channel number = 2
    Type = VIDEO
    State = OPEN
    Bandwidth = 374 kbps
    Time created = 731146317 ms
  Channel number = 1
    Type = AUDIO
    State = OPEN
    Bandwidth = 81 kbps
    Time created = 731146316 ms
  Channel number = 2
    Type = VIDEO
    State = OPEN
    Bandwidth = 374 kbps
    Time created = 731146318 ms
  Channel number = 1
    Type = AUDIO
    State = OPEN
    Bandwidth = 81 kbps
    Time created = 731146317 ms
H245 logical channels for call leg pte111@zone1.com<->50.0.0.41:
```

```

Channel number = 2
  Type = VIDEO
  State = OPEN
  Bandwidth = 374 kbps
  Time created = 731146317 ms
Channel number = 1
  Type = AUDIO
  State = OPEN
  Bandwidth = 81 kbps
  Time created = 731146316 ms
Channel number = 2
  Type = VIDEO
  State = OPEN
  Bandwidth = 374 kbps
  Time created = 731146318 ms
Channel number = 1
  Type = AUDIO
  State = OPEN
  Bandwidth = 81 kbps
  Time created = 731146317 ms

```

Sample Display 2

The following is sample output from the **show proxy h323 detail-call** command with the proxy statistics enabled:

```

router#show proxy h323 detail-call 1
ConferenceID = [677EB106BD0D111976200002424F832]
Calling endpoint call signalling address = 172.21.127.49
  Calling endpoint aliases:
    H323_ID: intel2
    E164_ID: 2134
Called endpoint aliases:
  H323_ID: mcs@sanjose.cisco.com
Peer proxy call signalling address = 171.68.183.199
Peer proxy aliases:
  H323_ID: proxy.sanjose.cisco.com
Time call was initiated = 730949651 ms
Inbound CRV = 2505
Outbound CRV = 67
Call state = H245 open logical channels
H245 logical channels for call leg intel2 <-> cisco7-pxy:
  Channel number = 259
    RTP stream from intel2 to cisco7-pxy
      Type = VIDEO
      State = OPEN
      Bandwidth = 225 kbps
      Time created = 730949676 ms
  Channel number = 257
    RTP stream from intel2 to cisco7-pxy
      Type = AUDIO
      State = OPEN
      Bandwidth = 18 kbps
      Time created = 730949658 ms
  Channel number = 2
    RTP stream from cisco7-pxy to intel2
      Type = VIDEO
      State = OPEN
      Bandwidth = 225 kbps
      Time created = 730949664 ms
    RTP Statistics:
      Packet Received Count = 3390
      Packet Dropped Count = 0
      Packet Out of Sequence Count = 0

```

Command Reference

Number of initial packets used for Arrival-Spacing bin setup = 200
min_arrival_spacing = 0(ms) max_arrival_spacing = 856(ms)

Average Arrival Rate = 86(ms)

Arrival-Spacing(ms)	Packet-Count
0	2116
26	487
52	26
78	0
104	0
130	1
156	0
182	1
208	0
234	4
260	99
286	315
312	154
338	8
364	0
390	2
416	10
442	73
468	51
494	43

=====
Min Jitter = 34(ms) Max Jitter = 408(ms)
Average Jitter Rate = 117

Jitter Rate(ms)	Packet-Count
0	0
41	514
82	2117

Number of initial packets used for Arrival-Spacing bin setup = 200
min_arrival_spacing = 32(ms) max_arrival_spacing = 96(ms)

Average Arrival Rate = 60(ms)

Arrival-Spacing(ms)	Packet-Count
32	35
34	0
36	177
38	0
40	56
42	0
44	10
46	0
48	27
50	0
52	541
54	0
56	2642
58	1
60	1069
62	0
64	77
66	0
68	6
70	257

=====
Min Jitter = 0(ms) Max Jitter = 28(ms)
Average Jitter Rate = 5

Jitter Rate(ms)	Packet-Count
0	1069
3	2720
6	0
9	804
12	27
15	10

```

        18                0
        21                56
        24                177
        27                35
H245 logical channels for call leg cisco7-pxy <->
proxy.sanjose.cisco.com:
  Channel number = 259
    RTP stream from cisco7-pxy to proxy.sanjose.cisco.com
      Type = VIDEO
      State = OPEN
      Bandwidth = 225 kbps
      Time created = 730949676 ms
      RTP Statistics:
        Packet Received Count = 3398
        Packet Dropped Count = 1
        Packet Out of Sequence Count = 0
        Number of initial packets used for Arrival-Spacing bin setup = 200
        min_arrival_spacing = 0(ms)  max_arrival_spacing = 872(ms)
        Average Arrival Rate = 85(ms)
        Arrival-Spacing(ms)  Packet-Count
          0                   2636
          28                   0
          56                   0
          84                   0
          112                   0
          140                   1
          168                   0
          196                   0
          224                   0
          252                   0
          280                   2
          308                   425
          336                   154
          364                   5
          392                   0
          420                   0
          448                   0
          476                   114
          504                   41
          532                   20
        =====
        Min Jitter = 55(ms)  Max Jitter = 447(ms)
        Average Jitter Rate = 127
        Jitter Rate(ms)  Packet-Count
          0                0
          45                1
          90                2636
          135                0
          180                2
          225                425
          270                159
          315                0
          360                0
          405                175
  Channel number = 257
    RTP stream from cisco7-pxy to proxy.sanjose.cisco.com
      Type = AUDIO
      State = OPEN
      Bandwidth = 18 kbps
      Time created = 730949658 ms
      RTP Statistics:
        Packet Received Count = 2537
        Packet Dropped Count = 3
        Packet Out of Sequence Count = 0
        Number of initial packets used for Arrival-Spacing bin setup = 200

```

min_arrival_spacing = 0(ms) max_arrival_spacing = 32716(ms)
Average Arrival Rate = 112(ms)

Arrival-Spacing(ms)	Packet-Count
0	2191
72	253
144	31
216	7
288	3
360	4
432	4
504	2
576	1
648	3
720	2
792	1
864	2
936	1
1008	1
1080	1
1152	1
1224	1
1296	0
1368	28

=====
Min Jitter = 32(ms) Max Jitter = 1256(ms)
Average Jitter Rate = 121

Jitter Rate(ms)	Packet-Count
0	284
126	2201
252	4
378	6
504	4
630	3
756	2
882	2
1008	2
1134	29

Channel number = 2
RTP stream from proxy.sanjose.cisco.com to cisco7-pxy
Type = VIDEO
State = OPEN
Bandwidth = 225 kbps
Time created = 730949664 ms
Channel number = 1
RTP stream from proxy.sanjose.cisco.com to cisco7-pxy
Type = AUDIO
State = OPEN
Bandwidth = 18 kbps
Time created = 730949661 ms

Related Commands

h323 qos

show proxy h323 status

To display the overall status of a proxy, use the **show proxy h323 status** privileged EXEC command.

show proxy h323 status

Syntax Description

This command has no arguments or keywords.

Command Mode

EXEC

Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(2)NA.

Sample Display

The following is sample output from the **show proxy h323 status** command:

```
router#show proxy h323 status

H.323 Proxy Status
=====
H.323 Proxy Mode: Enabled
Proxy interface = Serial1: UP
Application Specific Routing: Disabled
RAS Initialization: Complete
Proxy aliases configured:
  H323_ID: px2
Proxy aliases assigned by Gatekeeper:
  H323_ID: px2
Gatekeeper multicast discovery: Disabled
Gatekeeper:
  Gatekeeper ID: gk.zone2.com
  IP address: 70.0.0.31
Gatekeeper registration succeeded
T.120 Mode: BYPASS
RTP Statistics: OFF
Number of calls in progress: 1
```

Debug Commands

This section describes the following **debug** commands:

- **debug h225 asn1**
- **debug h225 events**
- **debug h245 asn1**
- **debug h245 events**
- **debug proxy h323 statistics**
- **debug ras**

debug h225 asn1

Use the **debug h225 asn1 EXEC** command to display ASN1 contents of RAS and Q.931 messages. The **no** form of this command disables debugging output.

[no] debug h225 asn1

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(2)NA.

Note This command slows the system down considerably and connections may time out.

Sample Displays

The following are sample output from the **debug h225 asn1** command.

Sample 1: Gatekeeper Trace with ASN1 Turned On, Call Being Established

This report shows two proxy call scenarios. A trace is collected on the gatekeeper with ASN1 turned on. The call is being established.

```
gk1#debug h225 asn1
H.225 ASN1 Messages debugging is on
gk1#24800006 03C00030 00300036 00380041 00450037 00430030 00300030 00300030
00300030 00310140 0F007000 74006500 6C003200 33004000 7A006F00 6E006500
32002E00 63006F00 6D020180 AAAA4006 00700074 0065006C 00320031 0033401E
0000015F C8490FB4 B9D111BF AF0060B0 00E94500
value RasMessage ::= admissionRequest :
{
  requestSeqNum 7,
  callType pointToPoint : NULL,
  endpointIdentifier "0068AE7C00000001",
  destinationInfo
  {
    h323-ID : "ptel23@zone2.com"
  },
  srcInfo
```

```

    {
      e164 : "7777",
      h323-ID : "ptel213"
    },
    bandwidth 7680,
    callReferenceValue 1,
    conferenceID '5FC8490FB4B9D111BFAF0060B000E945'H,
    activeMC FALSE,
    answerCall FALSE
  }
value RasMessage ::= admissionConfirm :
  {
    requestSeqNum 7,
    bandwidth 7680,
    callModel direct : NULL,
    destCallSignalAddress ipAddress :
      {
        ip '65000001'H,
        port 1720
      },
    irrFrequency 30
  }
29000006 401E0000 65000001 06B8001D
2480001D 03C00030 00300036 00380041 00390036 00300030 00300030 00300030
00300030 00320140 0F007000 74006500 6C003200 33004000 7A006F00 6E006500
32002E00 63006F00 6D014006 00700074 0065006C 00320031 00334002 8000015F
C8490FB4 B9D111BF AF0060B0 00E94540
value RasMessage ::= admissionRequest :
  {
    requestSeqNum 30,
    callType pointToPoint : NULL,
    endpointIdentifier "0068A96000000002",
    destinationInfo
      {
        h323-ID : "ptel23@zone2.com"
      },
    srcInfo
      {
        h323-ID : "ptel213"
      },
    bandwidth 640,
    callReferenceValue 1,
    conferenceID '5FC8490FB4B9D111BFAF0060B000E945'H,
    activeMC FALSE,
    answerCall TRUE
  }
value ACFnonStandardInfo ::=
  {
    srcTerminalAlias
      {
        e164 : "7777",
        h323-ID : "ptel213"
      },
    dstTerminalAlias
      {
        h323-ID : "ptel23@zone2.com"
      },
    dstProxyAlias
      {
        h323-ID : "px2"
      },
    dstProxySignalAddress
      {
        ip '66000001'H,
        port 1720
      }
  }

```

Command Reference

```
    }
  }
C00203AA AA800600 70007400 65006C00 32003100 3301800F 00700074 0065006C
00320033 0040007A 006F006E 00650032 002E0063 006F006D 01800200 70007800
32660000 0106B8
value RasMessage ::= admissionConfirm :
{
  requestSeqNum 30,
  bandwidth 7680,
  callModel direct : NULL,
  destCallSignalAddress ipAddress :
  {
    ip '66000001'H,
    port 1720
  },
  irrFrequency 30,
  nonStandardData
  {
    nonStandardIdentifier h221NonStandard :
    {
      t35CountryCode 181,
      t35Extension 0,
      manufacturerCode 18
    },
    data
    'C00203AAAA8006007000740065006C00320031003301800F007000740065006C003200 ...'H
  }
}
2980001D 401E0000 66000001 06B8001D 40B50000 1247C002 03AAAA80 06007000
74006500 6C003200 31003301 800F0070 00740065 006C0032 00330040 007A006F
006E0065 0032002E 0063006F 006D0180 02007000 78003266 00000106 B8
24C0001E 03C00030 00300036 00380041 00390036 00300030 00300030 00300030
00300030 00320140 0F007000 74006500 6C003200 33004000 7A006F00 6E006500
32002E00 63006F00 6D006600 000106B8 020180AA AA400600 70007400 65006C00
32003100 33401E00 00435FC8 490FB4B9 D111BFAF 0060B000 E94500
value RasMessage ::= admissionRequest :
{
  requestSeqNum 31,
  callType pointToPoint : NULL,
  endpointIdentifier "0068A96000000002",
  destinationInfo
  {
    h323-ID : "ptel23@zone2.com"
  },
  destCallSignalAddress ipAddress :
  {
    ip '66000001'H,
    port 1720
  },
  srcInfo
  {
    e164 : "7777",
    h323-ID : "ptel213"
  },
  bandwidth 7680,
  callReferenceValue 67,
  conferenceID '5FC8490FB4B9D111BFAF0060B000E945'H,
  activeMC FALSE,
  answerCall FALSE
}
value RasMessage ::= admissionConfirm :
{
  requestSeqNum 31,
  bandwidth 7680,
  callModel direct : NULL,
```

```

destCallSignalAddress ipAddress :
{
  ip '66000001'H,
  port 1720
},
irrFrequency 30
}

```

Sample 2: Source Proxy Trace with ASN1 Turned On, Call Being Torn Down

This report shows two proxy call scenarios. A trace is collected on the source proxy with ASN1 turned on. The call is being torn down

```

px1#debug h225 asn1
H.225 ASN1 Messages debugging is on
px1#
value H323-UserInformation ::=
{
  h323-uu-pdu
  {
    h323-message-body setup :
    {
      protocolIdentifier { 0 0 8 2250 0 1 },
      sourceAddress
      {
        h323-ID : "ptel213"
      },
      sourceInfo
      {
        terminal
        {
          },
        mc FALSE,
        undefinedNode FALSE
      },
      destinationAddress
      {
        h323-ID : "ptel23@zone2.com"
      },
      activeMC FALSE,
      conferenceID '5FC8490FB4B9D111BFAF0060B000E945'H,
      conferenceGoal create : NULL,
      callType pointToPoint : NULL,
      sourceCallSignalAddress ipAddress :
      {
        ip '3200000C'H,
        port 1720
      }
    }
  }
}
value RasMessage ::= admissionRequest :
{
  requestSeqNum 30,
  callType pointToPoint : NULL,
  endpointIdentifier "0068A96000000002",
  destinationInfo
  {
    h323-ID : "ptel23@zone2.com"
  },
  srcInfo
  {
    h323-ID : "ptel213"
  },
}

```

Command Reference

```
bandWidth 640,
callReferenceValue 1,
conferenceID '5FC8490FB4B9D111BFAF0060B000E945'H,
activeMC FALSE,
answerCall TRUE
}
2480001D 03C00030 00300036 00380041 00390036 00300030 00300030 00300030
00300030 00320140 0F007000 74006500 6C003200 33004000 7A006F00 6E006500
32002E00 63006F00 6D014006 00700074 0065006C 00320031 00334002 8000015F
C8490FB4 B9D111BF AF0060B0 00E94540
2980001D 401E0000 66000001 06B8001D 40B50000 1247C002 03AAAA80 06007000
74006500 6C003200 31003301 800F0070 00740065 006C0032 00330040 007A006F
006E0065 0032002E 0063006F 006D0180 02007000 78003266 00000106 B8
value RasMessage ::= admissionConfirm :
{
  requestSeqNum 30,
  bandWidth 7680,
  callModel direct : NULL,
  destCallSignalAddress ipAddress :
  {
    ip '66000001'H,
    port 1720
  },
  irrFrequency 30,
  nonStandardData
  {
    nonStandardIdentifier h221NonStandard :
    {
      t35CountryCode 181,
      t35Extension 0,
      manufacturerCode 18
    },
    data
    'C00203AAAA8006007000740065006C00320031003301800F007000740065006C003200 ...'H
  }
}
C00203AA AA800600 70007400 65006C00 32003100 3301800F 00700074 0065006C
00320033 0040007A 006F006E 00650032 002E0063 006F006D 01800200 70007800
32660000 0106B8
value ACFnonStandardInfo ::=
{
  srcTerminalAlias
  {
    e164 : "7777",
    h323-ID : "ptel213"
  },
  dstTerminalAlias
  {
    h323-ID : "ptel23@zone2.com"
  },
  dstProxyAlias
  {
    h323-ID : "px2"
  },
  dstProxySignalAddress
  {
    ip '66000001'H,
    port 1720
  }
}
value RasMessage ::= admissionRequest :
{
  requestSeqNum 31,
  callType pointToPoint : NULL,
  endpointIdentifier "0068A96000000002",
```

```

destinationInfo
{
  h323-ID : "ptel23@zone2.com"
},
destCallSignalAddress ipAddress :
{
  ip '66000001'H,
  port 1720
},
srcInfo
{
  e164 : "7777",
  h323-ID : "ptel213"
},
bandWidth 7680,
callReferenceValue 67,
conferenceID '5FC8490FB4B9D111BFAF0060B000E945'H,
activeMC FALSE,
answerCall FALSE
}
24C0001E 03C00030 00300036 00380041 00390036 00300030 00300030 00300030
00300030 00320140 0F007000 74006500 6C003200 33004000 7A006F00 6E006500
32002E00 63006F00 6D006600 000106B8 020180AA AA400600 70007400 65006C00
32003100 33401E00 00435FC8 490FB4B9 D111BFAF 0060B000 E94500
2900001E 401E0000 66000001 06B8001D
value RasMessage ::= admissionConfirm :
{
  requestSeqNum 31,
  bandWidth 7680,
  callModel direct : NULL,
  destCallSignalAddress ipAddress :
  {
    ip '66000001'H,
    port 1720
  },
  irrFrequency 30
}
value H323-UserInformation ::=
{
  h323-uu-pdu
  {
    h323-message-body callProceeding :
    {
      protocolIdentifier { 0 0 8 2250 0 1 },
      destinationInfo
      {
        gateway
        {
          protocol
          {
            h323 :
            {
              {
            }
          },
          mc FALSE,
          undefinedNode FALSE
        }
      }
    }
  }
}
01000600 08914A00 01088001 2800
value H323-UserInformation ::=
{
  h323-uu-pdu

```

Command Reference

```
{
  h323-message-body setup :
  {
    protocolIdentifier { 0 0 8 2250 0 1 },
    sourceAddress
    {
      h323-ID : "ptel213"
    },
    sourceInfo
    {
      vendor
      {
        vendor
        {
          t35CountryCode 181,
          t35Extension 0,
          manufacturerCode 18
        }
      },
      gateway
      {
        protocol
        {
          h323 :
          {
          }
        }
      },
      mc FALSE,
      undefinedNode FALSE
    },
    destinationAddress
    {
      h323-ID : "ptel23@zone2.com"
    },
    destCallSignalAddress ipAddress :
    {
      ip '66000001'H,
      port 1720
    },
    activeMC FALSE,
    conferenceID '5FC8490FB4B9D111BFAF0060B000E945'H,
    conferenceGoal create : NULL,
    callType pointToPoint : NULL,
    sourceCallSignalAddress ipAddress :
    {
      ip '65000001'H,
      port 1720
    },
    remoteExtensionAddress h323-ID : "ptel23@zone2.com"
  }
}
}
00B80600 08914A00 01014006 00700074 0065006C 00320031 00332800 B5000012
40012800 01400F00 70007400 65006C00 32003300 40007A00 6F006E00 65003200
2E006300 6F006D00 66000001 06B8005F C8490FB4 B9D111BF AF0060B0 00E94500
0E070065 00000106 B822400F 00700074 0065006C 00320033 0040007A 006F006E
00650032 002E0063 006F006D
value H323-UserInformation ::=
{
  h323-uu-pdu
  {
    h323-message-body callProceeding :
    {
      protocolIdentifier { 0 0 8 2250 0 1 },
```

```

destinationInfo
{
  gateway
  {
    protocol
    {
      h323 :
      {
      }
    }
  },
  mc FALSE,
  undefinedNode FALSE
}
}
}
value H323-UserInformation ::=
{
  h323-uu-pdu
  {
    h323-message-body alerting :
    {
      protocolIdentifier { 0 0 8 2250 0 1 },
      destinationInfo
      {
        mc FALSE,
        undefinedNode FALSE
      }
    }
  }
}
value H323-UserInformation ::=
{
  h323-uu-pdu
  {
    h323-message-body alerting :
    {
      protocolIdentifier { 0 0 8 2250 0 1 },
      destinationInfo
      {
        mc FALSE,
        undefinedNode FALSE
      }
    }
  }
}
03000600 08914A00 010000
value H323-UserInformation ::=
{
  h323-uu-pdu
  {
    h323-message-body connect :
    {
      protocolIdentifier { 0 0 8 2250 0 1 },
      h245Address ipAddress :
      {
        ip '66000001'H,
        port 11011
      },
      destinationInfo
      {
        gateway
        {
          protocol

```

```

        {
            h323 :
                {
                }
            }
        },
        mc FALSE,
        undefinedNode FALSE
    },
    conferenceID '5FC8490FB4B9D111BFAF0060B000E945'H
}
}
}
value H323-UserInformation ::=
{
    h323-uu-pdu
    {
        h323-message-body connect :
        {
            protocolIdentifier { 0 0 8 2250 0 1 },
            h245Address ipAddress :
            {
                ip '65000001'H,
                port 11007
            },
            destinationInfo
            {
                gateway
                {
                    protocol
                    {
                        h323 :
                        {
                        }
                    }
                },
                mc FALSE,
                undefinedNode FALSE
            },
            conferenceID '5FC8490FB4B9D111BFAF0060B000E945'H
        }
    }
}
}
02400600 08914A00 01006500 00012AFF 08800128 005FC849 0FB4B9D1 11BFAF00
60B000E9 45

```

Sample 3: Destination Router Trace, Both RAS and H.225 Traces Are Enabled

This report shows two proxy call scenarios. A trace is collected on a destination router where both destination proxy and destination gatekeeper co-exist. Both RAS and H.225 traces are enabled for one complete call.

```

px2#
RASLib::RASRecvData: successfully rcvd message of length 80 from 40.0.0.33:1585
RASLib::RASRecvData: LRQ rcvd from [40.0.0.33:1585] on sock [6880372]
RASlib::ras_sendto: msg length 111 sent to 40.0.0.33
RASLib::RASSendLCF: LCF sent to 40.0.0.33
H225Lib::h225TAccept: TCP connection accepted from 101.0.0.1:11002 on
socket [2]
H225Lib::h225TAccept: Q.931 Call State is initialized to be [Null].
Hex representation of the received TPKT
030000A60802008005040488988CA56C0591373737377E008D0500B8060008914A000101400
6007000740065006C0032003100332800B50000124001280001400F007000740065006C00320

```

```
0330040007A006F006E00650032002E0063006F006D006600000106B8003DC8490FB4B9D111B
FAF0060B000E945000E07006500000106B822400F007000740065006C003200330040007A006
F006E00650032002E0063006F006D
H225Lib::h225RecvData: Q.931 SETUP received from socket [2]
H225Lib::h225RecvData: State changed to [Call Present].
RASLib::ras_sendto: msg length 119 sent to 102.0.0.1
RASLib::RASSendARQ: ARQ sent to 102.0.0.1
RASLib::RASRecvData: successfully rcvd message of length 119 from 102.0.0.1:24999
RASLib::RASRecvData: ARQ rcvd from [102.0.0.1:24999] on sock [0x68FC74]
RASLib::ras_sendto: msg length 16 sent to 70.0.0.31
RASLib::RASSendACF: ACF sent to 70.0.0.31
RASLib::RASRecvData: successfully rcvd message of length 16 from 102.0.0.1:1719
RASLib::RASRecvData: ACF rcvd from [102.0.0.1:1719] on sock [0x67E6A4]
RASLib::ras_sendto: msg length 119 sent to 102.0.0.1
RASLib::RASSendARQ: ARQ sent to 102.0.0.1
RASLib::RASRecvData: successfully rcvd message of length 119 from 102.0.0.1:24999
RASLib::RASRecvData: ARQ rcvd from [102.0.0.1:24999] on sock [0x68FC74]
RASLib::ras_sendto: msg length 16 sent to 70.0.0.31
RASLib::RASSendACF: ACF sent to 70.0.0.31
RASLib::RASRecvData: successfully rcvd message of length 16 from 102.0.0.1:1719
RASLib::RASRecvData: ACF rcvd from [102.0.0.1:1719] on sock [0x67E6A4]
Hex representation of the CALL PROCEEDING TPKT to send.
0300001B08028080027E000F050100060008914A00010880012800
H225Lib::h225CallProcRequest: Q.931 CALL PROCEEDING sent from socket
[2]. Call state remains unchanged (Q.931 FSM simplified for H.225.0)
H225Lib::h225TConn: connect in progress on socket [4]
H225Lib::h225TConn: Q.931 Call State is initialized to be [Null].
Hex representation of the SETUP TPKT to send.
030000A50802008005040388C0A56C05913737377E008D0500B8060008914A00010140060
07000740065006C0032003100332800B500000124001280001400F007000740065006C0032003
30040007A006F006E00650032002E0063006F006D005A00000D06B8003DC8490FB4B9D111BFA
F0060B000E945000E07006600000106B822400F007000740065006C003200330040007A006F0
06E00650032002E0063006F006D
H225Lib::h225SetupRequest: Q.931 SETUP sent from socket [4]
H225Lib::h225SetupRequest: Q.931 Call State changed to [Call Initiated].
RASLib::RASRecvData: successfully rcvd message of length 123 from 90.0.0.13:1700
RASLib::RASRecvData: ARQ rcvd from [90.0.0.13:1700] on sock [0x68FC74]
RASLib::ras_sendto: msg length 16 sent to 90.0.0.13
RASLib::RASSendACF: ACF sent to 90.0.0.13
Hex representation of the received TPKT
0300001808028080027E000C050100060008914A00010200
H225Lib::h225RecvData: Q.931 CALL PROCEEDING received from socket [4]
Hex representation of the received TPKT
0300001808028080017E000C050300060008914A00010200
H225Lib::h225RecvData: Q.931 ALERTING received from socket [4]
H225Lib::h225RecvData: Q.931 Call State changed to [Call Delivered].
Hex representation of the ALERTING TPKT to send.
0300001808028080017E000C050300060008914A00010000
H225Lib::h225AlertRequest: Q.931 ALERTING sent from socket [2]. Call
state changed to [Call Received].
Hex representation of the received TPKT
0300003508028080070404889886A57E0023050240060008914A0001005A00000D06A402003
DC8490FB4B9D111BFAF0060B000E945
H225Lib::h225RecvData: Q.931 CONNECT received from socket [4]
H225Lib::h225RecvData: Q.931 Call State changed to [Active].
Hex representation of the CONNECT TPKT to send.
030000370802808007040388C0A57E0026050240060008914A000100660000012AFC0880012
8003DC8490FB4B9D111BFAF0060B000E945
H225Lib::h225SetupResponse: Q.931 CONNECT sent from socket [2]
H225Lib::h225SetupResponse: Q.931 Call State changed to [Active].
RASLib::ras_sendto: msg length 108 sent to 102.0.0.1
RASLib::RASSendIRR: IRR sent to 102.0.0.1
RASLib::RASRecvData: successfully rcvd message of length 108 from 102.0.0.1:24999
RASLib::RASRecvData: IRR rcvd from [102.0.0.1:24999] on sock [0x68FC74]
RASLib::RASRecvData: successfully rcvd message of length 101 from 90.0.0.13:1700
```

Command Reference

```
RASLib::RASRecvData: IRR rcvd from [90.0.0.13:1700] on sock [0x68FC74]
Hex representation of the received TPKT
0300001A080280805A080280107E000A050500060008914A0001
H225Lib::h225RecvData: Q.931 RELEASE COMPLETE received from socket [2]
H225Lib::h225RecvData: Q.931 Call State changed to [Null].
RASLib::ras_sendto: msg length 55 sent to 102.0.0.1
RASLib::RASSendDRQ: DRQ sent to 102.0.0.1
H225Lib::h225RecvData: no connection on socket [2]
RASLib::RASRecvData: successfully rcvd message of length 55 from 102.0.0.1:24999
RASLib::RASRecvData: DRQ rcvd from [102.0.0.1:24999] on sock [0x68FC74]
RASLib::ras_sendto: msg length 3 sent to 70.0.0.31
RASLib::RASSendDCF: DCF sent to 70.0.0.31
Hex representation of the RELEASE COMPLETE TPKT to send.
0300001A080280805A080280107E000A050500060008914A0001
H225Lib::h225TerminateRequest: Q.931 RELEASE COMPLETE sent from socket [2]. Call state changed to
[Null].
H225Lib::h225TClose: TCP connection from socket [2] closed
RASLib::ras_sendto: msg length 55 sent to 102.0.0.1
RASLib::RASSendDRQ: DRQ sent to 102.0.0.1
RASLib::RASRecvData: successfully rcvd message of length 3 from 102.0.0.1:1719
RASLib::RASRecvData: DCF rcvd from [102.0.0.1:1719] on sock [0x67E6A4]
RASLib::RASRecvData: successfully rcvd message of length 55 from 102.0.0.1:24999
RASLib::RASRecvData: DRQ rcvd from [102.0.0.1:24999] on sock [0x68FC74]
RASLib::ras_sendto: msg length 3 sent to 70.0.0.31
RASLib::RASSendDCF: DCF sent to 70.0.0.31
RASLib::RASRecvData: successfully rcvd message of length 3 from 102.0.0.1:1719
RASLib::RASRecvData: DCF rcvd from [102.0.0.1:1719] on sock [0x67E6A4]
Hex representation of the RELEASE COMPLETE TPKT to send.
0300001A080280805A080280107E000A050500060008914A0001
H225Lib::h225TerminateRequest: Q.931 RELEASE COMPLETE sent from socket [4]. Call state changed to
[Null].
H225Lib::h225TClose: TCP connection from socket [4] closed
RASLib::RASRecvData: successfully rcvd message of length 55 from 90.0.0.13:1700
RASLib::RASRecvData: DRQ rcvd from [90.0.0.13:1700] on sock [0x68FC74]
RASLib::ras_sendto: msg length 3 sent to 90.0.0.13
RASLib::RASSendDCF: DCF sent to 90.0.0.13
```

debug h225 events

Use the **debug h225 events** EXEC command to display Q.931 events. The **no** form of this command disables debugging output.

[no] debug h225 events

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(2)NA.

Sample Displays

The following are sample output from the **debug h225 events** command.

Sample 1: Source Proxy Trace with H.225 Turned On, Call Being Established

This report shows two proxy call scenarios. A trace is collected on the source proxy with H.225 turned on. The call is being established.

```

pxl#debug h225 events
H.225 Event Messages debugging is on
pxl# H225Lib::h225TAccept: TCP connection accepted from 50.0.0.12:1701 on
socket [2]
    H225Lib::h225TAccept: Q.931 Call State is initialized to be [Null].
Hex representation of the received TPKT
0300007408020001050404889886A56C0580373737377E005B0500B0060008914A000101400
6007000740065006C003200310033020001400F007000740065006C003200330040007A006F0
06E00650032002E0063006F006D004EC8490FB4B9D111BFAF0060B000E945000C07003200000
C06B8
    H225Lib::h225RecvData: Q.931 SETUP received from socket [2]
    H225Lib::h225RecvData: State changed to [Call Present].
Hex representation of the CALL PROCEEDING TPKT to send.
0300001B08028001027E000F050100060008914A00010880012800
    H225Lib::h225CallProcRequest: Q.931 CALL PROCEEDING sent from socket
[2]. Call state remains unchanged (Q.931 FSM simplified for H.225.0)
    H225Lib::h225TConn: connect in progress on socket [4]
    H225Lib::h225TConn: Q.931 Call State is initialized to be [Null].
Hex representation of the SETUP TPKT to send.
030000A60802008405040488988CA56C0591373737377E008D0500B8060008914A000101400
6007000740065006C0032003100332800B50000124001280001400F007000740065006C00320
0330040007A006F006E00650032002E0063006F006D006600000106B8004EC8490FB4B9D111B
FAF0060B000E945000E07006500000106B822400F007000740065006C003200330040007A006
F006E00650032002E0063006F006D
    H225Lib::h225SetupRequest: Q.931 SETUP sent from socket [4]
    H225Lib::h225SetupRequest: Q.931 Call State changed to [Call Initiated].
Hex representation of the received TPKT
0300001B08028084027E000F050100060008914A00010880012800
    H225Lib::h225RecvData: Q.931 CALL PROCEEDING received from socket [4]
Hex representation of the received TPKT
0300001808028084017E000C050300060008914A00010000
    H225Lib::h225RecvData: Q.931 ALERTING received from socket [4]
    H225Lib::h225RecvData: Q.931 Call State changed to [Call Delivered].
Hex representation of the ALERTING TPKT to send.
0300001808028001017E000C050300060008914A00010000
    H225Lib::h225AlertRequest: Q.931 ALERTING sent from socket [2]. Call

```

```
state changed to [Call Received].
Hex representation of the received TPKT
030000370802808407040388C0A57E0026050240060008914A000100660000012AFF0880012
8004EC8490FB4B9D111BFAF0060B000E945
    H225Lib::h225RecvData: Q.931 CONNECT received from socket [4]
    H225Lib::h225RecvData: Q.931 Call State changed to [Active].
Hex representation of the CONNECT TPKT to send.
0300003808028001070404889886A57E0026050240060008914A000100650000012AFC08800
128004EC8490FB4B9D111BFAF0060B000E945
    H225Lib::h225SetupResponse: Q.931 CONNECT sent from socket [2]
    H225Lib::h225SetupResponse: Q.931 Call State changed to [Active].
```

Sample 2: Source Proxy Trace with H.225 Turned On, Call Being Torn Down

This report shows two proxy call scenarios. A trace is collected on the source proxy with H.225 turned on. The call is being torn down.

```
px1#debug h225 events
H.225 Event Messages debugging is on
px1#
Hex representation of the received TPKT
0300001A080200015A080200907E000A050500060008914A0001
    H225Lib::h225RecvData: Q.931 RELEASE COMPLETE received from socket [2]
    H225Lib::h225RecvData: Q.931 Call State changed to [Null].
    H225Lib::h225RecvData: no connection on socket [2]
Hex representation of the RELEASE COMPLETE TPKT to send.
0300001A080280015A080280107E000A050500060008914A0001
    H225Lib::h225TerminateRequest: Q.931 RELEASE COMPLETE sent from socket [2]. Call
state changed to [Null].
    H225Lib::h225TClose: TCP connection from socket [2] closed
Hex representation of the RELEASE COMPLETE TPKT to send.
0300001A080280845A080280107E000A050500060008914A0001
    H225Lib::h225TerminateRequest: Q.931 RELEASE COMPLETE sent from socket [4]. Call
state changed to [Null].
    H225Lib::h225TClose: TCP connection from socket [4] closed
```

debug h245 asn1

Use the **debug h245 asn1** EXEC command to display ASN1 contents of H.245 messages. The **no** form of this command disables debugging output.

[no] debug h245 asn1

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(2)NA.

Note This command slows the system down considerably and connections may time out.

debug h245 events

Use the **debug h245 events** EXEC command to display H.245 events. The **no** form of this command disables debugging output.

[no] debug h245 events

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(2)NA.

debug proxy h323 statistics

Use the **debug proxy h323 statistics** command to enable proxy RTP statistics. The no form of this command disables the proxy RTP statistics.

[no] debug proxy h323 statistics

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(2)NA.

Enter the **show proxy h323 detail-call** command to see the statistics.

Related Commands

show proxy h323 detail-call

debug ras

Use the **debug ras** EXEC command to display RAS events. The **no** form of this command disables debugging output.

[no] debug ras

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

This command first appeared in Cisco IOS Release 11.3(2)NA.

Sample Displays

The following are sample output from the **debug ras** command.

Sample 1: Proxy Details Trace with RAS Trace Enabled

In the following reports, the proxy registers with the gatekeeper and the trace is collected on the proxy with RAS trace enabled. A report is taken from a proxy and a gatekeeper.

```
px1#debug ras
H.323 RAS Messages debugging is on
px1#
  RASLib::ras_sendto: msg length 34 sent to 40.0.0.33
  RASLib::RASSendGRQ: GRQ sent to 40.0.0.33
  RASLib::RASRecvData: successfully rcvd message of length 45 from 40.0.0.33:1719
  RASLib::RASRecvData: GCF rcvd from [40.0.0.33:1719] on sock[0x67E570]
  RASLib::ras_sendto: msg length 76 sent to 40.0.0.33
  RASLib::RASSendRRQ: RRQ sent to 40.0.0.33
  RASLib::RASRecvData: successfully rcvd message of length 81 from 40.0.0.33:1719
  RASLib::RASRecvData: RCF rcvd from [40.0.0.33:1719] on sock [0x67E570]

gk1#debug ras
H.323 RAS Messages debugging is on
gk1#
  RASLib::RASRecvData: successfully rcvd message of length 34 from 101.0.0.1:24999
  RASLib::RASRecvData: GRQ rcvd from [101.0.0.1:24999] on sock[5C8D28]
  RASLib::ras_sendto: msg length 45 sent to 40.0.0.31
  RASLib::RASSendGCF: GCF sent to 40.0.0.31
  RASLib::RASRecvData: successfully rcvd message of length 76 from 101.0.0.1:24999
  RASLib::RASRecvData: RRQ rcvd from [101.0.0.1:24999] on sock [0x5C8D28]
  RASLib::ras_sendto: msg length 81 sent to 40.0.0.31
  RASLib::RASSendRCF: RCF sent to 40.0.0.31
```

Sample 2: Gatekeeper Trace with RAS Turned On, Call Being Established

This report shows a proxy call scenario. A trace is collected on a gatekeeper with RAS turned on. The call is being established.

```
gk1#debug ras
H.323 RAS Messages debugging is on
gk1# RASLib::RASRecvData: successfully rcvd message of length 116 from 50.0.0.12:1700
RASLib::RASRecvData: ARQ rcvd from [50.0.0.12:1700] on sock [0x5C8D28]
RASLib::RAS_WK_TInit: ipsock [0x68BD30] setup successful
RASlib::ras_sendto: msg length 80 sent to 102.0.0.1
RASLib::RASSendLRQ: LRQ sent to 102.0.0.1
RASLib::RASRecvData: successfully rcvd message of length 111 from 102.0.0.1:1719
RASLib::RASRecvData: LCF rcvd from [102.0.0.1:1719] on sock [0x68BD30]
RASLib::parse_lcf_nonstd: LCF Nonstd decode succeeded, remlen = 0
RASlib::ras_sendto: msg length 16 sent to 50.0.0.12
RASLib::RASSendACF: ACF sent to 50.0.0.12
RASLib::RASRecvData: successfully rcvd message of length 112 from 101.0.0.1:24999
RASLib::RASRecvData: ARQ rcvd from [101.0.0.1:24999] on sock [0x5C8D28]
RASlib::ras_sendto: msg length 93 sent to 40.0.0.31
RASLib::RASSendACF: ACF sent to 40.0.0.31
RASLib::RASRecvData: successfully rcvd message of length 123 from 101.0.0.1:24999
RASLib::RASRecvData: ARQ rcvd from [101.0.0.1:24999] on sock [0x5C8D28]
RASlib::ras_sendto: msg length 16 sent to 40.0.0.31
RASLib::RASSendACF: ACF sent to 40.0.0.31
```

Sample 3: Gatekeeper Trace with RAS Turned On, Call Being Torn Down

This report shows two proxy call scenarios. A trace is collected on the gatekeeper with RAS turned on. The call is being torn down.

```
gk1#debug ras
H.323 RAS Messages debugging is on
gk1#
RASlib::ras_sendto: msg length 3 sent to 40.0.0.31
RASLib::RASSendDCF: DCF sent to 40.0.0.31
RASLib::RASRecvData: successfully rcvd message of length 55 from 101.0.0.1:24999
RASLib::RASRecvData: DRQ rcvd from [101.0.0.1:24999] on sock [0x5C8D28]
RASlib::ras_sendto: msg length 3 sent to 40.0.0.31
RASLib::RASSendDCF: DCF sent to 40.0.0.31
RASLib::RASRecvData: successfully rcvd message of length 55 from 50.0.0.12:1700
RASLib::RASRecvData: DRQ rcvd from [50.0.0.12:1700] on sock [0x5C8D28]
RASlib::ras_sendto: msg length 3 sent to 50.0.0.12
RASLib::RASSendDCF: DCF sent to 50.0.0.12
```

Sample 4: Source Proxy Trace with RAS Turned On, Call Being Established

This report shows two proxy call scenarios. A trace is collected on the source proxy with RAS turned on. The call is being established.

```
px1#debug ras
H.323 RAS Messages debugging is on
px1# RASlib::ras_sendto: msg length 112 sent to 40.0.0.33
RASLib::RASSendARQ: ARQ sent to 40.0.0.33
RASLib::RASRecvData: successfully rcvd message of length 93 from 40.0.0.33:1719
RASLib::RASRecvData: ACF rcvd from [40.0.0.33:1719] on sock [0x67E570]
RASLib::parse_acf_nonstd: ACF Nonstd decode succeeded, remlen = 0
RASlib::ras_sendto: msg length 123 sent to 40.0.0.33
RASLib::RASSendARQ: ARQ sent to 40.0.0.33
RASLib::RASRecvData: successfully rcvd message of length 16 from 40.0.0.33:1719
RASLib::RASRecvData: ACF rcvd from [40.0.0.33:1719] on sock [0x67E570]
```

Sample 5: Source Proxy Trace with RAS Turned On, Call Being Torn Down

This report shows two proxy call scenarios. A trace is collected on the source proxy with RAS turned on. The call is being torn down.

```
px1#debug ras
H.323 RAS Messages debugging is on
px1# RASLib::RASSendDRQ: DRQ sent to 40.0.0.33
RASLib::ras_sendto: msg length 55 sent to 40.0.0.33
RASLib::RASSendDRQ: DRQ sent to 40.0.0.33
RASLib::RASRecvData: successfully rcvd message of length 3 from 40.0.0.33:1719
RASLib::RASRecvData: DCF rcvd from [40.0.0.33:1719] on sock [0x67E570]
RASLib::RASRecvData: successfully rcvd message of length 3 from 40.0.0.33:1719
RASLib::RASRecvData: DCF rcvd from [40.0.0.33:1719] on sock [0x67E570]
```