Use voice call debug filtering on Cisco voice gatekeepers to get selected debugging traces for voice calls. This feature allows you to filter and trace voice call debug messages based on selected filtering criteria, reducing the volume of output for more efficient troubleshooting.

Guide Contents
Troubleshooting Cisco IOS Voice Overview
Debug Command Output on Cisco IOS Voice Gateways
Filtering Troubleshooting Output
Cisco VoIP Internal Error Codes
Troubleshooting Cisco IOS Voice Telephony
Troubleshooting Cisco IOS Voice Protocols
Troubleshooting Cisco IOS Telephony Applications
Monitoring the Cisco IOS Voice Network
Cause Codes and Debug Values

Contents

Restrictions for Voice Call Debug Filtering on Voice Gatekeepers

The following restrictions apply to the Cisco voice gatekeepers:

- Voice call debug filtering on gatekeepers is available only if you have a Cisco IOS image that contains the gatekeeper functionality or joint functionality (for both gateways and gatekeepers).
- Filtering of debugs in the gatekeeper DNS process is not supported.
- Filtering of ASN and GUP messages is not supported.
- The following are not supported by call filtering:
  - If debug messages are printed in non-ARQ path, the path will not support call filtering.
  - H.323V1, H323 proxy, and DGK are not supported.
  - Calls transferred from primary gatekeeper to the secondary gatekeeper will not be filtered in the secondary gatekeeper, but new calls in secondary gatekeeper will apply filtering.
  - All error, incorrect, and delayed GKTMP messages are not printed in call filtering because the gatekeeper ignores the message at a lower level and the messages are not printed.
  - GKTMP messages that are not related to calls are not supported by call filtering (registration, unregistration, and resource messages).

Information About Voice Call Debug Filtering

Information from using debug commands for voice calls is crucial for troubleshooting, but the volume of raw data can be very large. In order to isolate the most valuable data, you can use the Voice Call Debug Filtering feature on gatekeepers. Debug output for the voice gatekeepers can be filtered according to the following criteria:

- Incoming called number
- Incoming calling number
- Gatekeeper resolved final destination endpoint (IPv4) address

Note: In addition to working on voice gatekeepers, call filtering works on IP-to-IP gateway connections using H.323.

The selected criteria are set on the gatekeeper, and different sets of criteria can be stored.

To better understand the voice call debug filtering on Cisco voice gatekeepers, see the following sections:

- Debug Commands that Support Voice Call Filtering on Cisco Voice Gatekeepers
- Gatekeeper Filter Module
- Calling and Called Number Strings
- Exact and Partial Matching Conditions Rules and Information

Debug Commands that Support Voice Call Filtering on Cisco Voice Gatekeepers

When a call filter is applied, the filtering applies to all of the debugs affected by the call filter. Debug commands that support voice call debug filtering on gatekeepers are the following:

- **debug gatekeeper call** *level*
- **debug gatekeeper main** level
- **debug gatekeeper servers**

Note:

In the syntax, the level argument can be a number from 1 through 10. Refer to the Cisco IOS Debug Command Reference for detailed information about these debug commands.

Gatekeeper Filter Module

The filtering for these modules is managed by the gatekeeper filter module (GKFM). The filtering conditions are configured in the GKFM, and then the individual modules are informed when a call has to be filtered. The GKFM coordinates between multiple modules to handle filtering conditions.

The GKFM provides the core infrastructure for the gatekeeper module to use for providing call tracing capabilities. This module is part of the GCFM subsystem (sub_gcfm) and provides all the necessary implementations for the command-line interface (CLI) functions, conditions matching logics for the gatekeeper.

The gatekeeper debug subsystem uses the GKFM infrastructure to implement standard style header formats.

Activity in the GKFM can be traced using the **debug call filter detail** and **debug call filter inout** commands. Refer to the Cisco IOS Debug Command Reference for more information about these debug commands.

The GKFM functionalities are summarized as follows:

- Maintain a set of matching lists that are activated through the CLI interface for filtering out desired calls.
- Provide an interface to the gatekeeper modules for filter query for exact and partial matches.
- Implement an algorithm to execute the matching logic by examining the activated filter tags provisioned through the CLI interface. The algorithm is optimized in a way that the logic is executed only when there is a new call parameter populated in the context or there is a change in the CLI provisioned data.
- Employ a filter rules engine to prevent activation of illegal and conflicting filter tags (that is, filter tags having conflicting conditions provisioned across them).

Calling and Called Number Strings

The string pattern for calling and called numbers can be either a complete telephone number or a partial telephone number with wildcard digits, represented by a period (.) character. Each "." represents a wildcard for an individual digit that the originating voice gateway expects to match. For example, if the calling and called number strings is defined as "555....", then any dialed string beginning with 555, plus at least four additional digits, matches this calling or called number.

Table: Wild CardSymbols Supported in Calling and Called Number Strings shows all of the wildcard symbols that are supported in the calling and called number strings.

Table: Wild CardSymbols Supported in Calling and Called Number Strings

Symbol	Description
.	Indicates a single-digit placeholder. For example, 555.... matches any dialed string beginning with 555, plus at least four additional digits.
[ ]	Indicates a range of digits. A consecutive range is indicated with a hyphen (-); for example, [5-7]. A nonconsecutive range is indicated with a comma (,); for example, [5,8]. Hyphens and commas can be used in combination; for example, [5-7,9]. **Note**: Only single-digit ranges are supported. For example, [98-102] is invalid.

( )	Indicates a pattern; for example, 408(555). It is used in conjunction with the symbol ?, %, or +.
?	Indicates that the preceding digit occurred zero or one time. Press **ctrl-v** before entering ? from your keyboard.
%	Indicates that the preceding digit occurred zero or more times. This functions the same as the "*" used in regular expression.
+	Indicates that the preceding digit occurred one or more times.
T	Indicates the interdigit timeout. The voice gateway pauses to collect additional dialed digits.

Note: The wildcard symbols follow regular expression rules, and whatever wildcard applies to the gateway applies to the gatekeeper as well. The period (.) is the only wildcard character that is supported for dial strings that are configured using the **answer-address** or **incoming called-number** command.

Exact and Partial Matching Conditions Rules and Information

To filter out desired calls, you must define a list of matching conditions. There must be separate filters defined for VoIP gateways and VoIP gatekeepers, because both the gateway and gatekeeper can exist on the same box.

Multiple filter tags could be provisioned for the gatekeeper, each with one or all three different conditions under them. But the GKFM provides a rules engine to validate while activating thefilter tags when one or more filter tags are contradictory or conflicting with each other. Matching conditions are as follows:

- Exact match-All related debug output is filtered and shown when all conditions in the match list are explicitly met. This is the best choice for most situations because the output is the most concise.
- Partial match-Related debug output is filtered and shown when even only one condition matches. This choice is useful in debugging call startup problems like digit collection, but is not ideal for many situations because of the large amount of debug output that might be generated before matches explicitly fail.

To filter out the desired calls, you must define a list of matching conditions. Separate filters need to be defined for VoIP gateways and VoIP gatekeepers, because both the gateway and gatekeeper can exist on the same box.

The following global CLI is used to define the conditions on the gatekeeper. The gatekeeper keyword appears in the CLI only if you have a Cisco IOS image that contains gatekeeper debug filter functionality or a combination of gateway and gatekeeper debug filter functionality.

call filter match-list number voice gatekeeper

Then, under the submode, a set of conditions can be defined. Only the following will be valid for filtering calls on the gatekeeper:

incoming calling-number string incoming called-number string

One new match condition is added that is valid for gatekeeper filtering only:

gatekeeper resolved ipv4 ipaddress

For applying the match filters configured, the CLI for gatekeepers is the same as that used for gateways:

debug condition match-list <1-16> exact-match | partial-match

Table: Wild CardSymbols Supported in Calling and Called Number Strings

Table: Rules Applied for Executing Exact and Partial Matching Logic summarizes the rules applied while you are executing the exact and partial match logic.

Table: Rules Applied for Executing Exact and Partial Matching Logic

Number	Decription

1	The exact match is an AND logic across all the provisioned conditions. For example:
	
	`call filter match-list 2 gatekeeper`
	` incoming called-number 9876`
	` incoming calling-number 2345`
	` resolved destination ipv4 1.2.3.4`
	
	The debugs are emitted only when all the callp provided input matches.
	
	IC Called Number IC Calling Number Resolved destination IP address

2	The partial match is a logical OR across all the provisioned conditions. For example:
	
	`call filter match-list 2 gatekeeper`
	` incoming called-number 9876`
	` incoming calling-number 2345`
	` resolved destination ipv4 1.2.3.4`
	
	The debugs are emitted only when any one of the callp-provided inputs matches.
	
	`IC Called Number || IC Calling Number || Resolved destination IP address`

3	When the admin provisions a set of conditions under a filter tag that may not contain all three (supported) conditions, then the rules are applied only on that subset of conditions. The non-provisioned conditions are considered as wildcard or don't care. For example:
	
	`call filter match-list 2 gatekeeper`
	` incoming called-number 9876`
	` resolved destination ipv4 1.2.3.4`
	
	Because incoming calling number was not provided, it is taken as a don't care condition and excluded when the GKFM is executing the matching logic. This tag shall be an exact match for all calls with called number 9876, resolved destination 1.2.3.4, and any calling number xxxx.

4	When the admin provisions only the resolved destination address under a filter tag, then all debugs (for which filtering is employed) shall be throwing the outputs until the callp learns the resolved destination address (that is, until the RAS-LCF message was obtained or the route server provides the resolved address during a later part of the call). This scenario should be well understood before such a filter tag is activated. For example:
	
	`call filter match-list 1 gatekeeper`
	`  resolved destination ipv4 1.2.3.4`
	
	Then debugs for all the calls with any IC called/calling numbers shall be thrown until the resolved destination address is known. When the resolved destination address happens to be 1.2.3.4, then the debug outputs would continue-if not, it would stop at that point.

5	If the admin provisions a filter tag with no matching conditions under it, such a tag cannot be activated. For example:
	
	`!`
	`call filter match-list 3 gatekeeper`
	`!`

	`#deb condition match-list 3 exact`
	
	`Failed: Activation of tag with no filter condition is not allowed.`

6	If an activated filter tag condition is modified, that tag needs to be explicitly reactivated after any changes. For example:
	
	`!`
	`call filter match-list 2 gatekeeper`
	`incoming called-number 9876`
	`incoming calling-number 2345`
	`resolved destination ipv4 1.2.3.4`
	`!`
	`# show call filter match-list`
	`**********`
	`call filter match-list 2 gatekeeper`
	`**********`
	
	`incoming called-number 9876`
	`incoming calling-number 2345`
	`gatekeeper resolved destination address 1.2.3.4`
	`debug condition match-list is set to EXACT_MATCH`
	`!`
	`# conf t`
	`(config)# call filter match-list 2 gatekeeper`
	`(conf-call-filter-mlist)# incoming calling-number 4089`
	`(conf-call-filter-mlist)# end`
	`# # show call filter match-list`
	`**********`
	`call filter match-list 2 gatekeeper`
	`**********`
	`incoming called-number 9876`
	`incoming calling-number 2345`
	`gatekeeper resolved destination address 1.2.3.4`
	`debug condition match-list is not ARMED`

7	A call for which the GKFM decides failed match (suppress) does not result in any debugs or reexecution of matching logic during the lifespan of that call, even though the CLI admin modifies the provisioned conditions during the lifespan of that call.

GKFM Debug Filter Rules Engine

The following are the rules employed by the filter activation rules engine in order to avoid the conflicts between one or more activated filter tags.

Y- indicates configured

X- indicates not configured (don't care)

1. New condition pattern to be activated-CDN:Y, CGN:X, RES_IPV4:X

Existing activated filter patterns

Activated CDN	Activated CGN	Activated IPV4	Employed Rules
Y	X	X	Compare CDN only. Reject - if same Allow - if different
X	X	Y	Reject always.
X	Y	X	Reject always.
X	Y	Y	Reject always.
Y	X	Y	Compare CDN only. Reject - if same Allow - if different

Table: Rules Applied for Executing Exact and Partial Matching Logic

Y	Y	X	Compare CDN only. Reject - if same Allow - if different
Y	Y	Y	Compare CDN only. Reject - if same Allow - if different

2. New condition pattern to be activated-CDN:X, CGN:Y, RES_IPV4:X

Existing activated filter patterns

Activated CDN	Activated CGN	Activated IPV4	Employed Rules
X	Y	X	Compare CGN only. Reject - if same Allow - if different
X	Y	Y	Compare CGN only. Reject - if same Allow - if different
Y	Y	X	Compare CGN only. Reject - if same Allow - if different
Y	Y	Y	Compare CGN only. Reject - if same Allow - if different
X	X	Y	Reject always.
Y	X	X	Reject always.
Y	X	Y	Reject always.

3. New condition pattern to be activated-CDN:Y, CGN:Y, RES_IPV4:X

Existing activated filter patterns

Activated CDN	Activated CGN	Activated IPV4	Employed Rules
Y	Y	X	Compare CDN and CGN. Reject - if both are same Allow - if anyone different
Y	Y	Y	Compare CDN and CGN only. Reject - if both are same Allow - if anyone different
X	Y	X	Compare CGN only. Reject - if same Allow - if different
Y	X	X	Compare CDN only. Reject - if same Allow - if different
X	X	Y	Reject always.
X	Y	Y	Compare CGN only. Reject - if same Allow - if different
Y	X	Y	Compare CDN only. Reject - if same Allow if different

4. New condition pattern to be activated-CDN:X, CGN:X, RES_IPV4:Y

Existing activated filter patterns

Activated CDN	Activated CGN	Activated IPV4	Employed Rules
X	X	Y	Compare Res. IPv4 only. Reject - if same Allow - if different
Y	X	Y	Compare Res. IPv4 only. Reject - if same Allow - if different
X	Y	Y	Compare Res. IPv4 only. Reject - if same Allow - if different
Y	Y	Y	Compare Res. IPv4 only. Reject - if same Allow - if different
X	Y	X	Reject always.
Y	X	X	Reject always.
Y	Y	X	Reject always.

5. New condition pattern to be activated-CDN:Y, CGN:X, RES_IPV4:Y

Existing activated filter patterns

Activated CDN	Activated CGN	Activated IPV4	Employed rules
Y	X	Y	Compare CDN and Res. IPv4. Reject - if both are same Allow - if anyone different
Y	Y	Y	Compare CDN and Res. IPv4. Reject - if both are same Allow - if anyone different.
Y	X	X	Compare CDN only. Reject - if same Allow - if different
Y	Y	X	Compare CDN only. Reject - if same Allow - if different
X	X	Y	Compare IPv4 only. Reject - if same Allow - if different
X	Y	Y	Compare IPv4 only. Reject - if same Allow - if different
X	Y	X	Reject always.

6. New condition pattern to be activated-CDN:X, CGN:Y, RES_IPV4:Y

Existing activated filter patterns

Activated CDN	Activated CGN	Activated IPV4	Employed Rules
X	Y	Y	Compare CGN and Res. IPv4. Reject - if both are same Allow - if anyone different
Y	Y	Y	Compare CGN and Res. IPv4. Reject - if both are same Allow - if anyone different.
X	X	Y	Compare Res. IPv4 only. Reject - if same Allow - if different
Y	X	Y	Compare Res. IPv4 only. Reject - if same Allow - if different
X	Y	X	Compare CGN only. Reject - if same Allow - if different
Y	Y	X	Compare CGN only. Reject - if same Allow - if different
Y	X	X	Reject always.

7. New condition pattern to be activated-CDN:Y, CGN:Y, RES_IPV4:Y

Existing activated filter patterns

Activated CDN	Activated CGN	Activated IPV4	Employed Rules
Y	Y	Y	Compare CDN, CGN and Res. IPv4. Reject - if all are same. Allow - if any one different
X	X	Y	Compare Res. IPv4 only. Reject - if same Allow - if different.
X	Y	X	Compare CGN only. Rejectect - if same Allow - if different
X	Y	Y	Compare CGN & Res. IPv4. Rejectect - if both are same Allow - if any one different
Y	X	X	Compare CDN only. Reject - if same Allow - if different
Y	X	Y	Compare CDN and Res. IPv4 only. Reject - if both are same Allow - if any one different

Y	Y	X	Compare CDN and CGN only. Reject - if both are same Allow - if any one different.

Configuring the Voice Call Debug Filter for Cisco Gatekeepers

To configure the voice call debug filter on Cisco gatekeepers, perform the following tasks:

- Configuring Call-Specific Conditions for Gatekeepers (required)
- Enabling Debug for the Set Filtering Conditions (required)

Configuring Call-Specific Conditions for Gatekeepers

Configure call-specific conditions to set the attributes that are filtered for voice calls.

SUMMARY STEPS

1. **enable**
2. **configure** terminal
3. **call filter match-list** *number* **gatekeeper**
4. **incoming calling-number** *string*
5. **incoming called-number** *string*
6. **gatekeeper resolved ipv4** *ipaddress*
7. **exit**
8. **exit**

DETAILED STEPS

	Command or Action	Purpose
1.	**enable** Example: `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
2.	**configure** terminal Example: `Router# configure terminal`	Enters global configuration mode.
3.	**call filter match-list** *number* **gatekeeper** Example: `Router(config)# call filter match-list 1 gatekeeper`	Enters call filter match list configuration mode to define the filter conditions on the gatekeeper. • *number*-Numeric label that uniquely identifies the match list. Range is 1 to 16. **Note**: At least one of the following optional parameters (Step 4 to Step 7) for call filtering must be configured.

4.	**incoming calling-number** *string* Example: `Router(conf-call-filter-mlist)# incoming calling-number 408525`	(Optional) Specifies the incoming calling number to be filtered. • *string*-Numeric string that identifies all or part of the incoming calling number.
5.	**incoming called-number** *string* Example: `Router(conf-call-filter-mlist)# incoming called-number 408525`	(Optional) Specifies the incoming called number to be filtered. • *string*-Numeric string that identifies all or part of the incoming called number.
6.	**gatekeeper resolved ipv4** *ipaddress* Example: `Router(conf-call-filter-mlist)# gatekeeper resolved ipv4 10.1.1.1`	(Optional) Specifies the gatekeeper-resolved destination IP address to be filtered. • ipaddress-IP address that identifies the destination.
7.	**exit** Example: `Router(conf-call-filter-mlist)# exit`	Exits to global configuration mode.
8.	**exit** Example: `Router(config)# exit`	Exits to privileged EXEC mode.

Enabling Debug for the Set Filtering Conditions

Use the **debug** commands to enable the set conditions to get the filtered output.

Prerequisites

The conditions for the voice call debug filter must be set as described in the Configuring Call-Specific Conditions for Gatekeepers.

SUMMARY STEPS

1. **enable**
2. **debug condition match-list** *tag* {**exact-match** | **partial-match**}
3. **debug gatekeeper call** level

   or
   debug gatekeeper main level
   or
   debug gatekeeper servers

DETAILED STEPS

Command or Action	Purpose

1.	**enable** Example: `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
2.	**debug condition match-list** *tag* {**exact-match** | **partial-match**} Example: `Router# debug condition match-list 1 exact-match`	Enables the filter match list for the set conditions. • *tag*-Numeric label that uniquely identifies the match list. Range is 1 to 16. The number for the match list is set using the **call filter match-list** command. • **exact-match**-All related debug output is filtered until all conditions in the match list are explicitly met. This is the best choice for most situations because the output is the most concise. • **partial-match**-No related debug output is filtered until there is a single explicit match failure. As long as zero or more conditions are met, debug output is not filtered. This choice is useful in debugging call startup problems like digit collection, but is not ideal for many situations because a large amount of debug output is generated before matches explicitly fail.

3.	**debug gatekeeper call** level
	
	or **debug gatekeeper main** level
	
	or **debug gatekeeper servers**
	
	Example:
	
	`Router# debug gatekeeper call 5`
	
	Example:
	
	`Router# debug gatekeeper main 3`
	
	Example:
	
	`Router# debug gatekeeper servers`
		Enables the appropriate gatekeeper call debug commands.
		
		• level-Specifies a level for the debug message. Entries can be 1 through 10.
		• Refer to the Cisco IOS Debug Command Reference for detailed descriptions of these debug commands.
		• The debug output commences at this point.

Troubleshooting Tips

To verify debug conditions, use the following commands:

- **show debug**

  This command displays the debugs that are enabled.

- **show call filter components**

  This command displays the components that register internally with the filtering module. This command shows which components are registered with the GKFM, which is the internal module that controls which components are filtered.

- **show call filter match-list**

  This command displays the criteria set for the specified match list. It shows a list of all the match lists, shows which ones are enabled, and shows whether they are enabled for partial or exact matching.

Examples

This section provides sample output and a typical debug listing:

Sample Output for show call filter and show debug: Example

When the exact match condition is used for gatekeeper voice call debug filtering, all related debug output is filtered until all conditions in the match list are explicitly met:

```
Router# show call filter match-list
*****************************************
call filter match-list 1 gatekeeper
*****************************************
gatekeeper resolved destination address 10.77.154.91
incoming called-number 444
debug condition match-list is not armed
*****************************************
call filter match-list 2 gatekeeper
*****************************************
incoming called-number 204
debug condition match-list is set to EXACT_MATCH
Router#
##############################
Router# show debug
Gatekeeper:
Gatekeeper Server Messages debugging is on (filter is ON)
gk call debug level = 10 (filter is ON)
gk zone debug level = 10
c2691-1-OGK#
```

Sample Output for show debug: Example

Following is a sample output resulting from the **show debug** command:

```
*Mar 2 03:12:04.259: //C1989D10805C/C1989D10805D/GK/rassrv_arq_select_viazone: matched zone is GK
*Mar 2 03:12:04.259: //C1989D10805C/C1989D10805D/GK/rassrv_arq_select_viazone: about to check the
*Mar 2 03:12:04.259: //C1989D10805C/C1989D10805D/GK/rassrv_arq_select_viazone: matched zone is GK
*Mar 2 03:12:04.259: //C1989D10805C/C1989D10805D/GK/rassrv_get_addrinfo: No tech prefix
*Mar 2 03:12:04.259: //C1989D10805C/C1989D10805D/GK/rassrv_get_addrinfo: Alias not found
*Mar 2 03:12:04.259: //C1989D10805C/C1989D10805D/GK/rassrv_get_addrinfo: (203) unknown address ar
```