Embedded Syslog Manager Configuration Guide, Cisco IOS XE Fuji 16.8.x

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED "AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

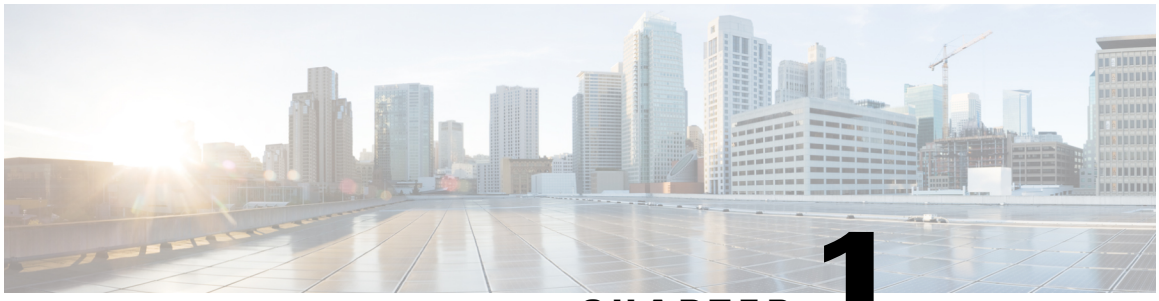
IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: https://www.cisco.com/go/trademarks. Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1721R)

© 2018 Cisco Systems, Inc. All rights reserved.

CONTENTS



CHAPTER 1

Read Me First

Important Information about Cisco IOS XE 16

Effective Cisco IOS XE Release 3.7.0E (for Catalyst Switching) and Cisco IOS XE Release 3.17S (for Access and Edge Routing) the two releases evolve (merge) into a single version of converged release—the Cisco IOS XE 16—providing one release covering the extensive range of access and edge products in the Switching and Routing portfolio.

Feature Information

Use Cisco Feature Navigator to find information about feature support, platform support, and Cisco software image support. An account on Cisco.com is not required.

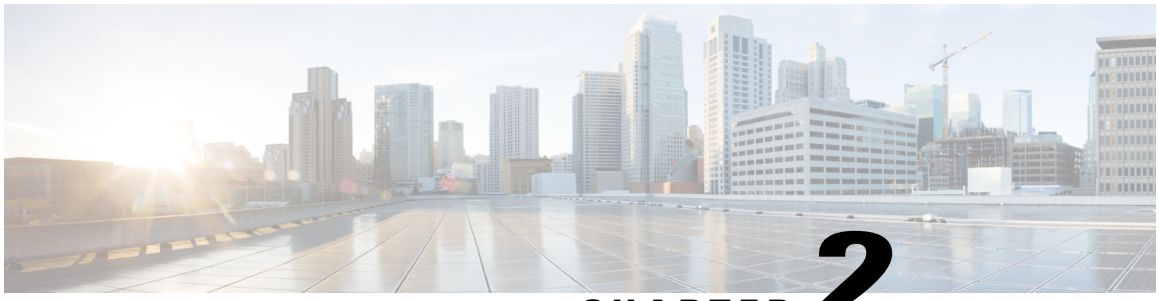
Related References

- Cisco IOS Command References, All Releases

Obtaining Documentation and Submitting a Service Request

For information on obtaining documentation, using the Cisco Bug Search Tool (BST), submitting a service request, and gathering additional information, see What's New in Cisco Product Documentation.

To receive new and revised Cisco technical content directly to your desktop, you can subscribe to the What's New in Cisco Product Documentation RSS feed. RSS feeds are a free service.



CHAPTER 2

Embedded Syslog Manager (ESM)

The Embedded Syslog Manager (ESM) feature provides a programmable framework that allows you to filter, escalate, correlate, route, and customize system logging messages prior to delivery by the system message logger.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table at the end of this module.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Restrictions for Embedded Syslog Manager

Embedded Syslog Manager (ESM) depends on the Tcl 8.3.4 subsystem, because ESM filters are written in Tool Command Language (Tcl). ESM is available only in images that support Tcl version 8.3.4 or later versions. Support for Tcl 8.3.4 is added depending on your release.

ESM filters are written in Tcl.

ESM filtering cannot be applied to SNMP “history□? logging. Therefore, ESM filtering will not be applied to messages logged using the **logging history** and **snmp-server enable traps syslog** commands.

Information About the Embedded Syslog Manager

System Message Logging

With the introduction of the Embedded Syslog Manager, system messages can be logged independently as standard messages, XML-formatted messages, or ESM filtered messages. These outputs can be sent to any of the traditional syslog targets. For example, you could enable standard logging to the console connection, XML-formatted message logging to the buffer, and ESM filtered message logging to the monitor. Similarly, each type of output could be sent to different remote hosts. A benefit of separate logging processes is that standard logging will not be affected if, for example, there is some problem with the ESM filter modules.

System Logging Message Formatting

System logging messages are displayed in the following format:

%<facility>-<severity>-<mnemonic>: <message-text>

The following is an example of a system logging message:

```
%LINK-5-CHANGED: Interface Serial3/3, changed state to administratively down
```

Usually, these messages are preceded by additional text, such as the error sequence number and time stamp:

<sequence-number>: <time stamp>:%<facility>-<severity>-<mnemonic>: <message-text>

The following is an example of a system logging message preceded by an error sequence number and time stamp:

```
000013: Mar 18 14:52:10.039:%LINK-5-CHANGED: Interface Serial3/3, changed state
to administratively down
```

Note The time stamp format used in system logging messages is determined by the **service timestamps** global configuration mode command. The **service sequence-numbers** global configuration command enables or disables the leading sequence number. An asterisk (*) before the time indicates that the time may be incorrect because the system clock has not synchronized to a reliable time source.

Benefits of Embedded Syslog Manager

The Embedded Syslog Manager (ESM) is a feature integrated in Cisco software that allows complete control over system message logging at the source. ESM provides a programmatic interface to allow you to write custom filters that meet your specific needs relating to system logging. Benefits of this feature are:

- Customization--Fully customizable processing of system logging messages, with support for multiple, interfacing syslog collectors.

- Severity escalation for key messages--The ability to configure your own severity levels for syslog messages instead of using the system-defined severity levels.
- Specific message targeting--The ability to route specific messages or message types, based on type of facility or type of severity, to different syslog collectors.
- SMTP-base e-mail alerts--Capability for notifications using TCP to external servers, such as TCP-based syslog collectors or Simple Mail Transfer Protocol (SMTP) servers.
- Message limiting--The ability to limit and manage syslog "message storms" by correlating device-level events.

The ESM is not a replacement for the UDP-based syslog mechanism; instead, it is an optional subsystem that can operate in parallel with the current system logging process. For example, you can continue to have the original syslog message stream collected by server A, while the filtered, correlated, or otherwise customized ESM logging stream is sent to server B. All of the current targets for syslog messages (console, monitor, buffer, and syslog host list) can be configured to receive either the original syslog stream or the ESM stream. The ESM stream can be further divided into user-defined streams and routed to collectors accordingly.

Syslog Filter Modules

Embedded Syslog Manager (ESM) uses syslog filter modules to process system logging messages. Syslog filter modules are scripts written in the Tool Command Language (Tcl) stored in local system memory or on a remote file server. The ESM is customizable because you can write and reference your own scripts.

Syslog filter modules can be written and stored as plain-text files or as precompiled files. Tcl script pre-compiling can be done with tools such as TclPro. Precompiled scripts allow a measure of security and managed consistency because they cannot be edited.

Note Because Tcl script modules contain executable commands, you should manage the security of these files using the same processes you use to manage configuration files.

How to Use the Embedded Syslog Manager

Writing ESM Syslog Filter Modules

Before referencing syslog filter modules in the Embedded Syslog Manager (ESM) configuration, you must write or obtain the modules you want to apply to system logging messages. Syslog filter modules can be stored in local system memory, or on a remote file server. Before you write syslog filter modules, you should understand the following concepts:

ESM Filter Process

When ESM is enabled, all system logging messages are processed through the referenced syslog filter modules. Syslog filter modules are processed in their order in the filter chain. The position of a syslog filter module in the filter chain is determined by the position tag applied in the **logging filter** global configuration mode

command. If a position is not specified, the modules are processed in the order in which they were added to the configuration.

The output of each filter module is used as the input for the next filter module in the chain. Therefore, the Tcl global variable containing the original syslog message (::orig_msg) is set to the return value of each filter before invoking the next filter in the chain. Thus, if a filter returns NULL, no message will be sent out to the ESM stream. Once all filters have processed the message, the message is queued for distribution by the logger.

The console, buffer, monitor, and syslog hosts can be configured to receive a particular message stream (normal, XML, or ESM). The syslog hosts can be further restricted to receive user-defined numbered streams. Each target examines each message and accepts or rejects the message based on its stream tag. ESM filters can change the destination stream by altering the messages' stream tag by changing the Tcl global variable "::stream."

Syslog Filter Module Input

When Embedded Syslog Manager (ESM) is enabled, system logging messages are sent to the logging process. Each data element in the system logging message, and in the formatted syslog message as a whole, is recorded as a Tcl global variable. The data elements format for the syslog message are as follows:

<sequence-number>: <time stamp>:%<facility>-<severity>-<mnemonic>: <message-text>

The message-text will often contain message arguments.

When messages are received on a syslog host a "syslog-count" number is also added:

<syslog-count>: <sequence-number>: <time stamp>:%<facility>-<severity>-<mnemonic>: <message-text>

The following examples shows the syslog-count number included in the beginning of the sequence:

The table below lists the Tcl script input variables used in syslog filter modules. The syslog message data that the filter must operate on is passed as Tcl global namespace variables. Therefore, variables should be prefixed by a double-colon within the script module.

Standard ESM Filter Processing

Each time a system logging message is generated, the syslog filter modules are called in a series. This series is determined by the ::module_position variable, which in turn is typically the order in which the modules are referenced in the system configuration (the order in which they are configured).

The output of one filter module becomes the input to the next. Because the input to the filters is the Tcl global namespace variables, each filter can change any or all of these variables depending upon the purpose of the filter.

The only Tcl global variables that are automatically updated by the Embedded Syslog Manager (ESM) framework between subsequent filter executions are the ::orig_msg and ::cli_args variables. The framework automatically sets the value of ::orig_msg to the return value of the filter module. Thus a filter that is designed to alter or filter the original message must not manually set the value for the ::orig_msg variable; the filter only needs to return the desired value. For example, the following one-line ESM filter

```
return "This is my new syslog message."
```

would ignore any message passed to it, and always change the output to the constant string "This is my new syslog message." If the module was the last filter in the chain, all ESM targets would receive this string as the final syslog message.

The one-line ESM filter

```
return “”
```

would block all syslog messages to the ESM stream. For example, the line

```
return $::orig_msg
```

would do nothing but pass the message along to the next filter in the chain. Thus, an ESM filter designed to suppress unwanted messages would look something like this:

```
if { [my_procedure_to_check_this_message] == 1 } {
     return $::orig_msg
} else {
     return “”
}
```

Depending upon their design, some filters may not use the ::orig_msg variable at all, but rather reconstruct a syslog message from its data elements (using ::format_string, ::msg_args, ::timestamp, and so on). For example, an XML tagging filter will tag the individual data elements, and disregard the original formatted message. It is important for such modules to check the ::orig_msg variable at the beginning of the Tcl script, so that if a previous filter indicated that the message should not be sent out (::orig_msg is NULL), the message would not be processed, but return NULL also.

Commands can also be added to syslog filter modules using the **exec** and **config** Tcl commands. For example, if you wanted to add the source IP address to the syslog messages, and syslog messages were configured to be sent from the Ethernet 2/0 interface (using the **logging source-interface** command) you could issue the **show interface Ethernet 2/0** command during the module initialization by using the **exec** Tcl command within the script:

```
set source_ip_string [exec show ip int E2/0 | inc Internet]
puts $source_ip_string
"  Internet address is 10.4.2.63/24"
```

Background ESM Filter Processing

In Tcl, commands can be queued for future processing by using the **after** Tcl command. The most common use of this command is to correlate (gather and summarize) events over a fixed interval of time, called the “correlation window.” Once the window of interest expires, the filter will need to “wake up,” and calculate or summarize the events that occurred during the window, and often send out a new syslog message to report the events. This background process is handled by the ESM Event Loop process, which allows the Tcl interpreter to execute queued commands after a certain amount of time has passed.

If your syslog filter module needs to take advantage of correlation windows, it must use the **after** Tcl command to call a summary procedure once the correlation window expires (see examples in the "Configuration Examples for the Embedded Syslog Manager" section). Because there is no normal filter chain processing when background processes are run, in order to produce output these filters must use one of two ESM Tcl extensions: **errmsg** or **esm_errmsg**.

During background processing, the commands that have been queued by the **after** command are not run in the context of the filter chain (as in normal processing), but rather are autonomous procedures that are executed in series by the Tcl interpreter. Thus, these background procedures should not operate on the normal Tcl global namespace variables (except for setting the global namespace variables for the next filter when using **esm_errmsg**), but should operate on variables stored in their own namespace. If these variables are declared outside of a procedure definition, they will be persistent from call to call.

The purpose of the **errmsg** Tcl command is to create a new message and send it out for distribution, bypassing any other syslog filter modules. The syntax of the **errmsg** command is:

```
errmsg <severity> <stream> <message_string>
```

The purpose of the **esm_errmsg** Tcl command is to create a new message, process it with any syslog filter modules below it in the filter chain, and then send it out for distribution. The syntax of the **esm_errmsg** command is:

```
esm_errmsg <module_position>
```

The key difference between the errmsg() Tcl function and the esm_errmsg() Tcl function is that **errmsg** ignores the filters and directly queues a message for distribution, while **esm_errmsg** will send a syslog message down the chain of filters.

In the following example, a new syslog message is created and sent out tagged as Alert severity 1 to the configured ESM logging targets (stream 2). The purpose of this filter is to suppress the individual SYS-5-CONFIG messages over a thirty minute correlation window, and send out a summary message at the end of the window.

```
errmsg 1 2 “*Jan 24 09:34:02.539: %SYS-1-CONFIG_I: There have been 12
configuration changes to the router between Jan 24 09:04:02.539 and Jan 24
09:34:01.324”
```

In order to use **esm_errmsg**, because the remaining filters following this one will be called, this background process must populate the needed Tool Command Language (Tcl) global namespace variables prior to calling **esm_errmsg**. Passing the ::module_position tells the ESM framework which filter to start with. Thus, filters using the **esm_errmsg** command should store their ::module_position (passed in the global namespace variables during normal processing) in their own namespace variable for use in background processing. Here is an example:

```
proc ::my_filter_namespace::my_summary_procedure{}
{
 set ::orig_msg “*Jan 24 09:34:02.539: %SYS-1-CONFIG_I: There have been 12
configuration changes to the router between Jan 24 09:04:02.539 and Jan 24
09:34:01.324”
 set ::timestamp “*Jan 24 09:34:02.539”
 set ::severity 1
 set ::stream 2
 set ::traceback “”
 set ::pid “”
set ::process “”
set ::format_string “There have been %d configuration changes to the router
between %s and %s”
set ::msg_args {12 “Jan 24 09:04:01.539” “Jan 24 09:34:01.324”}
esm_errmsg $::my_filter_namespace::my_module_position
}
```

The benefit of setting all the global namespace variables for the **esm_errmsg** command is that your filters will be modular, and the order they are used in the ESM framework will not matter. For example, if you want all of the messages destined for the ESM targets to be suffixed with the message originator’s hostname, you could write a one-line “hostname” filter and place it at the bottom of the filter chain:

```
return “$::orig_msg -- $::hostname”
```

In this example, if any of your filters generate new messages during background processing and they use **esm_errmsg** instead of **errmsg**, these messages will be clearly suffixed with the hostname.

What to Do Next

After creating your syslog filter module, you should store the file in a location accessible to the device. You can copy the file to local system memory, or store it on a network file server.

Configuring the Embedded Syslog Manager

To configure the Embedded Syslog Manager (ESM), specify one or more filters to be applied to generated syslog messages, and specify the syslog message target.

Before You Begin

One or more syslog filter modules must be available to the device.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **logging filter** *filter-url* [*position*] [**args** *filter-arguments*]
4. Repeat Step 3 for each syslog filter module that should be applied to system logging output.
5. Enter one of the following:
   - **logging** [**console** | **buffered** | **monitor**] **filtered** [*security-level*]
   - or
   - **logging host** {*ip-address* | *hostname*} **filtered** [**stream** *stream-id*]
6. Repeat Step 5 for each desired system logging destination.
7. **logging source-interface** *type number*
8. **logging origin-id** {**hostname** | **ip** | **ipv6** |**string** *user-defined-id*}
9. **end**
10. **show logging**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**logging filter** *filter-url* [*position*] [**args** *filter-arguments*] **Example:** `Device(config)# logging filter slot0:/escalate.tcl 1 args CONFIG_I 1`	Specifies one or more syslog filter modules to be applied to generated system logging messages. • Repeat this command for each syslog filter module that should be used. • The *filter-url* argument is the Cisco IOS File System location of the syslog filter module (script). The location can be in local memory, or a remote server using **tftp:**, **ftp:**, or **rcp:**. • The optional *position* argument specifies the order in which the syslog filter modules should be executed. If this argument is omitted, the specified module will be positioned as the last module in the chain. • Filters can be reordered quickly by again entering the **logging filter** command and specifying a different position. • The optional **args** *filter-arguments* syntax can be added to pass arguments to the specified filter. Multiple arguments can be specified. The number and type of arguments should be defined in the syslog filter module. For example, if the syslog filter module is designed to accept a specific e-mail address as an argument, you could pass the e-mail address using the **args user@host.com** syntax. Multiple arguments are typically delimited by spaces. • To remove a module from the list of modules to be executed, use the **no** form of this command.
Step 4	Repeat Step 3 for each syslog filter module that should be applied to system logging output.	--

Step 5	Enter one of the following:
	
	• **logging** [**console** | **buffered** | **monitor**] **filtered** [*security-level*]
	
	• or
	
	• **logging host** {*ip-address* | *hostname*} **filtered** [**stream** *stream-id*]
	
	Example:
	
	`Device(config)# logging console filtered informational`
	
	Example:
	
	`Device(config)# logging host 209.165.200.225 filtered stream 20`
		Specifies the target for ESM filtered syslog output.
		
		• ESM filtered syslog messages can be sent to the console, a monitor (TTY and Telnet connections), the system buffer, or remote hosts.
		
		• The optional *level* argument limits the sending of messages to those at or numerically lower than the specified value. For example, if level **1** is specified, only messages at level 1 (alerts) or level 0 (emergencies) will be sent to the specified target. The level can be specified as a keyword or number.
		
		• When you log to the console, monitor connection, or system buffer, the severity threshold specified by the *level* argument takes precedence over the ESM filtering. Even if the ESM filters return a message to be delivered to ESM targets, if the severity does not meet the configured threshold (is numerically higher than the level value), the message will not be delivered.
		
		• When you log to remote hosts, the stream tag allows you to specify a destination based on the type of message. The **stream** *stream-id* syntax allows you to configure the ESM to send only messages that have a specified stream value to a certain host.
		
		• The stream value is applied to messages by the configured syslog filter modules. For example, all Severity 5 messages could have a stream tag of

	Command or Action	Purpose
		"20" applied. You can then specify that all messages with a stream tag of "20" be sent to the host at 209.165.200.225.:
Step 6	Repeat Step 5 for each desired system logging destination.	• By issuing the logging host command multiple times, you can specify different targets for different system logging streams. • You can configure messages at different severity levels to be sent to the console, monitor connection, or system buffer. For example, you may want to display only important messages to the screen (using a monitor or console connection) at your network operations center (NOC).
Step 7	**logging source-interface** *type number* **Example:** `Device(config)# logging source-interface Gigabitethernet 0/0`	(Optional) Specifies the source interface for syslog messages sent to remote syslog hosts. • Normally, a syslog messages sent to remote hosts will use whatever interface is available at the time of the message generation. This command forces the device to send syslog messages to remote hosts only from the specified interface.
Step 8	**logging origin-id** {**hostname** | **ip** | **ipv6** |**string** *user-defined-id*} **Example:** `Device(config)# logging origin-id string "Domain 2, Router 5"`	(Optional) Allows you to add an origin identifier to syslog messages sent to remote hosts. • The origin identifier is added to the beginning of all syslog messages sent to remote hosts. The identifier can be the hostname, the IP address, or any text that you specify. • The origin identifier is useful for identifying the source of system logging messages in cases where you send syslog output from multiple devices to a single syslog host.
Step 9	**end** **Example:** `Device(config)# end`	Ends your current configuration session and returns the CLI to privileged EXEC mode.
Step 10	**show logging** **Example:** `Device# show logging`	(Optional) Displays the status of system logging, including the status of ESM filtered logging. • If filtered logging to the buffer is enabled, this command also shows the data stored in the buffer. • The order in which syslog filter modules are listed in the output of this command is the order in which the filter modules are executed.

Configuration Examples for the Embedded Syslog Manager

Example: Configuring the Embedded Syslog Manager Example

In the following example, the Embedded Syslog Manager (ESM) filter logging is enabled for the console connection, standard logging is enabled for the monitor connection and for the buffer, and XML-formatted logging is enabled for the host at 209.165.200.225:

```
Device(config)# logging filter tftp://209.165.200.225/ESM/escalate.tcl
Device(config)# logging filter slot0:/email.tcl user@example.com
Device(config)# logging filter slot0:/email_guts.tcl
Device(config)# logging console filtered
Device(config)# logging monitor 4
Device(config)# logging buffered debugging
Device(config)# logging host 209.165.200.225 xml
Device(config)# end

Device# show logging
Syslog logging: enabled (0 messages dropped, 8 messages rate-limited,
                0 flushes, 0 overruns, xml disabled, filtering enabled)
    Console logging: level debugging, 21 messages logged, xml disabled,
                     filtering enabled
    Monitor logging: level warnings , 0 messages logged, xml disabled,
                     filtering disabled
    Buffer logging: level debugging, 30 messages logged, xml disabled,
                     filtering disabled
    Logging Exception size (8192 bytes)
    Count and timestamp logging messages: disabled

Filter modules:
    tftp://209.165.200.225/ESM/escalate.tcl
    slot0:/email.tcl user@example.com

    Trap logging: level informational, 0 message lines logged
        Logging to 209.165.200.225, 0 message lines logged, xml enabled,
              filtering disabled

Log Buffer (8192 bytes):

*Jan 24 09:34:28.431: %SYS-5-CONFIG_I: Configured from console by console
*Jan 24 09:34:51.555: %SYS-5-CONFIG_I: Configured from console by console
*Jan 24 09:49:44.295: %SYS-5-CONFIG_I: Configured from console by console
Device#
```

Example: Syslog Filter Module

Syslog Script Modules are Tcl scripts. The following examples are provided to assist you in developing your own Syslog Script Modules.

Note These script modules are provided as examples only, and are not supported by Cisco. No guarantees, expressed or implied, are provided for the functionality or impact of these scripts.

Example: Severity Escalation

This ESM syslog filter module example watches for a single mnemonic (supplied via the first CLI argument) and escalates the severity of the message to that specified by the second CLI argument.

```
#  ===================================================================
#  Embedded Syslog Manager                              ||        ||
#                                                       ||        ||
#  Severity Escalation Filter                          ||||      ||||
#                                                  ..:||||||:..:||||||:..
#                                                 -----------------------
#                                                 C i s c o  S y s t e m s
#  ===================================================================
#
# Usage: Set CLI Args to "mnemonic new_severity"
#
# Namespace: global
# Check for null message
if { [string length $::orig_msg] == 0} {
   return ""
}

if { [info exists ::cli_args] } {
    set args [split $::cli_args]
    if { [ string compare -nocase [lindex $args 0] $::mnemonic ] == 0 } {
        set ::severity [lindex $args 1]
        set sev_index [ string first [lindex $args 0] $::orig_msg ]
        if {  $sev_index >= 2 } {
           incr sev_index -2
           return [string replace $::orig_msg $sev_index $sev_index \
              [lindex $args 1]]
        }
    }
}
return $::orig_msg
```

Example: Message Counting

This ESM syslog filter module example is divided into two files for readability. The first file allows the user to configure those messages that they want to count and how often to summarize (correlation window) by populating the msg_to_watch array. The actual procedures are in the counting_guts.tcl file. Note the use of the separate namespace “counting” to avoid conflict with other ESM filters that may also perform background processing.

```
#  ===================================================================
#  Embedded Syslog Manager                              ||        ||
#                                                       ||        ||
#  Message Counting Filter                             ||||      ||||
#                                                  ..:||||||:..:||||||:..
#                                                 -----------------------
#                                                 C i s c o  S y s t e m s
# ================================================================================


#
# Usage:
# 1) Define the location for the counting_guts.tcl script
#
# 2) Define message categories to count and how often to dump them (sec)
#    by populating the "msg_to_watch" array below.
#    Here we define category as facility-severity-mnemonic
#    Change dump time to 0 to disable counting for that category
#
# Namespace: counting
namespace eval ::counting {
```

```
    set sub_script_url tftp://172.16.0.0/12/ESM/counting_guts.tcl
    array set msg_to_watch {
        SYS-5-CONFIG_I            5
    }
#  ======================= End User Setup ==============================
# Initialize processes for counting
    if { [info exists init] == 0 } {
       source $sub_script_url
       set position $module_position
    }
# Process the message
process_category
} ;# end namespace counting
```

Message Counting Support Module (counting_guts.tcl)

```
#  ====================================================================
#  Embedded Syslog Manager                              ||        ||
#                                                       ||        ||
#  Message Counting Support Module                     ||||      ||||
#                                                   ..:||||||:..:||||||:..
#  (No User Modification)                          ------------------------
#                                                  C i s c o  S y s t e m s
# ====================================================================================


namespace eval ::counting {

# namespace variables

array set cat_msg_sev {}
array set cat_msg_traceback {}
array set cat_msg_pid {}
array set cat_msg_proc {}
array set cat_msg_ts {}
array set cat_msg_buginfseq {}
array set cat_msg_name {}
array set cat_msg_fac {}
array set cat_msg_format {}
array set cat_msg_args {}
array set cat_msg_count {}
array set cat_msg_dump_ts {}

# Should I count this message ?
   proc query_category {cat} {
        variable msg_to_watch
        if { [info exists msg_to_watch($cat)] } {
            return $msg_to_watch($cat)
        } else {
            return 0
        }
    }
   proc clear_category {index} {
        variable cat_msg_sev
        variable cat_msg_traceback
        variable cat_msg_pid
        variable cat_msg_proc
        variable cat_msg_ts
        variable cat_msg_buginfseq
        variable cat_msg_name
        variable cat_msg_fac
        variable cat_msg_format
        variable cat_msg_args
        variable cat_msg_count
        variable cat_msg_dump_ts
        unset cat_msg_sev($index) cat_msg_traceback($index) cat_msg_pid($index)\
              cat_msg_proc($index) cat_msg_ts($index) \
              cat_msg_buginfseq($index)  cat_msg_name($index) \
              cat_msg_fac($index) cat_msg_format($index) cat_msg_args($index)\
```

```
            cat_msg_count($index) cat_msg_dump_ts($index)
    }
# send out the counted messages
   proc dump_category {category} {
        variable cat_msg_sev
        variable cat_msg_traceback
        variable cat_msg_pid
        variable cat_msg_proc
        variable cat_msg_ts
        variable cat_msg_buginfseq
        variable cat_msg_name
        variable cat_msg_fac
        variable cat_msg_format
        variable cat_msg_args
        variable cat_msg_count
        variable cat_msg_dump_ts
        variable poll_interval
        set dump_timestamp [cisco_service_timestamp]
foreach index [array names cat_msg_count $category] {
      set fsm "$cat_msg_fac($index)-$cat_msg_sev($index)-$cat_msg_name($index)"
      set ::orig_msg \
        [format "%s%s: %%%s: %s %s %s %s - (%d occurence(s) between %s and %s)"\
        $cat_msg_buginfseq($index)\
          $dump_timestamp\
          $fsm \
          [uplevel 1 [linsert $cat_msg_args($index) 0 ::format
$cat_msg_format($index) ]] \
          $cat_msg_pid($index) \
          $cat_msg_proc($index) \
          $cat_msg_traceback($index) \
          $cat_msg_count($index) \
          $cat_msg_ts($index) \
          $dump_timestamp]
# Prepare for remaining ESM filters
                set ::severity $cat_msg_sev($index)
                set ::traceback $cat_msg_traceback($index)
                set ::pid $cat_msg_pid($index)
                set ::process $cat_msg_proc($index)
                set ::timestamp $cat_msg_ts($index)
                set ::buginfseq $cat_msg_buginfseq($index)
                set ::mnemonic $cat_msg_name($index)
                set ::facility $cat_msg_fac($index)
                set ::format_string $cat_msg_format($index)
                set ::msg_args [split $cat_msg_args($index)]
                esm_errmsg $counting::position
                clear_category $index
        }
  }
# See if this message already has come through since the last dump.
# If so, increment the count, otherwise store it.
   proc process_category {} {
        variable cat_msg_sev
        variable cat_msg_traceback
        variable cat_msg_pid
        variable cat_msg_proc
        variable cat_msg_ts
        variable cat_msg_buginfseq
        variable cat_msg_name
        variable cat_msg_fac
        variable cat_msg_format
        variable cat_msg_args
        variable cat_msg_count
        variable cat_msg_dump_ts
        if { [string length $::orig_msg] == 0} {
           return ""
        }
        set category "$::facility-$::severity-$::mnemonic"
        set correlation_window [expr [ query_category $category ] * 1000]
        if { $correlation_window == 0 } {
            return $::orig_msg
        }
        set message_args [join $::msg_args]
        set index "$category,[lindex $::msg_args 0]"
```

```
        if { [info exists cat_msg_count($index)] } {
                incr cat_msg_count($index)
         } else {
              set cat_msg_sev($index) $::severity
              set cat_msg_traceback($index) $::traceback
              set cat_msg_pid($index) $::pid
              set cat_msg_proc($index) $::process
              set cat_msg_ts($index) $::timestamp
              set cat_msg_buginfseq($index) $::buginfseq
              set cat_msg_name($index) $::mnemonic
              set cat_msg_fac($index) $::facility
              set cat_msg_format($index) $::format_string
              set cat_msg_args($index) $message_args
              set cat_msg_count($index) 1
              set cat_msg_dump_ts($index) [clock seconds]
              catch [after $correlation_window counting::dump_category $index]
         }
         return ""
   }
# Initialized
set init 1
} ;#end namespace counting
```

Example: XML Tagging

This ESM syslog filter module applies user-defined XML tags to syslog messages:

```
#  ===================================================================
#  Embedded Syslog Manager                            ||        ||
#                                                     ||        ||
#  XML Tagging Filter                                ||||      ||||
#                                                 ..:||||||:..:||||||:..
#                                                ------------------------
#                                               C i s c o  S y s t e m s
#  ===================================================================
#
# Usage: Define desired tags below.
#
# Namespace: xml
# Check for null message
    if { [string length $::orig_msg] == 0} {
       return ""
    }
namespace eval xml {
####  define tags ####
set MSG_OPEN "<ios-log-msg>"
set MSG_CLOSE "</ios-log-msg>"
set FAC_OPEN   "<facility>"
set FAC_CLOSE  "</facility>"
set SEV_OPEN   "<severity>"
set SEV_CLOSE  "</severity>"
set MNE_OPEN   "<msg-id>"
set MNE_CLOSE  "</msg-id>"
set SEQ_OPEN   "<seq>"
set SEQ_CLOSE  "</seq>"
set TIME_OPEN  "<time>"
set TIME_CLOSE "</time>"
set ARGS_OPEN  "<args>"
set ARGS_CLOSE "</args>"
set ARG_ID_OPEN "<arg id="
set ARG_ID_CLOSE "</arg>"
set PROC_OPEN "<proc>"
set PROC_CLOSE "</proc>"
set PID_OPEN "<pid>"
set PID_CLOSE "</pid>"
set TRACE_OPEN "<trace>"
set TRACE_CLOSE "</trace>"
#  ====================== End User Setup ==============================
#### clear result ####
set result ""
```

```
#### message opening, facility, severity, and name ####
append result $MSG_OPEN $FAC_OPEN $::facility $FAC_CLOSE $SEV_OPEN $::severity
$SEV_CLOSE $MNE_OPEN $::mnemonic $MNE_CLOSE
#### buginf sequence numbers ####
if { [string length $::buginfseq ] > 0 } {
    append result $SEQ_OPEN $::buginfseq $SEQ_CLOSE
}
#### timestamps ####
if { [string length $::timestamp ] > 0 } {
    append result $TIME_OPEN $::timestamp $TIME_CLOSE
}
#### message args ####
if { [info exists ::msg_args] } {
    if { [llength ::msg_args] > 0 } {
        set i 0
        append result $ARGS_OPEN
        foreach arg $::msg_args {
            append result $ARG_ID_OPEN $i ">" $arg $ARG_ID_CLOSE
                incr i
        }
        append result $ARGS_CLOSE
    }
}
#### traceback ####
if { [string length $::traceback ] > 0 } {
    append result $TRACE_OPEN $::traceback $TRACE_CLOSE
}
#### process ####
if { [string length $::process ] > 0 } {
    append result $PROC_OPEN $::process $PROC_CLOSE
}
#### pid ####
if { [string length $::pid ] > 0 } {
    append result $PID_OPEN $::pid $PID_CLOSE
}
#### message close ####
append result $MSG_CLOSE
return "$result"
} ;# end namespace xml
```

Example: SMTP-Based E-Mail Alert

This ESM syslog filter module example watches for configuration messages and sends them to the e-mail address supplied as a CLI argument. This filter is divided into two files. The first file implements the filter, and the second file implements the Simple Mail Transfer Protocol (SMTP) client.

```
#  ====================================================================
#  Embedded Syslog Manager                              ||        ||
#                                                       ||        ||
#  Email Filter                                        ||||      ||||
#  (Configuration Change Warning)                   ..:||||||:..:||||||:..
#                                                  ------------------------
#                                                   C i s c o  S y s t e m s
#   ====================================================================
# Usage:  Provide email address as CLI argument.  Set email server IP in
#         email_guts.tcl
#
# Namespace: email
if { [info exists email::init] == 0 } {
   source tftp://123.123.123.123/ESM/email_guts.tcl
}
# Check for null message
if { [string length $::orig_msg] == 0} {
      return ""
   }
if { [info exists ::msg_args] } {
    if { [string compare -nocase CONFIG_I $::mnemonic ] == 0 } {
                email::sendmessage $::cli_args $::mnemonic \
                [string trim $::orig_msg]
```

```
    }
}
return $::orig_msg
```

E-Mail Support Module (email_guts.tcl)

```
#  ====================================================================
#  Embedded Syslog Manager                              ||        ||
#                                                       ||        ||
#  Email Support Module                                ||||      ||||
#                                                   ..:||||||:..:||||||:..
#                                                  ------------------------
#                                                   C i s c o  S y s t e m s
#   ====================================================================
#
# Usage: Set email host IP, from, and friendly strings below.
#
namespace eval email {
    set sendmail(smtphost)172.16.0.1
    set sendmail(from) $::hostname
    set sendmail(friendly) $::hostname
    proc sendmessage {toList subject body} {
        variable sendmail
        set smtphost $sendmail(smtphost)
        set from $sendmail(from)
        set friendly $sendmail(friendly)
        set sockid [socket $smtphost 25]
## DEBUG
set status [catch {
        puts $sockid "HELO $smtphost"
        flush $sockid
        set result [gets $sockid]
        puts $sockid "MAIL From:<$from>"
        flush $sockid
        set result [gets $sockid]
        foreach to $toList {
            puts $sockid "RCPT To:<$to>"
            flush $sockid
        }
        set result [gets $sockid]
        puts  $sockid "DATA "
        flush $sockid
        set result [gets  $sockid]
        puts  $sockid "From: $friendly <$from>"
        foreach to $toList {
            puts $sockid "To:<$to>"
        }
        puts  $sockid "Subject: $subject"
        puts  $sockid "\n"
        foreach line [split $body "\n"] {
            puts  $sockid " $line"
        }
        puts  $sockid "."
        puts  $sockid "QUIT"
        flush $sockid
        set result [gets  $sockid]
} result]
        catch {close $sockid }
    if {$status} then {
        return -code error $result
    }
}
} ;# end namespace email
set email::init 1
```

Example: Stream

This ESM syslog filter module example watches for a given facility (first CLI argument) and routes these messages to a given stream (second CLI argument:.

```
#  ===================================================================
#  Embedded Syslog Manager                              ||        ||
#                                                       ||        ||
#  Stream Filter (Facility)                            ||||      ||||
#                                                   ..:||||||:..:||||||:..
#                                                  -----------------------
#                                                   C i s c o  S y s t e m s
#   ===================================================================
# Usage:  Provide facility and stream as CLI arguments.
#
# Namespace: global
# Check for null message
#  ====================== End User Setup ==============================
set args [split $::cli_args]
if { [info exists ::msg_args] } {
    if { $::facility == [lindex $args 0] } {
                set ::stream [lindex $args 1]
    }
}
return $::orig_msg}
```

Example: Source IP Tagging

The **logging source-interface** CLI command can be used to specify a source IP address in all syslog packets sent from the device. The following syslog filter module example demonstrates the use of **show** CLI commands (**show running-config** and **show ip interface** in this case) within a filter module to add the source IP address to syslog messages. The script looks for the local namespace variable “source_ip::init” first. If the variable is not defined in the first syslog message processed, the filter will run the **show** commands and use regular expressions to get the source interface and then its IP address.

Note that in this script, the **show** commands are run only once. If the source interface or its IP address were to be changed, the filter would have to be reinitialized to pick up the new information. (You could have the show commands run on every syslog message, but this would not scale well.)

```
#  ===================================================================
#  Embedded Syslog Manager                              ||        ||
#                                                       ||        ||
#  Source IP Module                                    ||||      ||||
#                                                   ..:||||||:..:||||||:..
#                                                  -----------------------
#                                                   C i s c o  S y s t e m s
#   ===================================================================
# Usage: Adds Logging Source Interface IP address to all messages.
#
# Namespace:source_ip
#
#  ====================== End User Setup ==============================
namespace eval ::source_ip {
    if { [info exists init] == 0 } {
       if { [catch {regexp {^logging source-interface (.*$)} [exec show
run | inc logging source-interface] match source_int}]} {
            set suffix "No source interface specified"
       } elseif { [catch {regexp {Internet address is (.*)/.*$} [exec
show ip int $source_int | inc Internet] match ip_addr}]} {
            set suffix "No IP address configured for source interface"
       } else {
           set suffix $ip_addr
       }
```

```
        set init 1
    }

    if { [string length $::orig_msg] == 0} {
      return ""
    }
    return "$::orig_msg - $suffix"
} ;# end namespace source_ip
```

Additional References for the Embedded Syslog Manager

Related Documents

Related Topic	Document Title
Cisco IOS Commands	Cisco IOS Master Commands List, All Releases
Cisco IOS XE Commands	Command Lookup Tool
System Message Logging	Troubleshooting and Fault Management module
XML Formatted System Message Logging	XML Interface to Syslog Messages module
Tcl 8.3.4 Support in Cisco Software	*Cisco IOS Scripting with Tcl* module
Network Management commands (including logging commands): complete command syntax, defaults, command mode, command history, usage guidelines, and examples	*Cisco IOS Network Management Command Reference*

Standards and RFCs

Standard/RFC	Title
No new or modified standards are supported, and support for existing standards has not been modified.	--
RFC-3164	*The BSD Syslog Protocol* This RFC is informational only. The Cisco implementation of syslog does not claim full compliance with the protocol guidelines mentioned in this RFC. Not all supported RFCs are listed.

MIBs

MIB	MIBs Link
No new or modified standards are supported, and support for existing standards has not been modified.	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for the Embedded Syslog Manager

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 1: Feature Information for the Embedded Syslog Manager

Feature Name	Releases	Feature Information
Embedded Syslog Manager		The Embedded Syslog Manager (ESM) feature provides a programmable framework that allows you to filter, escalate, correlate, route, and customize system logging messages prior to delivery by the Cisco IOS system message logger.

Glossary

Note Refer to the "Internetworking Terms and Acronyms" section for terms not included in this glossary.

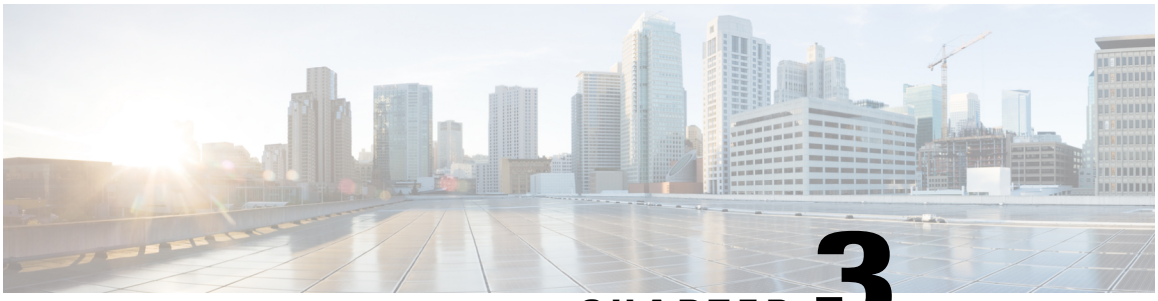
console--Specifies the connection (CTY or console line) to the console port of the device. Typically, this is a terminal attached directly to the console port, or a PC with a terminal emulation program. Corresponds to the **show terminal** command.

monitor--Specifies the TTY (TeleTYpe terminal) line connection at a line port. In other words, the “monitor” keyword corresponds to a terminal line connection or a Telnet (terminal emulation) connection. TTY lines (also called ports) communicate with peripheral devices such as terminals, modems, and serial printers. An example of a TTY connection is a PC with a terminal emulation program connected to the device using a dialup modem.

SEMs--Abbreviation for system error messages. “System error messages” is the term formerly used for messages generated by the system logging (syslog) process. Syslog messages use a standardized format, and come in eight severity levels, from “emergencies” (level 0) to “debugging” (level 7). The term “system error message” is actually misleading, because these messages can include notifications of device activity beyond “errors” (such as informational notices).

syslog--Abbreviation for the system message logging process in Cisco software. Also used to identify the messages generated, as in “syslog messages.” Technically, the term “syslog” refers only to the process of logging messages to a remote host or hosts, but is commonly used to refer to all Cisco system logging processes.

trap--A trigger in the system software for sending error messages. “Trap logging” means logging messages to a remote host. The remote host is actually a syslog host from the perspective of the device sending the trap messages, but because the receiving device typically provides collected syslog data to other devices, the receiving device is also referred to as a “syslog server.”



CHAPTER 3

Logging to Local Nonvolatile Storage

The Logging to Local Nonvolatile Storage feature enables system logging messages to be saved on an advanced technology attachment flash disk. Messages saved on bootflash or a harddisk persist after a device is rebooted.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table at the end of this module.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites for Logging to Local Nonvolatile Storage

The logging buffered Command Must Be Enabled

Before the Logging to Local Nonvolatile Storage feature can be enabled with the **logging persistent** command, you must enable the logging of messages to an internal buffer with the **logging buffered** command. For additional information, see the "Writing Logging Messages to Bootflash or a Harddisk" section.

Restrictions for Logging to Local Nonvolatile Storage

Available Bootflash or Harddisk Space Constrains the Size and Number of Stored Log Files

The amount of bootflash or harddisk space allocated to system logging messages constrains the number of logging files that can be stored. When the allocation threshold is passed, the oldest log file in the directory is deleted to make room for new system logging messages. To permanently store system logging messages, you must archive them to an external device. For more information, see "Copying Logging Messages to an External Disk" section.

Note Logging to local nonvolatile storage can use up to 2 GB of storage space.

Information About Logging to Local Nonvolatile Storage

System Logging Messages

System logging messages include error and debug messages generated by application programming interfaces (APIs) on the device. Typically, logging messages are stored in a device's memory buffer; when the buffer is full, older messages are overwritten by new messages. All logging messages are erased from the memory buffer when the device reboots.

How to Configure Logging to Local Nonvolatile Storage

Writing Logging Messages to Bootflash or a Harddisk

Perform this task to enable the Logging to Local Nonvolatile Storage feature and write logging messages to bootflash or a harddisk.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **logging buffered** [*buffer-size* | *severity-level*]
4. **logging persistent** [**url harddisk:**/*directory*] [**size** *filesystem-size*] [**filesize** *logging-file-size*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enables global configuration mode.
Step 3	**logging buffered** [*buffer-size* | *severity-level*] **Example:** `Device(config)# logging buffered`	Enables system message logging to a local buffer and limits messages logged to the buffer based on severity. • The optional *buffer-size* argument specifies the size of the buffer. Range is from 4096 to 4294967295. The default size varies by platform. • The optional *severity-level* argument limits the logging of messages to the buffer to those no less severe than the specified level.
Step 4	**logging persistent** [**url harddisk:**/*directory*] [**size** *filesystem-size*] [**filesize** *logging-file-size*] **Example:** `Device(config)# logging persistent url harddisk:/syslog size 134217728 filesize 16384` **Note** The default value is: url: bootflash:/syslog filesystem-size: 10% of total disk space logging-file-size: 262144	Writes logging messages from the memory buffer to the specified directory on the device's bootflash or a harddisk. • Before logging messages are written to a file on the bootflash or harddisk, the Cisco software checks to see if there is sufficient disk space. If not, the oldest file of logging messages (by timestamp) is deleted, and the current file is saved. • The filename format of log files is log_MM:DD:YYYY::hh:mm:ss. For example: log_11:26:2012::01:01:41. **Note** This feature supports only one log file per second due to its filename format, which contains a timestamp suffix down to the seconds level. **Note** The defaults for this command are as follows: • **url: bootflash:/syslog** Filesystem-size: 10% of total disk space. Logging-file-size: 262144

Copying Logging Messages to an External Disk

Perform this task to copy logging messages from the bootflash or a harddisk to an external disk.

SUMMARY STEPS

1. **enable**
2. **copy** *source-url destination-url*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**copy** *source-url destination-url* **Example:** `Device# copy harddisk:/syslog ftp://myuser/mypass@192.168.1.129/syslog`	Copies the specified file or directory on the bootflash or a harddisk via FTP to the specified URL.

Configuration Examples for Logging to Local Nonvolatile Storage

Example: Writing Logging Messages to Bootflash or a Harddisk

The following example shows how to write up to 134217728 bytes (128 MB) of logging messages to the syslog directory of disk 0, specifying a file size of 16384 bytes:

```
Device(config)# logging buffered
Device(config)# logging persistent url harddisk:/syslog size 134217728 filesize 16384
```

Example: Copying Logging Messages to an External Disk

The following example shows how to copy logging messages from the device's bootflash or harddisk to an external disk:

```
Device# copy harddisk:/syslog ftp://myuser/mypass@192.168.1.129/syslog
```

Additional References

Related Documents

Related Topic	Document Title
copy command	Cisco IOS Configuration Fundamentals Command Reference
Network management commands (including logging commands): complete command syntax, defaults, command mode, command history, usage guidelines, and examples	Cisco IOS Network Management Command Reference

MIBs

MIBs	MIBs Link
• No new or modified MIBs are supported by this feature, and support for existing MIBs has not been modified by this feature.	To locate and download MIBs for selected platforms, Cisco IOS XE releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

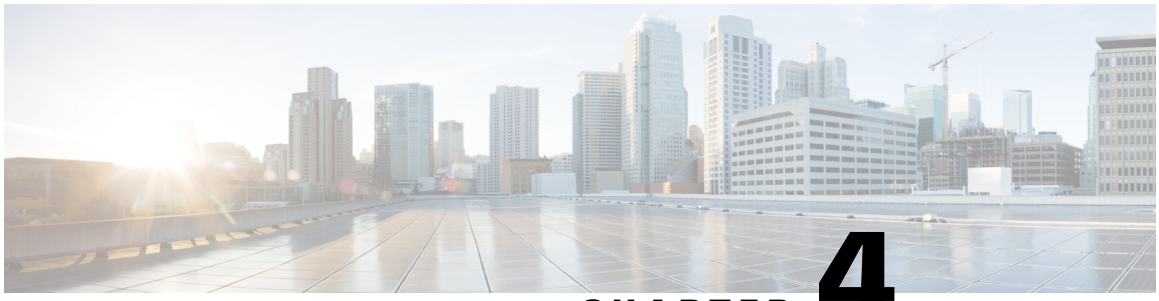
Feature Information for Logging to Local Nonvolatile Storage

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 2: Feature Information for Logging to Local Nonvolatile Storage

Feature Name	Releases	Feature Information
Logging to Local Nonvolatile Storage	Cisco IOS XE Release 2.1	The Logging to Local Nonvolatile Storage feature enables system logging messages to be saved on an advanced technology attachment flash disk. Messages saved on bootflash or a harddisk persist after a device is rebooted. The following command was introduced or modified: **logging persistent**.



CHAPTER 4

Reliable Delivery and Filtering for Syslog

The Reliable Delivery and Filtering for Syslog feature allows a device to be customized for receipt of syslog messages. This feature provides reliable and secure delivery for syslog messages using Blocks Extensible Exchange Protocol (BEEP). Additionally, it allows multiple sessions to a single logging host, independent of the underlying transport method, and provides a filtering mechanism called a message discriminator.

This module describes the functions of the Reliable Delivery and Filtering for Syslog feature and how to configure them in a network.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table at the end of this module.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites for Reliable Delivery and Filtering for Syslog

- The device level rate limit is set to meet business needs, network traffic requirements, or performance requirements.
- Each BEEP session must have an RFC 3195-compliant syslog-RAW exchange profile.
- A Simple Authentication and Security Layer (SASL) profile specifying “DIGEST-MD5” for provisioning services must be established when a crypto image is used.
- Syslog servers must be compatible with BEEP.
- Syslog server applications must be capable of handling multiple sessions to use the multiple session capability of the Reliable Delivery and Filtering for Syslog feature.

Restrictions for Reliable Delivery and Filtering for Syslog

- Only the syslog-RAW, SASL, and Transport Layer Security (TLS) profiles are supported.
- Both ends of a syslog session must use the same transport method.
- A message discriminator must be defined before it can be associated with a specific syslog session.
- A syslog session can be associated with only one message discriminator.
- Message delivery with User Datagram Protocol (UDP) will be faster than with either TCP or BEEP.

Information About Reliable Delivery and Filtering for Syslog

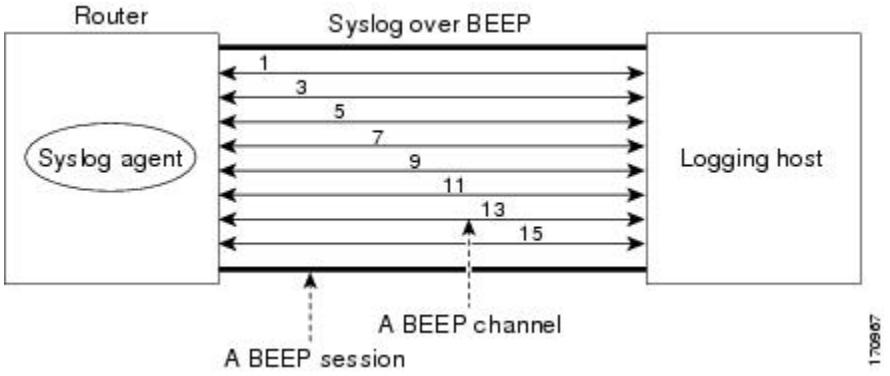
BEEP Transport Support

BEEP is a generic application protocol framework for connection-oriented, asynchronous interactions. It is intended to provide the features that traditionally have been duplicated in various protocol implementations. BEEP typically runs on top of TCP and allows the exchange of messages. Unlike HTTP and similar protocols, either end of the connection can send a message at any time. BEEP also includes facilities for encryption and authentication and is highly extensible.

BEEP as a transport protocol for syslog messages provides multiple channels. Each channel can be configured for a separate session to the same host. BEEP provides reliable transport. Syslog messages sent over a BEEP connection are guaranteed to be delivered in sequence.

With command-line interface (CLI) commands introduced in the Reliable Delivery and Filtering for Syslog feature, you can configure a new BEEP session to have a maximum of eight channels.

The figure below shows a BEEP session with eight channels, allowing eight separate syslog sessions.



Channels are identified as 1, 3, 5, 7, 9, 11, 13, and 15. The number of available channels (eight) was designed to correspond to the number of severity levels of classic RFC-3164 syslog messages (0 to 7). Message discriminators can be used such that severity levels are mapped to BEEP channels. An intelligent BEEP syslog server (depending upon the BEEP stack used) could use this mapping to prioritize messages with higher severity (see RFC 3081, section 3.1.4). Unless associated with a message discriminator, all syslog sessions (channels) receive all syslog messages.

Syslog Message

A syslog message has a sequence number that allows the host to use the number as an identifier for the message as well as to detect whether there were any gaps in the messages that were received. Syslog messages are numbered consecutively. The reliability of BEEP does not replace the need for sequence numbers, which are required for the following reasons:

- A sequence number provides an easy way to identify a syslog message. Independent of reliability considerations, the sequence number serves as a message identifier.
- A BEEP session may not be in place for the entire time that a device sending syslog messages is up. Sequence numbers provide a way for management applications to assess whether messages were missed between BEEP sessions.
- BEEP is only one of several transports. Unreliable transports are also used and the syslog protocol should not rely on a reliable transport always being provided.

The existing numbering scheme for syslog messages is limited with the extension of syslog to accommodate advanced message discrimination features and multiple hosts. Message discrimination leads to gaps in the sequence numbers, meaning that hosts lose the ability to detect whether they have missed a message. If syslog messages are numbered consecutively on each session to avoid the gaps in sequence numbers, it will not be possible to easily correlate which messages are the same and which ones are different because the sequence number would no longer uniquely identify a message.

To separate identification from sequencing and reliability, the following changes to syslog messages were made:

- The sequence number is retained as an identifier for the message. Messages with a lower number precede messages with a higher number, but they are not guaranteed to be consecutive.
- An additional field is added in the body portion of a syslog message to help ensure sequencing. The contents of this field contain a sequence number for a particular session. The same message transmitted over different sessions may have a different sequence number.

Syslog Session

A syslog session is a logical link from the syslog agent on a device to the recipient of a syslog message. For example, a syslog session can be established between a syslog agent and any of the following:

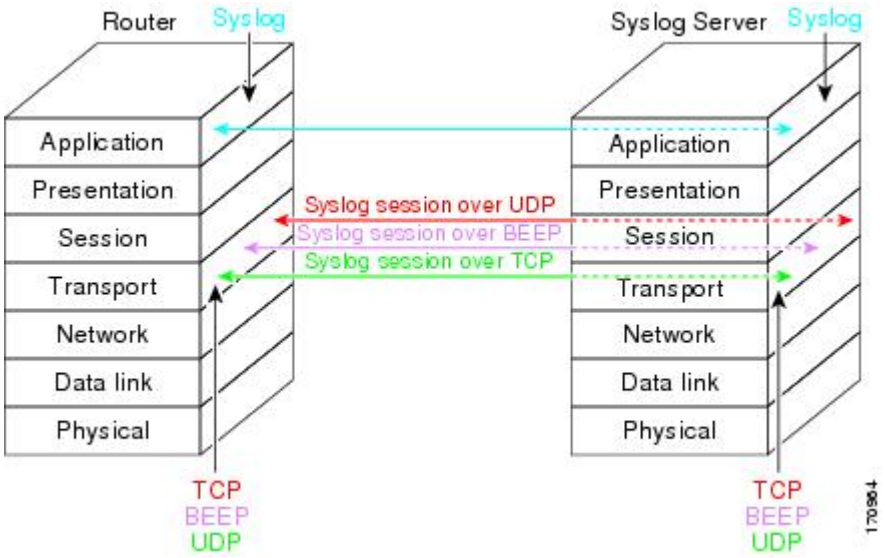
- Device console
- Device logging buffer
- Device monitor
- External syslog server

A syslog session runs over a transport connection between the syslog source and the syslog destination. A transport connection can use any of the following protocols:

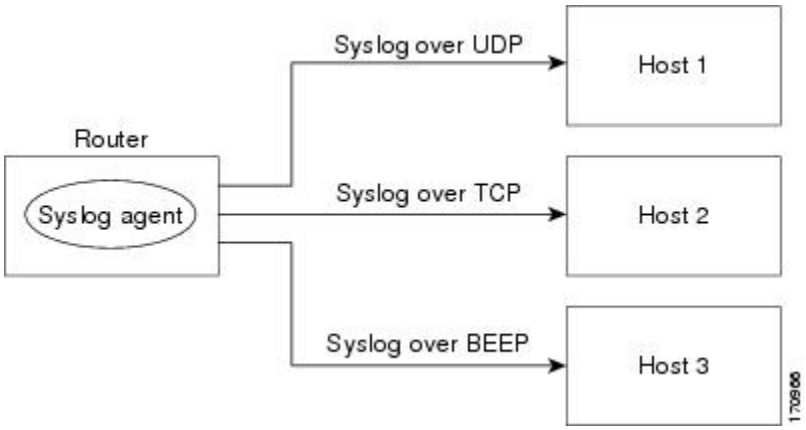
- TCP
- UDP (association to one remote address and port)
- BEEP (channel within a BEEP session)

The figure below shows a mapping of syslog sessions and transport protocols between a device and a syslog server using an Open Systems Interconnection (OSI) model.

Note The figure below is best viewed using Internet Explorer.



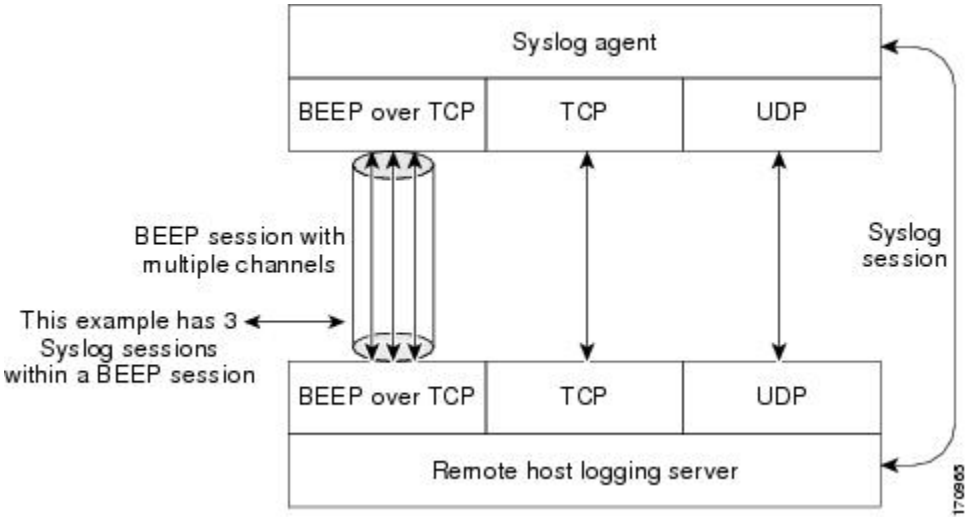
The figure below shows multiple syslog sessions from a single syslog agent to different hosts using UDP, TCP and BEEP.



Multiple Syslog Sessions

A syslog session is independent of a transport connection. A Cisco device can support multiple syslog sessions, each running over its own transport connection. Multiple syslog sessions cannot share the same transport connection, but multiple syslog sessions may terminate at the same remote host, each running over its own transport connection. An example is a BEEP session in which multiple channels are used.

The figure below shows an end-to-end view of a syslog session. Note the three syslog sessions within a single BEEP session.



The TCP and UDP protocols do not have multiplexed channels but the protocols do allow for using multiple ports to establish multiple syslog sessions to the same syslog host. To enable the UDP and TCP transport methods to have capability similar to BEEP's multiple channel capability, the Reliable Delivery and Filtering for Syslog feature allows multiple syslog sessions to be established via the UDP and TCP transport methods to the same logging host. Multiple syslog sessions going over BEEP sessions is also supported.

Message Discriminator

A message discriminator is a syslog processor. A message discriminator is associated with a syslog session and binds that session to a transport connection.

Prior to message delivery, the message is subject to the message discriminator with a user-specified list of criteria. After the first filtering criterion results in a message being blocked, the filtering check stops.

Note The sequence of criteria in the CLI does not affect the sequence in which criteria is checked.

- Following are filtering criteria. These criteria are checked in the order listed here:
  - Severity level or levels specified
  - Facility within the message body that matches a regular expression
  - Mnemonic that matches a regular expression
  - Part of the body of a message that matches a regular expression

A message discriminator offers the following capabilities:

- Optional rate limiting--Specifying a transmission rate of messages per time interval that is not to be exceeded. If the rate limit is exceeded, messages are either delayed or dropped, at the discretion of the device. The application of a rate limiter means that reliable delivery of syslog messages over that syslog session is no longer guaranteed. The purpose of a rate limiter is to avoid potential “flooding” at recipient syslog servers for applications that do not require guaranteed syslog delivery.
- Correlating--Inspecting candidate event messages and possibly aggregating information across events, creating a new event that contains the aggregated information. Correlating functions include:
  - Elimination of duplicate messages by maintaining a message count and waiting a specific time period between sending the first message of a certain type and sending the next message of that type
  - Elimination of oscillating messages
  - Simple message correlation; for example, if one message is a symptom of a cause reported by another message, one consolidated message is reported

A message discriminator can be associated with a specific destination and transport; that is, the filter can be host dependent. For this reason, a message discriminator is attached to a syslog session, transport, or channel, with possible device support for multiple sessions, transports, or channels, each of which can be attached to a different discriminator.

The establishment of a message discriminator should be separate from the establishment of a syslog session. A message discriminator should refer to the syslog session, transport, or channel to which it should be attached. The reasons for the separation are the following:

- Message discriminators can be managed separately from the connections, and refinements in the capabilities available to set up message discriminators need not affect how syslog sessions are set up and vice versa.

- Multiple connections can be attached to the same message discriminator, allowing for various syslog redundancy topologies.

When an explicit message discriminator is not associated with a syslog session, the generic message discriminator from the device-wide global settings is used. You can create an “empty” message discriminator without specifying attribute values (no rate limit and no filter configured).

Rate Limiting

The device-wide rate limiting capability in Cisco IOS XE syslog is preserved in the Reliable Delivery and Filtering for Syslog feature and is referred to as “global rate limiting.” If you do not use global rate limiting, all event messages are sent to remote syslog hosts if system resources can support the volume. When global rate limiting is set, it applies to all destinations. The value is set to the rate-limit attribute of the “generic message discriminator” if one has been set. The disadvantage of global rate limiting is that the rate limit of the least performing remote syslog host sets the rate for how fast a device can send out syslog messages.

The Reliable Delivery and Filtering for Syslog feature provides syslog session-based rate limiting to bypass the effects of global rate limiting. This session-based rate limiting is associated with a specific message discriminator and allows you to set the rate acceptance level independently for each syslog session.

Use of global rate limiting is not recommended when session-based rate limiting is in effect. A rate limit in a message discriminator specifies a not-to-exceed rate of syslog messages but does not guarantee that this rate will be reached. A configured global rate limit may cause messages on a session to be dropped even if the rate limit for that session has not been reached. These actions are important to understand if global rate limiting and session-based rate limiting are used concurrently.

Benefits of Reliable Delivery and Filtering for Syslog

- Authentication and encryption capabilities in BEEP provide reliable and secure delivery for syslog messages
- Multiple sessions to a single logging host independent of the underlying transport method
- Session-based message filtering and rate limiting
- Multiple connections can be attached to the same message discriminator, allowing various syslog redundancy topologies
- New CLI command to disable the default syslog count
- New CLI command to help identify relative positions of syslog messages that are dropped due to rate limiting

How to Configure Reliable Delivery and Filtering for Syslog

Creating a Message Discriminator

Perform this task to create a message discriminator for syslog messages.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **logging discriminator** *discr-name* [[**facility**] [**mnemonics**] [**msg-body**] {**drops** *string*| **includes** *string*}] [**severity** {**drops** *sev-num* | **includes** *sev-num*}] [**rate-limit** *msglimit*]
4. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**logging discriminator** *discr-name* [[**facility**] [**mnemonics**] [**msg-body**] {**drops** *string*| **includes** *string*}] [**severity** {**drops** *sev-num*	**includes** *sev-num*}] [**rate-limit** *msglimit*] **Example:** `Device(config)# logging discriminator pacfltr1 facility includes facl357`	Creates a message discriminator with a facility subfilter. In this example, all messages with “facl357” in the facility field will be delivered.
Step 4	**end** **Example:** `Device(config)# end`	Returns the CLI to privileged EXEC mode.

Associating a Message Discriminator with a Logging Buffer

Perform this task to associate a message discriminator with a specific buffer.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **logging discriminator** *discr-name* [[**facility**] [**mnemonics**] [**msg-body**] {**drops** *string*| **includes** *string*}] [**severity** {**drops** *sev-num* | **includes** *sev-num*}] [**rate-limit** *msglimit*]
4. **logging buffered** [**discriminator** *discr-name* | **xml**] [*buffer-size*] [*severity-level*]
5. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**logging discriminator** *discr-name* [[**facility**] [**mnemonics**] [**msg-body**] {**drops** *string*| **includes** *string*}] [**severity** {**drops** *sev-num*	**includes** *sev-num*}] [**rate-limit** *msglimit*] **Example:** `Device(config)# logging discriminator pacfltr2`	Creates a message discriminator.
Step 4	**logging buffered** [**discriminator** *discr-name*	**xml**] [*buffer-size*] [*severity-level*] **Example:** `Device(config)# logging buffered discriminator pacfltr2 5`	Enables logging to a local buffer and specifies a message discriminator.
Step 5	**end** **Example:** `Device(config)# end`	Returns the CLI to privileged EXEC mode.

Associating a Message Discriminator with a Console Terminal

Perform this task to associate a message discriminator with a console terminal.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **logging discriminator** *discr-name* [[**facility**] [**mnemonics**] [**msg-body**] {**drops** *string*| **includes** *string*}] [**severity** {**drops** *sev-num* | **includes** *sev-num*}] [**rate-limit** *msglimit*]
4. **logging console** [**discriminator** *discr-name* | **xml**] [*severity-level*]
5. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**logging discriminator** *discr-name* [[**facility**] [**mnemonics**] [**msg-body**] {**drops** *string*| **includes** *string*}] [**severity** {**drops** *sev-num*	**includes** *sev-num*}] [**rate-limit** *msglimit*] **Example:** `Device(config)# logging discriminator pacfltr3`	Creates a message discriminator.
Step 4	**logging console** [**discriminator** *discr-name*	**xml**] [*severity-level*] **Example:** `Device(config)# logging console discriminator pacfltr3 1`	Enables logging to the console and specifies a message discriminator filtering messages at a specific severity level.
Step 5	**end** **Example:** `Device(config)# end`	Returns the CLI to privileged EXEC mode.

Associating a Message Discriminator with Terminal Lines

Perform this task to associate a message discriminator with terminal lines and have messages display at a monitor.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **logging discriminator** *discr-name* [[**facility**] [**mnemonics**] [**msg-body**] {**drops** *string*| **includes** *string*}] [**severity** {**drops** *sev-num* | **includes** *sev-num*}] [**rate-limit** *msglimit*]
4. **logging monitor** [**discriminator** *discr-name*| **xml**] [*severity-level*]
5. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**logging discriminator** *discr-name* [[**facility**] [**mnemonics**] [**msg-body**] {**drops** *string*| **includes** *string*}] [**severity** {**drops** *sev-num*	**includes** *sev-num*}] [**rate-limit** *msglimit*] **Example:** `Device(config)# logging discriminator pacfltr4`	Creates a message discriminator.
Step 4	**logging monitor** [**discriminator** *discr-name*| **xml**] [*severity-level*] **Example:** `Device(config)# logging monitor discriminator pacfltr4 2`	Specifies a message discriminator named pacfltr4 and enables logging to the terminal lines of messages at severity level 2 and lower.

	Command or Action	Purpose
Step 5	**end** **Example:** `Device(config)# end`	Returns the CLI to privileged EXEC mode.

Enabling Message Counters

Perform this task to enable logging of debug, log, or syslog messages.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **logging message-counter** {**debug** | **log** | **syslog**}
4. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**logging message-counter** {**debug**	**log**	**syslog**} **Example:** `Device(config)# logging message-counter syslog`	Enables logging of syslog messages.
Step 4	**end** **Example:** `Device(config)# end`	Returns the CLI to privileged EXEC mode.

Adding and Removing a BEEP Session

Perform this task to add and remove a BEEP session.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **logging host** {{*ip-address* | *hostname*} [**vrf** *vrf-name*] | **ipv6**{*ipv6-address* | *hostname*}} [**discriminator** *discr-name* | [[**filtered** [**stream** *stream-id*] | **xml**]] [**transport** {[**beep** [**audit**] [**channel** *chnl-number*] [**sasl** *profile-name*] [**tls cipher** [*cipher-num*] **trustpoint** *trustpt-name*]]] | **tcp**[**audit**] | **udp**} [**port** *port-num*]] [**sequence-num-session**] [**session-id**{**hostname** | **ipv4** | **ipv6** | **string** *custom-string*}]
4. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**logging host** {{*ip-address* | *hostname*} [**vrf** *vrf-name*] | **ipv6**{*ipv6-address* | *hostname*}} [**discriminator** *discr-name* | [[**filtered** [**stream** *stream-id*] | **xml**]] [**transport** {[**beep** [**audit**] [**channel** *chnl-number*] [**sasl** *profile-name*] [**tls cipher** [*cipher-num*] **trustpoint** *trustpt-name*]]] | **tcp**[**audit**] | **udp**} [**port** *port-num*]] [**sequence-num-session**] [**session-id**{**hostname** | **ipv4** | **ipv6** | **string** *custom-string*}] **Example:** `Device(config)# logging host host3 transport beep port 600 channel 3`	Identifies a logging host and specifies the transport protocol, port, and channel for logging messages.

	Command or Action	Purpose
Step 4	**end** **Example:** `Device(config)# end`	Returns the CLI to privileged EXEC mode.

Configuration Examples for Reliable Delivery and Filtering for Syslog

Configuring Transport and Logging Example

```
Device(config)# do show running-config
 | include logging
logging buffered xml
logging host 209.165.201.1 transport udp port 601
Device(config)# logging host 209.165.201.1 transport beep port 600 channel 3
Device(config)# logging host 209.165.201.1 transport tcp port 602

Device(config)# show running-config | include logging
logging buffered xml
logging host 209.165.201.1 transport udp port 601
logging host 209.165.201.1 transport beep port 600 channel 3
logging host 209.165.201.1 transport tcp port 602
Device(config)#
```

Additional References for VRF-Aware Source Interfaces for Syslog Transactions

Related Documents

Related Topic	Document Title
Cisco IOS commands	*Cisco IOS Master Commands List, All Releases*
Network Management commands (including logging commands): complete command syntax, defaults, command mode, command history, usage guidelines, and examples	*Cisco IOS Network Management Command Reference*
Syslog logging	*Troubleshooting and Fault Management module*

Standards and RFCs

Standard/RFC	Title
No new or modified standards/RFCs are supported by this feature, and support for existing standards/RFCs has not been modified by this feature.	--

MIBs

MIB	MIBs Link
No new or modified MIBs are supported by this feature, and support for existing MIBs has not been modified by this feature.	To locate and download MIBs for selected platforms, Cisco software releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for Reliable Delivery and Filtering for Syslog

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 3: Feature Information for Reliable Delivery and Filtering for Syslog

Feature Name	Releases	Feature Information
Reliable Delivery and Filtering for Syslog	Cisco IOS XE Release 2.1	The Reliable Delivery and Filtering for Syslog feature allows a device to be customized for receipt of syslog messages. This feature provides for reliable and secure delivery for syslog messages using BEEP. Additionally it allows multiple sessions to a single logging host, independent of the underlying transport method, and provides a filtering mechanism called a message discriminator. In Cisco IOS XE Release 2.1, this feature was introduced on Cisco ASR 1000 Cisco ASR 1000 Series Aggregation Services Routers. The following commands were introduced or modified: **logging buffered**, **logging console**, **logging discriminator**, **logging host**, **logging message-counter**, **logging monitor**, **show logging**.