



System Monitoring Configuration Guide for Cisco NCS 5000 Series Routers, IOS XR Release 6.3.x

First Published: 2017-09-01

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 NON-INFRINGEMENT 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.

All printed copies and duplicate soft copies of this document are considered uncontrolled. See the current online version for the latest version.

Cisco has more than 200 offices worldwide. Addresses and phone numbers are listed on the Cisco website at www.cisco.com/go/offices.

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/c/en/us/about/legal/trademarks.html>. 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)

© 2017 Cisco Systems, Inc. All rights reserved.



CONTENTS

PREFACE

Preface vii

Changes to this Document vii

Communications, Services, and Additional Information vii

CHAPTER 1

New and Changed Feature Information 1

System Monitoring Features Added or Modified in IOS XR Release 6.3.x 1

CHAPTER 2

Implementing System Logging 3

Implementing System Logging 3

Prerequisites for Configuring System Logging 4

Configuring System Logging 5

Configuring Logging to the Logging Buffer 5

Configuring Logging to a Remote Server 5

Configuring Logging to Terminal Lines 6

Modifying Logging to Console Terminal 7

Modifying Time Stamp Format 7

Suppressing Duplicate Syslog Messages 7

Archiving System Logging Messages to a Local Storage Device 7

Platform Automated Monitoring 8

PAM Events 9

Disable and Re-enable PAM 11

Data Archiving in PAM 12

Files Collected by PAM Tool 12

CHAPTER 3

Implementing Alarm Log Correlation 15

Implementing Alarm Log Correlation 15

- Prerequisites for Implementing Alarm Log Correlation 15
- Information About Implementing Alarm Log Correlation 15
 - Alarm Logging and Debugging Event Management System 15
- Configuring Alarm Log Correlation 17
 - Configuring Logging Correlation Rules 17
 - Configuring a Logging Correlation Rule Set 18
 - Correlating a Root Cause and Non Root Cause Alarms 18
 - Configuring Logging Suppression Rules 18
 - Modifying Logging Events Buffer Settings 19
 - Modifying Logging Correlation Buffer Settings 19
 - Enabling Alarm Source Location Display Field for Bistate Alarms 19
 - Configuring SNMP Correlation Rules 19
 - Configuring SNMP Correlation Ruleset 20
- Alarm Logging Correlation-Details 20

CHAPTER 4

Onboard Failure Logging 25

- Prerequisites 25
- Information About OBFL 25

CHAPTER 5

Configuring and Managing Embedded Event Manager Policies 27

- Prerequisites for Configuring and Managing Embedded Event Manager Policies 28
- Information About Configuring and Managing Embedded Event Manager Policies 28
 - Event Management 28
 - System Event Processing 28
 - Embedded Event Manager Scripts 29
 - Embedded Event Manager Policy Tcl Command Extension Categories 29
 - Cisco File Naming Convention for Embedded Event Manager 30
 - Embedded Event Manager Built-in Actions 30
 - Application-specific Embedded Event Management 31
 - Event Detection and Recovery 32
 - System Manager Event Detector 32
 - Timer Services Event Detector 33
 - Syslog Event Detector 33
 - None Event Detector 34

Watchdog System Monitor Event Detector	34
Distributed Event Detectors	35
Embedded Event Manager Event Scheduling and Notification	35
Reliability Statistics	35
How to Configure and Manage Embedded Event Manager Policies	37
Configuring Environmental Variables	37
Registering Embedded Event Manager Policies	37
How to Write Embedded Event Manager Policies Using Tcl	38
Registering and Defining an EEM Tcl Script	38
Suspending EEM Policy Execution	39
Specifying a Directory for Storing EEM Policies	39
Programming EEM Policies with Tcl	39
Creating an EEM User Tcl Library Index	44
Creating an EEM User Tcl Package Index	47
EEM Policies Using TCL: Details	50

CHAPTER 6**Implementing IPFIX 315** 55

IP Flow Information Export (IPFIX) 315	55
IPFIX 315 Implementation Considerations	56
Configuring IPFIX 315	57



Preface

This product has reached end-of-life status. For more information, see the [End-of-Life and End-of-Sale Notices](#).

The *System Monitoring Configuration Guide for Cisco NCS 5000 Series Routers* preface contains these sections:

- [Changes to this Document, on page vii](#)
- [Communications, Services, and Additional Information, on page vii](#)

Changes to this Document

This table lists the changes made to this document since it was first published.

Date	Summary
September 2017	Initial release of this document.
March 2018	Republished for Release 6.3.2

Communications, Services, and Additional Information

- To receive timely, relevant information from Cisco, sign up at [Cisco Profile Manager](#).
- To get the business impact you're looking for with the technologies that matter, visit [Cisco Services](#).
- To submit a service request, visit [Cisco Support](#).
- To discover and browse secure, validated enterprise-class apps, products, solutions and services, visit [Cisco Marketplace](#).
- To obtain general networking, training, and certification titles, visit [Cisco Press](#).
- To find warranty information for a specific product or product family, access [Cisco Warranty Finder](#).

Cisco Bug Search Tool

[Cisco Bug Search Tool](#) (BST) is a web-based tool that acts as a gateway to the Cisco bug tracking system that maintains a comprehensive list of defects and vulnerabilities in Cisco products and software. BST provides you with detailed defect information about your products and software.



CHAPTER 1

New and Changed Feature Information

This table summarizes the new and changed feature information for the *System Monitoring Configuration Guide for Cisco NCS 5000 Series Routers*, and tells you where they are documented.

- [System Monitoring Features Added or Modified in IOS XR Release 6.3.x](#), on page 1

System Monitoring Features Added or Modified in IOS XR Release 6.3.x

Feature	Description	Changed in Release	Where Documented
None	No new features introduced	Not applicable	Not applicable



CHAPTER 2

Implementing System Logging

This module describes the tasks you need to implement logging services on the router.

The Cisco IOS XR Software provides basic logging services. Logging services provide a means to gather logging information for monitoring and troubleshooting, to select the type of logging information captured, and to specify the destinations of captured system logging (syslog) messages.

Feature History for Implementing System Logging

Release	Modification
Release 6.1.2	Platform Automated Monitoring (PAM) tool was introduced for all Cisco IOS XR 64-bit platforms.

- [Implementing System Logging](#) , on page 3

Implementing System Logging

System Logging (Syslog) is the standard application used for sending system log messages. Log messages indicates the health of the device and point to any encountered problems or simplify notification messages according to the severity level. The IOS XR router sends its syslog messages to a syslog process. By default, syslog messages will be sent to the console terminal. But, syslog messages can be send to destinations other than the console such as the logging buffer, syslog servers, and terminal lines.

Syslog Message Format

By default, the general format of syslog messages generated by the syslog process on the Cisco IOS XR software is as follows:

```
node-id : timestamp : process-name [pid] : % message category -group -severity -message  
-code : message-text
```

The following table describes the general format of syslog messages on Cisco IOS XR software.

Table 1: Format of Syslog Messages

Field	Description
node-id	Node from which the syslog message originated.

Field	Description
timestamp	Time stamp in the month day HH:MM:SS format, indicating when the message was generated. Note The time-stamp format can be modified using the service timestamps command.
process-name	Process that generated the syslog message.
size	Process ID (pid) of the process that generated the syslog message.
[pid]	Message category, group name, severity, and message code associated with the syslog message.
message-text	Text string describing the syslog message.

Syslog Message Severity Levels

In the case of logging destinations such as console terminal, syslog servers and terminal lines, you can limit the number of messages sent to a logging destination by specifying the severity level of syslog messages. However, for the logging buffer destination, syslog messages of all severity will be sent to it irrespective of the specified severity level. In this case, the severity level only limits the syslog messages displayed in the output of the command **show logging**, at or below specified value. The following table lists the severity level keywords that can be supplied for the severity argument and the corresponding UNIX syslog definitions in order from the most severe level to the least severe level.

Table 2: Syslog Message Severity Levels

Severity Keyword	Level	Description
emergencies	0	System unusable
alert	1	Immediate action needed
critical	2	Critical conditions
errors	3	Error conditions
warnings	4	Warning conditions
notifications	5	Normal but significant condition
informational	6	Informational messages only
debugging	7	Debugging messages

Prerequisites for Configuring System Logging

These prerequisites are required to configure the logging of system messages in your network operating center (NOC):

- You must be in a user group associated with a task group that includes the proper task IDs. The command reference guides include the task IDs required for each command. If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.
- You must have connectivity with syslog servers to configure syslog server hosts as the recipients for syslog messages.

Configuring System Logging

Perform the tasks in this section for configuring system logging as required.

Configuring Logging to the Logging Buffer

Syslog messages can be sent to multiple destinations including an internal circular buffer known as logging buffer. You can send syslog messages to the logging buffer using the **logging buffered** command.

Configuration Example

This example shows the configuration for sending syslog messages to the logging buffer. The size of the logging buffer is configured as 3000000 bytes. The default value for the size of the logging buffer is 2097152 bytes.

```
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# logging buffered 3000000
RP/0/RP0/CPU0:Router(config)# commit
```

Configuring Logging to a Remote Server

Syslog messages can be sent to destinations other than the console, such as logging buffer, syslog servers, snmp servers and terminal lines. You can send syslog messages to an external syslog server by specifying the ip address or hostname of the syslog server using the **logging** command. Also you can configure the syslog facility in which syslog messages are sent by using the **logging facility** command.

The following table list the features supported by Cisco IOS XR Software to help managing syslog messages sent to syslog servers.

Table 3: Features for Managing Syslog Messages

Features	Description
UNIX system log facility	Facility is the identifier used by UNIX to describe the application or process that submitted the log message. You can configure the syslog facility in which syslog messages are sent by using the logging facility command.

Features	Description
Hostname prefix logging	Cisco IOS XR Software supports hostname prefix logging. When enabled, hostname prefix logging appends a hostname prefix to syslog messages being sent from the router to syslog servers. You can use hostname prefixes to sort the messages being sent to a given syslog server from different networking devices. Use the logging hostname command to append a hostname prefix to syslog messages sent to syslog servers
Syslog source address logging	By default, a syslog message sent to a syslog server contains the IP address of the interface it uses to leave the router. Use the logging source-interface command to set all syslog messages to contain the same IP address, regardless of which interface the syslog message uses to exit the router.

Configuration Example for Logging to Syslog Server

This example shows the configuration for sending syslog messages to an external syslog server. The ip address 209.165.201.1 is configured as the syslog server.

```
Router# configure
Router(config)# logging 209.165.201.1 vrf default
Router(config)# logging facility kern (optional)
Router(config)# logging hostnameprefix 203.0.113.1 (optional)
Router(config)# logging source-interface TenGigE 0/0/0/10 (optional)
Router(config)# commit
```

Configuration Example for Logging to SNMP Server

This example shows the configuration for sending syslog messages to an SNMP server. The logging trap command is used to limit the logging of messages sent to the snmp servers based on severity.

```
Router# configure
Router(config)# snmp-server traps syslog
Router(config)# logging trap warnings
Router(config)# commit
```

For more information on SNMP server configurations, see the *Configuring Simple Network Management Protocol* chapter in the *System Management Configuration Guide for Cisco NCS 5000 Series Routers*

Configuring Logging to Terminal Lines

By default syslog messages will be sent to the console terminal. But, syslog messages can also be sent to terminal lines other than the console. You can send syslog messages to the logging buffer using the **logging monitor** command.

Configuration Example

This example shows the configuration for sending syslog messages to terminal lines other than console. In this example, severity level is configured as critical. The terminal monitor command is configured to display syslog messages during a terminal session. The default severity level is debugging.

```
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# logging monitor critical
RP/0/RP0/CPU0:Router(config)# commit
RP/0/RP0/CPU0:Router# terminal monitor
```

Modifying Logging to Console Terminal

By default syslog messages will be sent to the console terminal. You can modify the logging of syslog messages to the console terminal

Configuration Example

This example shows how to modify the logging of syslog messages to the console terminal.

```
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# logging console alerts
RP/0/RP0/CPU0:Router(config)# commit
```

Modifying Time Stamp Format

By default, time stamps are enabled for syslog messages. Time stamp is generated in the month day HH:MM:SS format indicating when the message was generated.

Configuration Example

This example shows how to modify the time-stamp for syslog and debugging messages.

```
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# service timestamps log datetime localtime msec or service
timestamps log uptime
RP/0/RP0/CPU0:Router(config)# service timestamps debug datetime msec show-timezone or service
timestamps debug uptime
RP/0/RP0/CPU0:Router(config)# commit
```

Suppressing Duplicate Syslog Messages

Suppressing duplicate messages, especially in a large network, can reduce message clutter and simplify the task of interpreting the log. The duplicate message suppression feature substantially reduces the number of duplicate event messages in both the logging history and the syslog file.

Configuration Example

This example shows how to suppress the consecutive logging of duplicate syslog messages.

```
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# logging suppress duplicates
RP/0/RP0/CPU0:Router(config)# commit
```

Archiving System Logging Messages to a Local Storage Device

Syslog messages can also be saved to an archive on a local storage device, such as the hard disk or a flash disk. Messages can be saved based on severity level, and you can specify attributes such as the size of the archive, how often messages are added (daily or weekly), and how many total weeks of messages the archive will hold. You can create a logging archive and specify how the logging messages will be collected and stored by using the **logging archive** command.

The following table lists the commands used to specify the archive attributes once you are in the logging archive submenu.

Table 4: Commands Used to Set Syslog Archive Attributes

Features	Description
archive-length weeks	Specifies the maximum number of weeks that the archive logs are maintained in the archive. Any logs older than this number are automatically removed from the archive.
archive-size size	Specifies the maximum total size of the syslog archives on a storage device. If the size is exceeded then the oldest file in the archive is deleted to make space for new logs.
device {disk0 disk1 harddisk}	Specifies the local storage device where syslogs are archived. By default, the logs are created under the directory device/var/log . If the device is not configured, then all other logging archive configurations are rejected. We recommend that syslogs be archived to the harddisk because it has more capacity than flash disks.
file-size size	Specifies the maximum file size (in megabytes) that a single log file in the archive can grow to. Once this limit is reached, a new file is automatically created with an increasing serial number.
frequency {daily weekly}	Specifies if logs are collected on a daily or weekly basis.
severity severity	Specifies the minimum severity of log messages to archive. All syslog messages greater than or equal to this configured level are archived while those lesser than this are filtered out.

Configuration Example

This example shows how to save syslog messages to an archive on a local storage device.

```
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# logging archive
RP/0/RP0/CPU0:Router(config-logging-arch)# device disk1
RP/0/RP0/CPU0:Router(config-logging-arch)# frequency weekly
RP/0/RP0/CPU0:Router(config-logging-arch)# severity warnings
RP/0/RP0/CPU0:Router(config-logging-arch)# archive-length 6
RP/0/RP0/CPU0:Router(config-logging-arch)# archive-size 50
RP/0/RP0/CPU0:Router(config-logging-arch)# file-size 10
RP/0/RP0/CPU0:Router(config)# commit
```

Platform Automated Monitoring

Platform Automated Monitoring (PAM) is a system monitoring tool integrated into Cisco IOS XR software image to monitor issues such as process crash, memory leak, CPU hog, tracebacks, syslog and disk usage. PAM is enabled by default on all Cisco IOS XR 64 bit platforms. When the PAM tool detects any of these

system issues, it collects the required data to troubleshoot the issue, and generates a syslog message stating the issue. The auto-collected troubleshooting information is then stored as a separate file in `harddisk:/cisco_support/` or in `/misc/disk1/cisco_support/` directory.

PAM Events

When PAM detects a process crash, traceback, potential memory leak, CPU hog or a full file system, it automatically collects logs and saves these logs (along with the core file in applicable cases) as a `.tgz` file in `harddisk:/cisco_support/` or in `/misc/disk1/cisco_support/` directory. PAM also generates a syslog message with severity level as warning, mentioning the respective issue.

The format of the `.tgz` file is: `PAM-<platform>-<PAM event>-<node-name>-<PAM process>-<YYYYMMDD>-<checksum>.tgz`. For example, `PAM-ncs5k-crash-xr_0_RP0_CPU0-ipv4_rib-2016Aug16-210405.tgz` is the file collected when PAM detects a process crash.

Because PAM assumes that core files are saved to the default archive folder (`harddisk:/` or `/misc/disk1/`), you must not modify the location of core archive (by configuring exception filepath) or remove the core files generated after PAM detects an event. Else, PAM does not detect the process crash. Also, once reported, the PAM does not report the same issue for the same process in the same node again.

For the list of commands used while collecting logs, refer [Files Collected by PAM Tool, on page 12](#).

The sections below describe the main PAM events:

Crash Monitoring

The PAM monitors process crash for all nodes, in real time. This is a sample syslog generated when the PAM detects a process crash:

```
RP/0/RP0/CPU0:Aug 16 21:04:06.442 : logger[69324]: %OS-SYSLOG-4-LOG_WARNING : PAM detected
  crash for ipv4_rib on 0_RP0_CPU0.
All necessary files for debug have been collected and saved at
0/RP0/CPU0 :
harddisk:/cisco_support/PAM-ncs5k-crash-xr_0_RP0_CPU0-ipv4_rib-2016Aug16-210405.tgz
Please copy tgz file out of the router and send to Cisco support. This tgz file will be
removed after 14 days.)
```

Traceback Monitoring

The PAM monitors tracebacks for all nodes, in real time. This is a sample syslog generated when the PAM detects a traceback:

```
RP/0/RP0/CPU0:Aug 16 21:42:42.320 : logger[66139]: %OS-SYSLOG-4-LOG_WARNING : PAM detected
  traceback for ipv4_rib on 0_RP0_CPU0.
All necessary files for debug have been collected and saved at
0/RP0/CPU0 :
harddisk:/cisco_support/PAM-ncs5k-traceback-xr_0_RP0_CPU0-ipv4_rib-2016Aug16-214242.tgz
Please copy tgz file out of the router and send to Cisco support. This tgz file will be
removed after 14 days.)
```

Memory Usage Monitoring

The PAM monitors the process memory usage for all nodes. The PAM detects potential memory leaks by monitoring the memory usage trend and by applying a proprietary algorithm to the collected data. By default, it collects top output on all nodes periodically at an interval of 30 minutes.

This is a sample syslog generated when the PAM detects a potential memory leak:

```
RP/0/RP0/CPU0:Aug 17 05:13:32.684 : logger[67772]: %OS-SYSLOG-4-LOG_WARNING : PAM detected
  significant memory increase
  (from 13.00MB at 2016/Aug/16/20:42:41 to 28.00MB at 2016/Aug/17/04:12:55) for
  pam_memory_leaker on 0_RP0_CPU0.
  All necessary files for debug have been collected and saved at
  0/RP0/CPU0 :
  harddisk:/cisco_support/PAM-ncs5k-memory_leak-xr_0_RP0_CPU0-pam_memory_leaker-2016Aug17-051332.tgz

  (Please copy tgz file out of the router and send to Cisco support. This tgz file will be
  removed after 14 days.)
```

CPU Monitoring

The PAM monitors CPU usage on all nodes periodically at an interval of 30 minutes. The PAM reports a CPU hog in either of these scenarios:

- When a process constantly consumes high CPU (that is, more than the threshold of 90 percentage)
- When high CPU usage lasts for more than 60 minutes

This is a sample syslog generated when the PAM detects a CPU hog:

```
RP/0/RP0/CPU0:Aug 16 00:56:00.819 : logger[68245]: %OS-SYSLOG-4-LOG_WARNING : PAM detected
  CPU hog for cpu_hogger on 0_RP0_CPU0.
  All necessary files for debug have been collected and saved at 0/RP0/CPU0 :
  harddisk:/cisco_support/PAM-ncs5k-cpu_hog-xr_0_RP0_CPU0-cpu_hogger-2016Aug16-005600.tgz
  (Please copy tgz file out of the router and send to Cisco support. This tgz file will be
  removed after 14 days.)
  RP/0/RP0/CPU0:Jun 21 15:33:54.517 : logger[69042]: %OS-SYSLOG-1-LOG_ALERT : PAM detected
  ifmgr is hogging CPU on 0_RP0_CPU0!
```

File System Monitoring

The PAM monitors disk usage on all nodes periodically at an interval of 30 minutes. This is a sample syslog generated when the PAM detects that a file system is full:

```
RP/0/RP0/CPU0:Jun 20 13:59:04.986 : logger[66125]: %OS-SYSLOG-4-LOG_WARNING : PAM detected
  /misc/config is full on 0_1_CPU0
  (please clean up to avoid any fault caused by this). All necessary files for debug have
  been collected and saved at
  0/RP0/CPU0 : harddisk:/cisco_support/PAM-ncs5k-disk_usage-xr_0_1_CPU0-2016Jun20-135904.tgz

  (Please copy tgz file out of the router and send to Cisco support. This tgz file will be
  removed after 14 days.)
```

Disable and Re-enable PAM

The PAM tool consists of three monitoring processes—`monitor_cpu.pl`, `monitor_crash.pl`, and `monitor_show_show_logging.pl`.

Before disabling or re-enabling the PAM, use these options to check if the PAM is installed in the router:

- From Cisco IOS XR Command Line Interface:

```
Router# show processes pam_manager location all
Tue Jun 14 17:58:42.791 UTC
node:      node0_RP0_CPU0
           Job Id: 317
           PID: 14070
           Executable path: /opt/cisco/XR/packages/iosxr-infra.rp-6.1.1.17I/bin/pam_manager

           Instance #: 1
           Version ID: 00.00.0000
           Respawn: ON
           Respawn count: 4
           Last started: Mon Jun 13 23:08:43 2016
           Process state: Run
           Package state: Normal
           core: MAINMEM
           Max. core: 0
           Level: 999
           Placement: None
           startup_path:
/opt/cisco/XR/packages/iosxr-infra.rp-6.1.1.17I/startup/pam_manager.startup
           Ready: 0.166s
           Process cpu time: 0.200 user, 0.310 kernel, 0.510 total
JID      TID  Stack  pri  state      NAME                rt_pri
317      14070  0K 20 Sleeping  pam_manager         0
317      14071  0K 20 Sleeping  lwm_debug_threa     0
317      14076  0K 20 Sleeping  pam_manager         0
317      14077  0K 20 Sleeping  lwm_service_thr     0
317      14078  0K 20 Sleeping  qsm_service_thr     0
317      14080  0K 20 Sleeping  pam_manager         0
```

- From router shell prompt:

```
Router# run ps auxw|egrep perl
Tue Jun 14 18:00:25.514 UTC
root      14324  0.0  0.2  84676 34556 ?  S Jun13   0:40 /usr/bin/perl
/pkg/opt/cisco/pam//monitor_cpu.pl
root      14414  0.0  0.1  65404 14620 ?  S Jun13   0:00 /usr/bin/perl
/pkg/opt/cisco/pam//monitor_crash.pl
```

Disable PAM

To shutdown PAM agents, execute these commands from the XR EXEC mode:

For local RP:

```
Router# process shutdown pam_manager
```

For all RPs:

```
Router# process shutdown pam_manager location all
```

Re-enable PAM

Because *pam_manager* is not a mandatory process, it does not restart automatically if it was manually disabled (unless in the case of a system reload). To restart PAM agents, execute the following commands from XR EXEC mode:

For local RP:

```
Router# process start pam_manager
```

For all RPs:

```
Router# process start pam_manager location all
```



Note To start PAM on all locations, the *pam_manager* process should be restarted on all nodes by using the **location all** option in the **process start pam_manager** command.

Data Archiving in PAM

At any given point of time, PAM does not occupy more than 200 MB of harddisk space. If more than 200 MB is needed, then PAM archives old files and rotates the logs automatically.

The PAM collects CPU or memory usage (using **top -b -n1** command) periodically at an interval of 30 minutes. The files are saved under `harddisk:/cisco_support/` directory with the filename as `<node name>.log` (for example, `harddisk:/cisco_support/xr-0_RP0_CPU0.log`). When the file size exceeds the limit of 15MB, the file is archived (compressed) into `.tgz` file, and then rotated for a maximum of two counts (that is, it retains only two `.tgz` files). The maximum rotation count of `.tgz` files is three. Also, the old file (ASCII data) is archived and rotated if a node is reloaded. For example, `xr-0_RP0_CPU0.log` is archived if RP0 is reloaded.

You must not manually delete the core file generated by the PAM. The core file is named as `<process name>_pid.by_user.<yyyymmdd>-<hhmmss>.<node>.<checksum>.core.gz`.

Files Collected by PAM Tool

The table below lists the various PAM events and the respective commands and files collected by the PAM for each event.

You can attach the respective `.tgz` file when you raise a service request (SR) with Cisco Technical Support.

Event Name	Commands and Files Collected by PAM
Process crash	<ul style="list-style-type: none"> • show install active • show platform • show version • core (gz) file • core.txt file

Event Name	Commands and Files Collected by PAM
Process traceback	<ul style="list-style-type: none"> • show dll • show install active • show logging • show platform • show version
Memory leak	<ul style="list-style-type: none"> • show install active • show platform • show version • core (gz) file • dumpcore running • continuous memory usage snapshots
Show logging event	<ul style="list-style-type: none"> • show install active • show logging • show platform • show version • core (gz) file • core.txt file
CPU hog	<ul style="list-style-type: none"> • follow process • pstack • show dll • show install active • show platform • show version • top -H • core (gz) file • CPU usage snapshots

Event Name	Commands and Files Collected by PAM
Disk usage	<ul style="list-style-type: none">• show install active• show platform• show version• console log• core (gz) file• Disk usage snapshots



CHAPTER 3

Implementing Alarm Log Correlation

This module describes the concepts and tasks related to configuring alarm log correlation. Alarm log correlation extends system logging to include the ability to group and filter messages generated by various applications and system servers and to isolate root messages on the router.

- [Implementing Alarm Log Correlation, on page 15](#)

Implementing Alarm Log Correlation

Alarm log correlation extends system logging to include the ability to group and filter messages generated by various applications and system servers and to isolate root messages on the router. This module describes the concepts and tasks related to configuring alarm log correlation and monitoring alarm logs.

Prerequisites for Implementing Alarm Log Correlation

You must be in a user group associated with a task group that includes the proper task IDs. The command reference guides include the task IDs required for each command. If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.

Information About Implementing Alarm Log Correlation

Alarm Logging and Debugging Event Management System

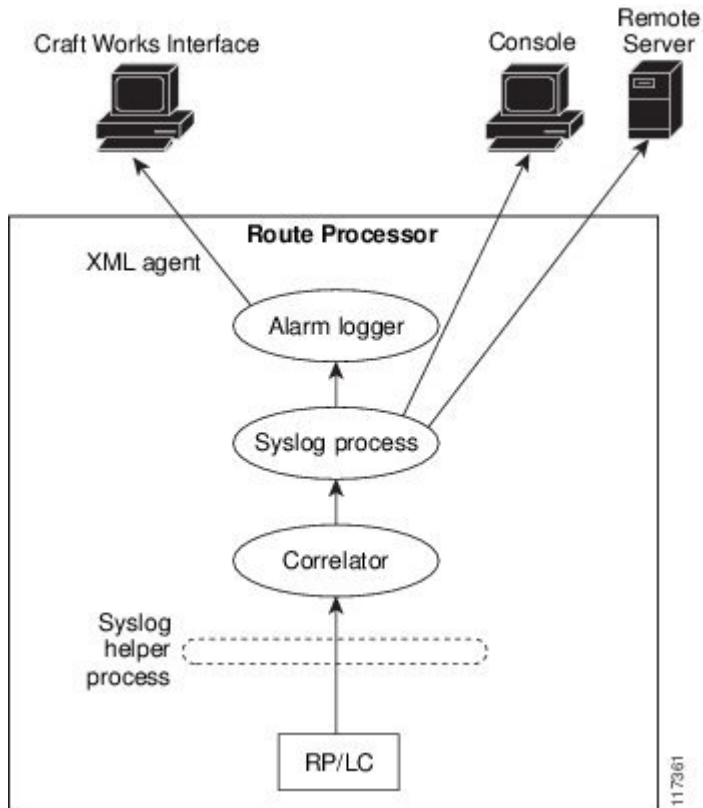
Cisco IOS XR Software Alarm Logging and Debugging Event Management System (ALDEMS) is used to monitor and store alarm messages that are forwarded by system servers and applications. In addition, ALDEMS correlates alarm messages forwarded due to a single root cause.

ALDEMS enlarges on the basic logging and monitoring functionality of Cisco IOS XR Software, providing the level of alarm and event management necessary for a highly distributed system with potentially hundreds of line cards and thousands of interfaces.

Cisco IOS XR Software achieves this necessary level of alarm and event management by distributing logging applications across the nodes on the system.

The following figure illustrates the relationship between the components that constitute ALDEMS.

Figure 1: ALDEMS Component Communications



Correlator

The correlator receives messages from system logging (syslog) helper processes that are distributed across the nodes on the router and forwards syslog messages to the syslog process. If a logging correlation rule is configured, the correlator captures messages searching for a match with any message specified in the rule. If the correlator finds a match, it starts a timer that corresponds to the timeout interval specified in the rule. The correlator continues searching for a match to messages in the rule until the timer expires. If the root case message was received, then a correlation occurs; otherwise, all captured messages are forwarded to the syslog. When a correlation occurs, the correlated messages are stored in the logging correlation buffer. The correlator tags each set of correlated messages with a correlation ID.

System Logging Process

The alarm logger is the final destination for system logging messages forwarded on the router. The alarm logger stores alarm messages in the logging events buffer. The logging events buffer is circular; that is, when full, it overwrites the oldest messages in the buffer.

Alarm Logger

The alarm logger is the final destination for system logging messages forwarded on the router. The alarm logger stores alarm messages in the logging events buffer. The logging events buffer is circular; that is, when full, it overwrites the oldest messages in the buffer.



Note Alarms are prioritized in the logging events buffer. When it is necessary to overwrite an alarm record, the logging events buffer overwrites messages in the following order: nonbistate alarms first, then bistate alarms in the CLEAR state, and, finally, bistate alarms in the SET state.

When the table becomes full of messages caused by bistate alarms in the SET state, the earliest bistate message (based on the message time stamp, not arrival time) is reclaimed before others. The buffer size for the logging events buffer and the logging correlation buffer, thus, should be adjusted so that memory consumption is within your requirements.

A table-full alarm is generated each time the logging events buffer wraps around. A threshold crossing notification is generated each time the logging events buffer reaches the capacity threshold.

Messages stored in the logging events buffer can be queried by clients to locate records matching specific criteria. The alarm logging mechanism assigns a sequential, unique ID to each alarm message.

Configuring Alarm Log Correlation

Perform the configuration tasks in this section to configure alarm log correlation as required.

Configuring Logging Correlation Rules

Logging correlation can be used to isolate the most significant root messages for events affecting system performance. When correlation rules are configured, a common root event that is generating secondary (non-root-cause) messages can be isolated and sent to the syslog, while secondary messages are suppressed. An operator can retrieve all correlated messages from the logging correlator buffer to view correlation events that have occurred. If a correlation rule is applied to the entire router, then correlation takes place only for those messages that match the configured cause values for the rule, regardless of the context or location setting of that message. If a correlation rule is applied to a specific set of contexts or locations, then correlation takes place only for those messages that match the configured cause values for the rule and that match at least one of those contexts or locations.

When a correlation rule is configured and applied, the correlator starts searching for a message match as specified in the rule. Timeout can be configured to specify the time interval for a message search once a match is found. Timeout begins when the correlator captures any alarm message specified for a correlation rule.

Configuration Example

This example shows how to configure and apply a logging correlation rule. In this example, timeout is configured as 60000 milliseconds.

```
Router# configure
Router(config)# logging correlator rule test type stateful
Router(config-corr-rule-st)# timeout 60000
Router(config)# logging correlator apply rule test
Router(config-corr-apply-rule)# all-of-router
or
Router(config-corr-apply-rule)# location 0/RP0/CPU0
or
Router(config-corr-apply-rule)# context rule_2
Router(config)# commit
```

Configuring a Logging Correlation Rule Set

You can configure a logging correlation rule set and include multiple correlation rules.

Configuration Example

This example shows how to configure and apply a logging correlation rule set for multiple correlation rules. The logging correlation rule set can be applied to the entire router or to a specific context or location.

```
Router# configure
Router(config)# logging correlator ruleset test
Router(config-corr-ruleset)# rulename test1
Router(config-corr-ruleset)# rulename test2
Router(config-corr-ruleset)# rulename test3
Router(config)# logging correlator apply ruleset test
Router(config-corr-apply-rule)# all-of-router
or
Router(config-corr-apply-rule)# location 0/RP0/CPU0
or
Router(config-corr-apply-rule)# context test123
Router(config)# commit
```

Correlating a Root Cause and Non Root Cause Alarms

The first message (with category, group, and code triplet) configured in a correlation rule defines the root-cause message. A root-cause message is always forwarded to the syslog process. You can correlate a root cause to one or more non-root-cause alarms and configure them as part of a rule.

Configuration Example

This example shows how to correlate a root cause to one or more non-root-cause alarms and configure them to a rule.

```
Router# configure
Router(config)# logging correlator rule rule_1 type stateful
Router(config-corr-rule-st)# rootcause CAT_b1 GROUP_b1 Root_b1
Router(config-corr-rule-st)# nonrootcause
Router(config-corr-rule-st-nonrc)# alarm CAT_b1 GROUP_b1 Root_b1
Router(config)# commit
```

Configuring Logging Suppression Rules

The alarm logging suppression feature enables you to suppress the logging of alarms by defining logging suppression rules that specify the types of alarms that you want to suppress. A logging suppression rule can specify all types of alarms or alarms with specific message categories, group names, and message codes. You can apply a logging suppression rule to alarms originating from all locations on the router or to alarms originating from specific nodes.

Configuration Example

This example shows how to configure logging suppression rules.

```
Router# configure
Router(config)# logging suppress rule infobistate
Router(config-suppr-rule)# alarm MBGL COMMIT SUCCEEDED
Router(config)# logging suppress apply rule infobistate
Router(config-suppr-apply-rule)# all-of-router
Router(config)# commit
```

Modifying Logging Events Buffer Settings

The alarm logger stores alarm messages in the logging events buffer. The logging events buffer overwrites the oldest messages in the buffer when it is full. Logging events buffer settings can be adjusted to respond to changes in user activity, network events, or system configuration events that affect network performance, or in network monitoring requirements. The appropriate settings depend on the configuration and requirements of the system. A threshold crossing notification is generated each time the logging events buffer reaches the capacity threshold.

Configuration Example

This example shows configuring the logging event buffer size, threshold, and alarm filter.

```
Router# configure terminal
Router(config)# logging events buffer-size 50000
Router(config)# logging events threshold 85
Router(config)# logging events level warnings
Router(config)# commit
```

Modifying Logging Correlation Buffer Settings

When a correlation occurs, the correlated messages are stored in the logging correlation buffer. The size of the logging correlation buffer can be adjusted to accommodate the anticipated volume of incoming correlated messages. Records can be removed from the buffer by specifying the records, or the buffer can be cleared of all records.

Configuration Example

This example shows configuring the correlation buffer size and removing the records from the buffer.

```
Router# configure terminal
Router(config)# logging correlator buffer-size 100000
Router(config)# exit
Router# clear logging correlator delete 48 49 50 (optional)
Router# clear logging correlator delete all-in-buffer (optional)
```

Enabling Alarm Source Location Display Field for Bistate Alarms

Bistate alarms are generated by state changes associated with system hardware. The bistate alarm message format is similar to syslog messages. You can optionally configure the output to include the location of the actual alarm source, which may be different from the process that logged the alarm. For more information about bistate alarms see, [Bistate Alarms, on page 21](#)

Configuration Example

This example shows how to enable the alarm source location display field for bistate alarms.

```
Router# configure
Router(config)# logging events display-location
Router(config)# commit
```

Configuring SNMP Correlation Rules

In large-scale systems, there may be situations when you encounter many SNMP traps emitted at regular intervals of time. These traps, in turn, cause additional time in the Cisco IOS XR processing of traps. The additional traps can also slow down troubleshooting and increases workload for the monitoring systems and the operators. SNMP alarm correlation helps to extract the generic pieces of correlation functionality from

the existing syslog correlator. You can configure correlation rules to define the correlation rules for SNMP traps and apply them to specific trap destinations.

Configuration Example

This example shows how to configure and apply correlation rules for SNMP traps. The SNMP correlator buffer size is also configured as 600 bytes. The default value for buffer size is 64KB.

```
Router# configure terminal
Router(config)# snmp-server correlator buffer-size 600 (optional)
Router(config)# snmp-server correlator rule test rootcause A varbind A1 value regex RA1
nonrootcause trap B varbind B1 index regex RB1
Router(config)# snmp-server correlator apply rule test host ipv4 address 1.2.3.4
Router(config)# commit
```

Configuring SNMP Correlation Ruleset

You can configure a SNMP correlation rule set and include multiple SNMP correlation rules.

Configuration Example

This example shows how to configure a ruleset that allows you to group two or more rules into a group. You can apply the specified group to a set of hosts or all of them.

```
RP/0/RP0/CPU0:Router# configure terminal
RP/0/RP0/CPU0:Router(config)# snmp-server correlator ruleset rule1 rulename rule2
RP/0/RP0/CPU0:Router(config)# snmp-server correlator apply ruleset rule1 host ipv4 address
1.2.3.4
RP/0/RP0/CPU0:Router(config)# commit
```

Alarm Logging Correlation-Details

Alarm logging correlation can be used to isolate the most significant root messages for events affecting system performance. For example, the original message describing a card online insertion and removal (OIR) of a line card can be isolated so that only the root-cause message is displayed and all subsequent messages related to the same event are correlated. When correlation rules are configured, a common root event that is generating secondary (non-root-cause) messages can be isolated and sent to the syslog, while secondary messages are suppressed. An operator can retrieve all correlated messages from the logging correlator buffer to view correlation events that have occurred.

Correlation Rules

Correlation rules can be configured to isolate root messages that may generate system alarms. Correlation rules prevent unnecessary stress on Alarm Logging and Debugging Event Management System (ALDEMS) caused by the accumulation of unnecessary messages. Each correlation rule depends on a message identification, consisting of a message category, message group name, and message code. The correlator process scans messages for occurrences of the message. If the correlator receives a root message, the correlator stores it in the logging correlator buffer and forwards it to the syslog process on the RP. From there, the syslog process forwards the root message to the alarm logger in which it is stored in the logging events buffer. From the syslog process, the root message may also be forwarded to destinations such as the console, remote terminals, remote servers, the fault management system, and the Simple Network Management Protocol (SNMP) agent, depending on the network device configuration. Subsequent messages meeting the same criteria (including another occurrence of the root message) are stored in the logging correlation buffer and are forwarded to the syslog process on the router.

If a message matches multiple correlation rules, all matching rules apply and the message becomes a part of all matching correlation queues in the logging correlator buffer. The following message fields are used to define a message in a logging correlation rule:

- Message category
- Message group
- Message code

Wildcards can be used for any of the message fields to cover wider set of messages.

There are two types of correlations configured in rules to isolate root-cause messages, stateful correlation and non-stateful correlation. Nonstateful correlation is fixed after it has occurred, and non-root-cause alarms that are suppressed are never forwarded to the syslog process. All non-root-cause alarms remain buffered in correlation buffers. Stateful correlation can change after it has occurred, if the bistate root-cause alarm clears. When the alarm clears, all the correlated non-root-cause alarms are sent to syslog and are removed from the correlation buffer. Stateful correlations are useful to detect non-root-cause conditions that continue to exist even if the suspected root cause no longer exists.

Alarm Severity Level and Filtering

Filter settings can be used to display information based on severity level. The alarm filter display indicates the severity level settings used to report alarms, the number of records, and the current and maximum log size.

Alarms can be filtered according to the severity level shown in this table.

Table 5: Alarm Severity Levels for Event Logging

Severity Level	System Condition
0	Emergencies
1	Alerts
2	Critical
3	Errors
4	Warnings
5	Notifications
6	Informational

Bistate Alarms

Bistate alarms are generated by state changes associated with system hardware, such as a change of interface state from active to inactive, the online insertion and removal (OIR) of a line card, or a change in component temperature. Bistate alarm events are reported to the logging events buffer by default; informational and debug messages are not.

Cisco IOS XR Software provides the ability to reset and clear alarms. Clients interested in monitoring alarms in the system can register with the alarm logging mechanism to receive asynchronous notifications when a monitored alarm changes state.

Bistate alarm notifications provide the following information:

- The origination ID, which uniquely identifies the resource that causes an alarm to be raised or cleared. This resource may be an interface, a line card, or an application-specific integrated circuit (ASIC). The origination ID is a unique combination of the location, job ID, message group, and message context.

By default, the general format of bistate alarm messages is the same as for all syslog messages:

```
node-id:timestamp : process-name [pid] : %category-group-severity-code : message-text
```

The following is a sample bistate alarm message:

```
LC/0/0/CPU0:Jan 15 21:39:11.325 2016:ifmgr[163]: %PKT_INFRA-LINEPRO
TO-5-UPDOWN : Line protocol on Interface TenGigE 0/0/0/0, changed state to Down
```

The message text includes the location of the process logging the alarm. In this example, the alarm was logged by the line protocol on TenGigE interface 0/0/0/0. Optionally, you can configure the output to include the location of the actual alarm source, which may be different from the process that logged the alarm. This appears as an additional display field before the message text.

When alarm source location is displayed, the general format becomes:

```
node-id:timestamp : process-name [pid] : %category-group-severity-code : source-location message-text
```

The following is a sample when alarm source location is displayed:

```
LC/0/0/CPU0:Jan 15 21:39:11.325 2016:ifmgr[163]: %PKT_INFRA-LINEPRO
TO-5-UPDOWN : interface TenGigE 0/0/0/0 : Line protocol on Interface TenGigE 0/0/0/0,
changed state to Down
```

Context Correlation Flag

The context correlation flag allows correlations to take place on a “per context” basis or not.

This flag causes behavior change only if the rule is applied to one or more contexts. It does not go into effect if the rule is applied to the entire router or location nodes.

The following is a scenario of context correlation behavior:

- Rule 1 has a root cause A and an associated non-root cause.
- Context correlation flag is not set on Rule 1.
- Rule 1 is applied to contexts 1 and 2.

If the context correlation flag is not set on Rule 1, a scenario in which alarm A generated from context 1 and alarm B generated from context 2 results in the rule applying to both contexts regardless of the type of context.

If the context correlation flag is now set on Rule 1 and the same alarms are generated, they are not correlated as they are from different contexts.

With the flag set, the correlator analyzes alarms against the rule only if alarms arrive from the same context. In other words, if alarm A is generated from context 1 and alarm B is generated from context 2, then a correlation does not occur.

Duration Timeout Flags

The root-cause timeout (if specified) is the alternative rule timeout to use in the situation in which a non-root-cause alarm arrives before a root-cause alarm in the given rule. It is typically used to give a shorter timeout in a situation under the assumption that it is less likely that the root-cause alarm arrives, and, therefore, releases the hold on the non-root-cause alarms sooner.



CHAPTER 4

Onboard Failure Logging

Onboard Failure Logging (OBFL) gathers boot, environmental, and critical hardware data for field-replaceable units (FRUs), and stores the information in the nonvolatile memory of the FRU. This information is used for troubleshooting, testing, and diagnosis if a failure or other error occurs, providing improved accuracy in hardware troubleshooting and root cause isolation analysis. Stored OBFL data can be retrieved in the event of a failure and is accessible even if the card does not boot.

Because OBFL is on by default, data is collected and stored as soon as the card is installed. If a problem occurs, the data can provide information about historical environmental conditions, uptime, downtime, errors, and other operating conditions.



Caution

OBFL is activated by default in FRUs. Do not deactivate OBFL without specific reasons, because the OBFL data is used to diagnose and resolve problems in FRUs.

- [Prerequisites](#) , on page 25
- [Information About OBFL](#), on page 25

Prerequisites

You must be in a user group associated with a task group that includes the proper task IDs. The command reference guides include the task IDs required for each command. If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.

Information About OBFL

OBFL feature is enabled by default. OBFL collects and stores both baseline and event-driven information in the nonvolatile memory of each supported card where OBFL is enabled. The data collected includes the following:

- FRU part serial number
- OS version
- Boot time
- Total run time

- Temperature and voltage at boot
- Temperature and voltage history

This data is collected in two different ways as baseline data and event- driven data.

Baseline Data Collection

Baseline data is stored independent of hardware or software failures and includes the information given in the following table.

Table 6: Data Types

Data Type	Details
Installation	Chassis serial number and slot number are stored at initial boot.
Temperature	Information on temperature sensors is recorded after boot. The subsequent recordings are specific to variations based on preset thresholds.
Run-time	Total run-time is limited to the size of the history buffer used for logging. This is based on the local router clock with logging granularity of 30 minutes.

Supported Cards and Platform

FRUs that have sufficient nonvolatile memory available for OBFL data storage support OBFL. The following table shows the card type and OBFL support.

Table 7: OBFL Support on Cisco NCS 5000 Series Router

Card Type	Cisco NCS 5000 Series Router
Route Processor (RP)	Supported
Power supply cards	Not Supported
Fan controller cards	Supported



CHAPTER 5

Configuring and Managing Embedded Event Manager Policies

The Cisco IOS XR Software Embedded Event Manager (EEM) functions as the central clearing house for the events detected by any portion of the Cisco IOS XR Software processor failover services. The EEM is responsible for detection of fault events, fault recovery, and process reliability statistics in a Cisco IOS XR Software system. The EEM events are notifications that something significant has occurred within the system, such as:

- Operating or performance statistics outside the allowable values (for example, free memory dropping below a critical threshold).
- Online insertion or removal (OIR).
- Termination of a process.

The EEM relies on software agents or event detectors to notify it when certain system events occur. When the EEM has detected an event, it can initiate corrective actions. Actions are prescribed in routines called *policies*. Policies must be registered before an action can be applied to collected events. No action occurs unless a policy is registered. A registered policy informs the EEM about a particular event that is to be detected and the corrective action to be taken if that event is detected. When such an event is detected, the EEM enables the corresponding policy. You can disable a registered policy at any time.

The EEM monitors the reliability rates achieved by each process in the system, allowing the system to detect the components that compromise the overall reliability or availability.

This module describes the tasks you need to perform to configure and manage EEM policies on your network and write and customize the EEM policies using Tool Command Language (Tcl) scripts to handle faults and events.

- [Prerequisites for Configuring and Managing Embedded Event Manager Policies, on page 28](#)
- [Information About Configuring and Managing Embedded Event Manager Policies, on page 28](#)
- [How to Configure and Manage Embedded Event Manager Policies, on page 37](#)

Prerequisites for Configuring and Managing Embedded Event Manager Policies

You must be in a user group associated with a task group that includes the proper task IDs. The command reference guides include the task IDs required for each command. If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.

Information About Configuring and Managing Embedded Event Manager Policies

Event Management

Embedded Event Manager (EEM) in the Cisco IOS XR Software system essentially involves system event management. An event can be any significant occurrence (not limited to errors) that has happened within the system. The Cisco IOS XR Software EEM detects those events and implements appropriate responses.

The EEM enables a system administrator to specify appropriate action based on the current state of the system. For example, a system administrator can use EEM to request notification by e-mail when a hardware device needs replacement.

The EEM interacts with routines, “event detectors,” that actively monitor the system for events. The EEM relies on an event detector that it has provided to syslog to detect that a certain system event has occurred. It uses a pattern match with the syslog messages and also relies on a timer event detector to detect that a certain time and date has occurred.

When the EEM has detected an event, it can initiate actions in response. These actions are contained in routines called policy handlers. Policies are defined by Tcl scripts (EEM scripts) written by the user through a Tcl API. While the data for event detection is collected, no action occurs unless a policy for responding to that event has been registered. At registration, a policy informs the EEM that it is looking for a particular event. When the EEM detects the event, it enables the policy.

The EEM monitors the reliability rates achieved by each process in the system. These metrics can be used during testing to determine which components do not meet their reliability or availability goals so that corrective action can be taken.

System Event Processing

When the EEM receives an event notification, it takes these actions:

- Checks for established policy handlers and if a policy handler exists, the EEM initiates callback routines (*EEM handlers*) or runs Tool Command Language (Tcl) scripts (*EEM scripts*) that implement policies. The policies can include built-in EEM actions.
- Notifies the processes that have *subscribed* for event notification.
- Records reliability metric data for each process in the system.
- Provides access to EEM-maintained system information through an application program interface (API).

Embedded Event Manager Scripts

When the EEM has detected an event, it can initiate corrective actions prescribed in routines called policies. Policies must be registered before any action can be applied to collected events. No action occurs unless a policy is registered. A registered policy informs the EEM about a particular event to detect and the corrective action to take if that event is detected. When such an event is detected, the EEM runs the policy. Tool Command Language (Tcl) is used as the scripting language to define policies and all Embedded Event Manager scripts are written in Tcl. EEM scripts are identified to the EEM using the **event manager policy** configuration command. An EEM script remains available to be scheduled by the EEM until the **no event manager policy** command is entered.

In addition the onboard Tcl scripts that come with the IOS XR operating system, users may write their own TCL-based policies. Cisco provides enhancements to the Tcl language in the form of Tcl command extensions that facilitate the writing of EEM policies. For more information about EEM Tcl command extensions, see [Embedded Event Manager Policy Tcl Command Extension Categories, on page 29](#)

Writing an EEM script includes the following steps:

- Selecting the event Tcl command extension that establishes the criteria used to determine when the policy is run.
- Defining the event detector options associated with detecting the event.
- Choosing the actions to implement recovery or respond to the detected event.

Embedded Event Manager Policy Tcl Command Extension Categories

This table lists the different categories of EEM policy Tcl command extensions.

Table 8: Embedded Event Manager Tcl Command Extension Categories

Category	Definition
EEM event Tcl command extensions(three types: event information, event registration, and event publish)	These Tcl command extensions are represented by the event_register_ xxx family of event-specific commands. There is a separate event information Tcl command extension in this category as well: event_reqinfo . This is the command used in policies to query the EEM for information about an event. There is also an EEM event publish Tcl command extension event_publish that publishes an application-specific event.
EEM action Tcl command extensions	These Tcl command extensions (for example, action_syslog) are used by policies to respond to or recover from an event or fault. In addition to these extensions, developers can use the Tcl language to implement any action desired.
EEM utility Tcl command extensions	These Tcl command extensions are used to retrieve, save, set, or modify application information, counters, or timers.
EEM system information Tcl command extensions	These Tcl command extensions are represented by the sys_reqinfo_ xxx family of system-specific information commands. These commands are used by a policy to gather system information.

Category	Definition
EEM context Tcl command extensions	These Tcl command extensions are used to store and retrieve a Tcl context (the visible variables and their values).

Cisco File Naming Convention for Embedded Event Manager

All EEM policy names, policy support files (for example, e-mail template files), and library filenames are consistent with the Cisco file-naming convention. In this regard, EEM policy filenames adhere to the following specifications:

- An optional prefix—Mandatory.—indicating, if present, that this is a system policy that should be registered automatically at boot time if it is not already registered; for example, Mandatory.sl_text.tcl.
- A filename body part containing a two-character abbreviation (see table below) for the first event specified; an underscore part; and a descriptive field part that further identifies the policy.
- A filename suffix part defined as .tcl.

EEM e-mail template files consist of a filename prefix of email_template, followed by an abbreviation that identifies the usage of the e-mail template.

EEM library filenames consist of a filename body part containing the descriptive field that identifies the usage of the library, followed by _lib, and a filename suffix part defined as .tcl.

Table 9: Two-Character Abbreviation Specification

Two-Character Abbreviation	Specification
ap	event_register_appl
ct	event_register_counter
st	event_register_stat
no	event_register_none
oi	event_register_oir
pr	event_register_process
sl	event_register_syslog
tm	event_register_timer
ts	event_register_timer_subscriber
wd	event_register_wdsysmon

Embedded Event Manager Built-in Actions

EEM built-in actions can be requested from EEM handlers when the handlers run.

This table describes each EEM handler request or action.

Table 10: Embedded Event Manager Built-In Actions

Embedded Event Manager Built-In Action	Description
Log a message to syslog	Sends a message to the syslog. Arguments to this action are priority and the message to be logged.
Execute a CLI command	Writes the command to the specified channel handler to execute the command by using the cli_exec command extension.
Generate a syslog message	Logs a message by using the action_syslog Tcl command extension.
Manually run an EEM policy	Runs an EEM policy within a policy while the event manager run command is running a policy in mode.
Publish an application-specific event	Publishes an application-specific event by using the event_publish appl Tcl command extension.
Reload the Cisco IOS software	Causes a router to be reloaded by using the EEM action_reload command.
Request system information	Represents the sys_reqinfo_xxx family of system-specific information commands by a policy to gather system information.
Send a short e-mail	Sends the e-mail out using Simple Mail Transfer Protocol (SMTP).
Set or modify a counter	Modifies a counter value.

EEM handlers require the ability to run CLI commands. A command is available to the Tcl shell to allow execution of CLI commands from within Tcl scripts.

Application-specific Embedded Event Management

Any Cisco IOS XR Software application can define and publish application-defined events. Application-defined events are identified by a name that includes both the component name and event name, to allow application developers to assign their own event identifiers. Application-defined events can be raised by a Cisco IOS XR Software component even when there are no subscribers. In this case, the EEM dismisses the event, which allows subscribers to receive application-defined events as needed.

An EEM script that subscribes to receive system events is processed in the following order:

1. This CLI configuration command is entered: **event manager policy scriptfilename username username**.
2. The EEM scans the EEM script looking for an **eem event event_type** keyword and subscribes the EEM script to be scheduled for the specified event.
3. The Event Detector detects an event and contacts the EEM.
4. The EEM schedules event processing, causing the EEM script to be run.
5. The EEM script routine returns.

Event Detection and Recovery

EEM is a flexible, policy-driven framework that supports in-box monitoring of different components of the system with the help of software agents known as event detectors. Event detectors are separate programs that provide an interface between other Cisco IOS XR Software components and the EEM. Event detectors (event publishers) screen events and publish them when there is a match on an event specification that is provided by event subscribers (policies). Event detectors notify the EEM server when an event of interest occurs.

An EEM event is defined as a notification that something significant has happened within the system. Two categories of events exist:

- System EEM events
- Application-defined events

System EEM events are built into the EEM and are grouped based on the fault detector that raises them. They are identified by a symbolic identifier defined within the API.

Some EEM system events are monitored by the EEM whether or not an application has requested monitoring. These are called *built-in* EEM events. Other EEM events are monitored only if an application has requested EEM event monitoring. EEM event monitoring is requested through an EEM application API or the EEM scripting interface.

Some event detectors can be distributed to other hardware cards within the same secure domain router (SDR) or within the administration plane to provide support for distributed components running on those cards.

These event detectors are supported:

System Manager Event Detector

The System Manager Event Detector has four roles:

- Records process reliability metric data.
- Screens for processes that have EEM event monitoring requests outstanding.
- Publishes events for those processes that match the screening criteria.
- Asks the System Manager to perform its default action for those events that do not match the screening criteria.

The System Manager Event Detector interfaces with the System Manager to receive process startup and termination notifications. The interfacing is made through a private API available to the System Manager. To minimize overhead, a portion of the API resides within the System Manager process space. When a process terminates, the System Manager invokes a helper process (if specified in the process.startup file) before calling the Event Detector API.

Processes can be identified by component ID, System Manager assigned job ID, or load module pathname plus process instance ID. Process instance ID is an integer assigned to a process to differentiate it from other processes with the same pathname. The first instance of a process is assigned an instance ID value of 1, the second 2, and so on.

The System Manager Event Detector handles EEM event monitoring requests for the EEM events shown in this table.

Table 11: System Manager Event Detector Event Monitoring Requests

Embedded Event Manager Event	Description
Normal process termination EEM event—built in	Occurs when a process matching the screening criteria terminates.
Abnormal process termination EEM event—built in	Occurs when a process matching the screening criteria terminates abnormally.
Process startup EEM event—built in	Occurs when a process matching the screening criteria starts.

When System Manager Event Detector abnormal process termination events occur, the default action restarts the process according to the built-in rules of the System Manager.

The relationship between the EEM and System Manager is strictly through the private API provided by the EEM to the System Manager for the purpose of receiving process start and termination notifications. When the System Manager calls the API, reliability metric data is collected and screening is performed for an EEM event match. If a match occurs, a message is sent to the System Manager Event Detector. In the case of abnormal process terminations, a return is made indicating that the EEM handles process restart. If a match does not occur, a return is made indicating that the System Manager should apply the default action.

Timer Services Event Detector

The Timer Services Event Detector implements time-related EEM events. These events are identified through user-defined identifiers so that multiple processes can await notification for the same EEM event.

The Timer Services Event Detector handles EEM event monitoring requests for the Date/Time Passed EEM event. This event occurs when the current date or time passes the specified date or time requested by an application.

Syslog Event Detector

The syslog Event Detector implements syslog message screening for syslog EEM events. This routine interfaces with the syslog daemon through a private API. To minimize overhead, a portion of the API resides within the syslog daemon process.

Screening is provided for the message severity code or the message text fields.

The Syslog Event Detector handles EEM event monitoring requests for the events are shown in this table.

Table 12: Syslog Event Detector Event Monitoring Requests

Embedded Event Manager Event	Description
Syslog message EEM event	Occurs for a just-logged message. It occurs when there is a match for either the syslog message severity code or the syslog message text pattern. Both can be specified when an application requests a syslog message EEM event.

Embedded Event Manager Event	Description
Process event manager EEM event—built in	Occurs when the event-processed count for a specified process is either greater than or equal to a specified maximum or is less than or equal to a specified minimum.

None Event Detector

The None Event Detector publishes an event when the Cisco IOS XR Software **event manager run** CLI command executes an EEM policy. EEM schedules and runs policies on the basis of an event specification that is contained within the policy itself. An EEM policy must be identified and registered to be permitted to run manually before the **event manager run** command will execute.

Event manager none detector provides user the ability to run a tcl script using the CLI. The script is registered first before running. Cisco IOS XR Software version provides similar syntax with Cisco IOS EEM (refer to the applicable EEM Documentation for details), so scripts written using Cisco IOS EEM is run on Cisco IOS XR Software with minimum change.

Watchdog System Monitor Event Detector

Watchdog System Monitor (IOSXRWDSysMon) Event Detector for Cisco IOS XR Software

The Cisco IOS XR Software Watchdog System Monitor Event Detector publishes an event when one of the following occurs:

- CPU utilization for a Cisco IOS XR Software process crosses a threshold.
- Memory utilization for a Cisco IOS XR Software process crosses a threshold.



Note Cisco IOS XR Software processes are used to distinguish them from Cisco IOS XR Software Modularity processes.

Two events may be monitored at the same time, and the event publishing criteria can be specified to require one event or both events to cross their specified thresholds.

The Cisco IOS XR Software Watchdog System Monitor Event Detector handles the events as shown in this table.

Table 13: Watchdog System Monitor Event Detector Requests

Embedded Event Manager Event	Description
Process percent CPU EEM event—built in	Occurs when the CPU time for a specified process is either greater than or equal to a specified maximum percentage of available CPU time or is less than or equal to a specified minimum percentage of available CPU time.
Total percent CPU EEM event—built in	Occurs when the CPU time for a specified processor complex is either greater than or equal to a specified maximum percentage of available CPU time or is less than or equal to a specified minimum percentage of available CPU time.

Embedded Event Manager Event	Description
Process percent memory EEM event—built in	Occurs when the memory used for a specified process has either increased or decreased by a specified value.
Total percent available Memory EEM event—built in	Occurs when the available memory for a specified processor complex has either increased or decreased by a specified value.
Total percent used memory EEM event—built in	Occurs when the used memory for a specified processor complex has either increased or decreased by a specified value.

Watchdog System Monitor (WDSysMon) Event Detector for Cisco IOS XR Software Modularity

The Cisco IOS XR Software Software Modularity Watchdog System Monitor Event Detector detects infinite loops, deadlocks, and memory leaks in Cisco IOS XR Software Modularity processes.

Distributed Event Detectors

Cisco IOS XR Software components that interface to EEM event detectors and that have substantially independent implementations running on a distributed hardware card should have a distributed EEM event detector. The distributed event detector permits scheduling of EEM events for local processes without requiring that the local hardware card to the EEM communication channel be active.

These event detectors run on a Cisco IOS XR Software line card:

- System Manager Fault Detector
- Wdsysmon Fault Detector
- Counter Event Detector
- OIR Event Detector
- Statistic Event Detector

Embedded Event Manager Event Scheduling and Notification

When an EEM handler is scheduled, it runs under the context of the process that creates the event request (or for EEM scripts under the Tcl shell process context). For events that occur for a process running an EEM handler, event scheduling is blocked until the handler exits. The defined default action (if any) is performed instead.

The EEM Server maintains queues containing event scheduling and notification items across client process restarts, if requested.

Reliability Statistics

Reliability metric data for the system is maintained by the EEM. The data is periodically written to checkpoint. Reliability metric data is kept for each hardware card and for each process handled by the System Manager.

Hardware Card Reliability Metric Data

Hardware card reliability metric data is recorded in a table indexed by disk ID.

Data maintained by the hardware card is as follows:

- Most recent start time
- Most recent normal end time (controlled switchover)
- Most recent abnormal end time (asynchronous switchover)
- Most recent abnormal type
- Cumulative available time
- Cumulative unavailable time
- Number of times hardware card started
- Number of times hardware card shut down normally
- Number of times hardware card shut down abnormally

Process Reliability Metric Data

Reliability metric data is kept for each process handled by the System Manager. This data includes standby processes running on either the primary or backup hardware card. Data is recorded in a table indexed by hardware card disk ID plus process pathname plus process instance for those processes that have multiple instances.

Process terminations include the following cases:

- Normal termination—Process exits with an exit value equal to 0.
- Abnormal termination by process—Process exits with an exit value not equal to 0.
- Abnormal termination by Linux—Linux operating system terminates the process.
- Abnormal termination by kill process API—API kill process terminates the process.

Data to be maintained by process is as follows:

- Most recent process start time
- Most recent normal process end time
- Most recent abnormal process end time
- Most recent abnormal process end type
- Previous ten process end times and types
- Cumulative process available time
- Cumulative process unavailable time
- Cumulative process run time (the time when the process is actually running on the CPU)
- Number of times started
- Number of times ended normally
- Number of times ended abnormally

- Number of abnormal failures within the past 60 minutes
- Number of abnormal failures within the past 24 hours
- Number of abnormal failures within the past 30 days

How to Configure and Manage Embedded Event Manager Policies

Configuring Environmental Variables

EEM environmental variables are Tcl global variables that are defined external to the policy before the policy is run. The EEM policy engine receives notifications when faults and other events occur. EEM policies implement recovery, based on the current state of the system and actions specified in the policy for a given event. Recovery actions are triggered when the policy is run.

By convention, the names of all environment variables defined by Cisco begin with an underscore character to set them apart; for example, `_show_cmd`.

You can configure the environment variable and values by using the **event manager environment** *var-name var-value* command.

Use the **show event manager environment** command to display the name and value of all EEM environment variables before and after they have been set using the **event manager environment** command.

Configuration Example

This example shows how to define a set of EEM environment variables.

```
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# event manager environment _cron_entry 0-59/2 0-23/1 * * 0-7
RP/0/RP0/CPU0:Router(config)# event manager environment _email_from beta@cisco.com
RP/0/RP0/CPU0:Router(config)# event manager environment _email_to beta@cisco.com
RP/0/RP0/CPU0:Router(config)# commit
RP/0/RP0/CPU0:Router(config)# end
RP/0/RP0/CPU0:Router# show event manager environment
```

No.	Name	Value
1	_email_to	beta@cisco.com
2	_cron_entry	0-59/2 0-23/1 * * 0-7
3	_email_from	beta@cisco.com

```
RP/0/RP0/CPU0:Router#
```

Registering Embedded Event Manager Policies

You should register an EEM policy to run a policy when an event is triggered. Registering an EEM policy is performed with the **event manager policy** command. An EEM script is available to be scheduled by the EEM until the **no** form of this command is entered. Prior to registering a policy, display EEM policies that are available to be registered with the **show event manager policy available** command.

The EEM schedules and runs policies on the basis of an event specification that is contained within the policy itself. When the **event manager policy** command is invoked, the EEM examines the policy and registers it to be run when the specified event occurs.

You need to specify the following while registering the EEM policy.

- **username**—Specifies the username that runs the script
- **persist-time**—Defines the number of seconds the username authentication is valid. This keyword is optional. The default **persist-time** is 3600 seconds (1 hour).
- **system** or **user**—Specifies the policy as a system defined or user defined policy. This keyword is optional.



Note AAA authorization (such as the **aaa authorization eventmanager** command) must be configured before EEM policies can be registered. See the *Configuring AAA Services* module of *Configuring AAA Services on Cisco IOS XR Software* for more information about AAA authorization configuration.

Once policies have been registered, their registration can be verified through the **show event manager policy registered** command.

Configuration Example

This example shows how to register a user defined EEM policy.

```
RP/0/RP0/CPU0:Router# show event manager policy available
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# event manager policy cron.tcl username tom type user
RP/0/RP0/CPU0:Router# show event manager policy registered
```

How to Write Embedded Event Manager Policies Using Tcl

This section provides information on how to write and customize Embedded Event Manager (EEM) policies using Tool Command Language (Tcl) scripts to handle Cisco IOS XR Software faults and events.

This section contains these tasks:

Registering and Defining an EEM Tcl Script

Perform this task to configure environment variables and register an EEM policy. EEM schedules and runs policies on the basis of an event specification that is contained within the policy itself. When an EEM policy is registered, the software examines the policy and registers it to be run when the specified event occurs.



Note A policy must be available that is written in the Tcl scripting language. Sample policies are stored in the system policy directory.

Configuration Example

This example shows how to register and define an EEM policy.

```
RP/0/RP0/CPU0:Router# show event manager environment all
RP/0/RP0/CPU0:Router# configure
```

```
RP/0/RP0/CPU0:Router(config)# event manager environment _cron_entry 0-59/2 0-23/1 * * 0-7
RP/0/RP0/CPU0:Router(config)# event manager policy tm_cli_cmd.tcl username user_a type
system
RP/0/RP0/CPU0:Router(config)# commit
RP/0/RP0/CPU0:Router# show event manager policy registered system
```



Note To unregister an EEM policy, use the **no event manager policy** command. This command removes an EEM policy from the running configuration file.

Suspending EEM Policy Execution

Suspending policies, instead of unregistering them, might be necessary for reasons of temporary performance or security. If required, you can immediately suspend the execution of all EEM policies by using the **event manager scheduler suspend** command.

Configuration Example

This example shows how to suspend the execution of all EEM policies.

```
RP/0/RP0/CPU0:Router# show event manager policy registered system
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# event manager scheduler suspend
RP/0/RP0/CPU0:Router(config)# commit
```

Specifying a Directory for Storing EEM Policies

A directory is essential to store the user-defined policy files or user library files. If you do not plan to write EEM policies, you do not have to create the directory. The EEM searches the user policy directory when you enter the **event manager policy *policy-name* user** command. To create a user policy directory before identifying it to the EEM, use the **mkdir** command. After creating the user policy directory, use the copy command to copy the policy files into the user policy directory. You can use the **show event manager directory user [library | policy]** command to display the directory to use for EEM user library files or user-defined policy files.

Configuration Example

This example shows how to specify a directory to use for storing user-library files .

```
RP/0/RP0/CPU0:Router# show event manager directory user library
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# event manager directory user library disk0:/usr/lib/tcl
RP/0/RP0/CPU0:Router(config)# commit
```

Programming EEM Policies with Tcl

Perform this task to help you program a policy using Tcl command extensions. We recommend that you copy an existing policy and modify it. There are two required parts that must exist in an EEM Tcl policy: the **event_register** Tcl command extension and the body. For detailed information about the Tcl policy structure and requirements, see [EEM Policies Using TCL: Details, on page 50](#)

SUMMARY STEPS

1. **show event manager policy available** [system | user]
2. Cut and paste the contents of the sample policy displayed on the screen to a text editor.
3. Define the required event_register Tcl command extension.
4. Add the appropriate namespace under the ::cisco hierarchy.
5. Program the must defines section to check for each environment variable that is used in this policy.
6. Program the body of the script.
7. Check the entry status to determine if a policy has previously run for this event.
8. Check the exit status to determine whether or not to apply the default action for this event, if a default action exists.
9. Set Cisco Error Number (_cerno) Tcl global variables.
10. Save the Tcl script with a new filename, and copy the Tcl script to the router.
11. **configure**
12. **event manager directory user** {library path | policy path}
13. **event manager policy** policy-name username username [persist-time [seconds | infinite] | type [system | user]]
14. Use the **commit** or **end** command.
15. Cause the policy to execute, and observe the policy.
16. Use debugging techniques if the policy does not execute correctly.

DETAILED STEPS

	Command or Action	Purpose
Step 1	show event manager policy available [system user] Example: <pre>RP/0/RP0/CPU0:Router# show event manager policy available</pre>	Displays EEM policies that are available to be registered.
Step 2	Cut and paste the contents of the sample policy displayed on the screen to a text editor.	—
Step 3	Define the required event_register Tcl command extension.	Choose the appropriate event_register Tcl command extension for the event that you want to detect, and add it to the policy. The following are valid Event Registration Tcl Command Extensions: <ul style="list-style-type: none"> • event_register_appl • event_register_counter • event_register_stat • event_register_wdsysmon • event_register_oir • event_register_process • event_register_syslog • event_register_timer

	Command or Action	Purpose
		<ul style="list-style-type: none"> • event_register_timer_subscriber • event_register_hardware • event_register_none
Step 4	Add the appropriate namespace under the ::cisco hierarchy.	<p>Policy developers can use the new namespace ::cisco in Tcl policies to group all the extensions used by Cisco IOS XR EEM. There are two namespaces under the ::cisco hierarchy. The following are the namespaces and the EEM Tcl command extension categories that belongs under each namespace:</p> <ul style="list-style-type: none"> • ::cisco::eem <ul style="list-style-type: none"> • EEM event registration • EEM event information • EEM event publish • EEM action • EEM utility • EEM context library • EEM system information • CLI library • ::cisco::lib <ul style="list-style-type: none"> • SMTP library <p>Note Ensure that the appropriate namespaces are imported, or use the qualified command names when using the preceding commands.</p>
Step 5	Program the must defines section to check for each environment variable that is used in this policy.	<p>This is an optional step. Must defines is a section of the policy that tests whether any EEM environment variables that are required by the policy are defined before the recovery actions are taken. The must defines section is not required if the policy does not use any EEM environment variables. EEM environment variables for EEM scripts are Tcl global variables that are defined external to the policy before the policy is run. To define an EEM environment variable, use the EEM configuration command event manager environment . By convention, all Cisco EEM environment variables begin with "_" (an underscore). To avoid future conflict, customers are urged not to define new variables that start with "_" .</p>

	Command or Action	Purpose
		<p>Note You can display the Embedded Event Manager environment variables set on your system by using the show event manager environment command.</p> <p>For example, EEM environment variables defined by the sample policies include e-mail variables. The sample policies that send e-mail must have the following variables set in order to function properly. The following are the e-mail-specific environment variables used in the sample EEM policies.</p> <ul style="list-style-type: none"> • _email_server—A Simple Mail Transfer Protocol (SMTP) mail server used to send e-mail (for example, mailserver.example.com) • _email_to—The address to which e-mail is sent (for example, engineering@example.com) • _email_from—The address from which e-mail is sent (for example, devtest@example.com) • _email_cc—The address to which the e-mail must be copied (for example, manager@example.com)
Step 6	Program the body of the script.	<p>In this section of the script, you can define any of the following:</p> <ul style="list-style-type: none"> • The event_reqinfo event information Tcl command extension that is used to query the EEM for information about the detected event. • The action Tcl command extensions, such as action_syslog, that are used to specify actions specific to EEM. • The system information Tcl command extensions, such as sys_reqinfo_routename, that are used to obtain general system information. • The context_save and context_retrieve Tcl command extensions that are used to save Tcl variables for use by other policies. • Use of the SMTP library (to send e-mail notifications) or the CLI library (to run CLI commands) from a policy.
Step 7	Check the entry status to determine if a policy has previously run for this event.	<p>If the prior policy is successful, the current policy may or may not require execution. Entry status designations may use one of three possible values: 0 (previous policy was successful), Not=0 (previous policy failed), and Undefined (no previous policy was executed).</p>

	Command or Action	Purpose
Step 8	Check the exit status to determine whether or not to apply the default action for this event, if a default action exists.	A value of zero means that the default action should not be performed. A value of nonzero means that the default action should be performed. The exit status is passed to subsequent policies that are run for the same event.
Step 9	Set Cisco Error Number (<code>_cerrno</code>) Tcl global variables.	Some EEM Tcl command extensions set a Cisco Error Number Tcl global variable <code>_cerrno</code> . Whenever <code>_cerrno</code> is set, four other Tcl global variables are derived from <code>_cerrno</code> and are set along with it (<code>_cerr_sub_num</code> , <code>_cerr_sub_err</code> , , and <code>_cerr_str</code>).
Step 10	Save the Tcl script with a new filename, and copy the Tcl script to the router.	<p>Embedded Event Manager policy filenames adhere to the following specification:</p> <ul style="list-style-type: none"> • An optional prefix—Mandatory.—indicating, if present, that this is a system policy that should be registered automatically at boot time if it is not already registered. For example: Mandatory.sl_text.tcl. • A filename body part containing a two-character abbreviation (see Table 9: Two-Character Abbreviation Specification, on page 30) for the first event specified, an underscore character part, and a descriptive field part further identifying the policy. • A filename suffix part defined as <code>.tcl</code>. <p>For more details, see the Cisco File Naming Convention for Embedded Event Manager, on page 30.</p> <p>Copy the file to the flash file system on the router—typically <code>disk0:</code>.</p>
Step 11	configure	Enters global configuration mode.
Step 12	event manager directory user {library path policy path} Example: <pre>RP/0/RP0/CPU0:Router(config)# event manager directory user library disk0:/user_library</pre>	Specifies a directory to use for storing user library files or user-defined EEM policies.
Step 13	event manager policy policy-name username username [persist-time [seconds infinite] type [system user]] Example: <pre>RP/0/RP0/CPU0:Router(config)# event manager policy test.tcl username user_a type user</pre>	Registers the EEM policy to be run when the specified event defined within the policy occurs.
Step 14	Use the commit or end command.	commit —Saves the configuration changes and remains within the configuration session.

	Command or Action	Purpose
		end —Prompts user to take one of these actions: <ul style="list-style-type: none"> • Yes — Saves configuration changes and exits the configuration session. • No —Exits the configuration session without committing the configuration changes. • Cancel —Remains in the configuration session, without committing the configuration changes.
Step 15	Cause the policy to execute, and observe the policy.	—
Step 16	Use debugging techniques if the policy does not execute correctly.	—

Creating an EEM User Tcl Library Index

Perform this task to create an index file that contains a directory of all the procedures contained in a library of Tcl files. This task allows you to test library support in EEM Tcl. In this task, a library directory is created to contain the Tcl library files, the files are copied into the directory, and an index `tclIndex` is created that contains a directory of all the procedures in the library files. If the index is not created, the Tcl procedures are not found when an EEM policy that references a Tcl procedure is run.

SUMMARY STEPS

1. On your workstation (UNIX, Linux, PC, or Mac) create a library directory and copy the Tcl library files into the directory.
2. **tclsh**
3. **auto_mkindex** *directory_name* *.tcl
4. Copy the Tcl library files from step 1 and the `tclIndex` file from step 3 to the directory used for storing user library files on the target router.
5. Copy a user-defined EEM policy file written in Tcl to the directory used for storing user-defined EEM policies on the target router.
6. **configure**
7. **event manager directory user library** *path*
8. **event manager directory user policy** *path*
9. **event manager policy** *policy-name* **username** *username* [**persist-time** [*seconds* | **infinite**]] | **type** [**system** | **user**]
10. **event manager run** *policy* [*argument*]
11. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	On your workstation (UNIX, Linux, PC, or Mac) create a library directory and copy the Tcl library files into the directory.	The following example files can be used to create a <code>tclIndex</code> on a workstation running the Tcl shell: lib1.tcl

	Command or Action	Purpose
		<pre>proc test1 {} { puts "In procedure test1" } proc test2 {} { puts "In procedure test2" } lib2.tcl proc test3 {} { puts "In procedure test3" }</pre>
<p>Step 2</p>	<p>tclsh</p> <p>Example:</p> <pre>workstation% tclsh</pre>	<p>Enters the Tcl shell.</p>
<p>Step 3</p>	<p>auto_mkindex <i>directory_name *.tcl</i></p> <p>Example:</p> <pre>workstation% auto_mkindex eem_library *.tcl</pre>	<p>Use the auto_mkindex command to create the tclIndex file. The tclIndex file contains a directory of all the procedures contained in the Tcl library files. We recommend that you run auto_mkindex inside a directory, because there can be only a single tclIndex file in any directory and you may have other Tcl files to be grouped together. Running auto_mkindex in a directory determines which Tcl source file or files are indexed using a specific tclIndex.</p> <p>The following sample TclIndex is created when the lib1.tcl and lib2.tcl files are in a library file directory and the auto_mkindex command is run:</p> <p>tclIndex</p> <pre># Tcl autoload index file, version 2.0 # This file is generated by the "auto_mkindex" command # and sourced to set up indexing information for one or # more commands. Typically each line is a command that # sets an element in the auto_index array, where the # element name is the name of a command and the value is # a script that loads the command. set auto_index(test1) [list source [file join \$dir lib1.tcl]] set auto_index(test2) [list source [file join \$dir lib1.tcl]] set auto_index(test3) [list source [file join \$dir lib2.tcl]]</pre>

	Command or Action	Purpose
Step 4	Copy the Tcl library files from step 1 and the tclIndex file from step 3 to the directory used for storing user library files on the target router.	—
Step 5	Copy a user-defined EEM policy file written in Tcl to the directory used for storing user-defined EEM policies on the target router.	<p>The directory can be the same directory used in step 4.</p> <p>The following example user-defined EEM policy can be used to test the Tcl library support in EEM:</p> <p>libtest.tcl</p> <pre> ::cisco::eem::event_register_none namespace import ::cisco::eem::* namespace import ::cisco::lib::* global auto_index auto_path puts [array names auto_index] if { [catch {test1} result]} { puts "calling test1 failed result = \$result \$auto_path" } if { [catch {test2} result]} { puts "calling test2 failed result = \$result \$auto_path" } if { [catch {test3} result]} { puts "calling test3 failed result = \$result \$auto_path" } </pre>
Step 6	<p>configure</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router# configure</pre>	Enters mode.
Step 7	<p>event manager directory user library path</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:Router(config)# event manager directory user library disk0:/eem_library</pre>	Specifies the EEM user library directory; this is the directory to which the files in step 4 were copied.
Step 8	<p>event manager directory user policy path</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:Router(config)# event manager directory user policy disk0:/eem_policies</pre>	Specifies the EEM user policy directory; this is the directory to which the file in step 5 was copied.
Step 9	<p>event manager policy policy-name username username [persist-time [seconds infinite] type [system user]]</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:Router(config)# event manager policy libtest.tcl username user_a</pre>	Registers a user-defined EEM policy.

	Command or Action	Purpose
Step 10	event manager run <i>policy</i> [<i>argument</i>] Example: <pre>RP/0/RP0/CPU0:Router(config)# event manager run libtest.tcl</pre>	Manually runs an EEM policy.
Step 11	Use the commit or end command.	commit —Saves the configuration changes and remains within the configuration session. end —Prompts user to take one of these actions: <ul style="list-style-type: none"> • Yes — Saves configuration changes and exits the configuration session. • No —Exits the configuration session without committing the configuration changes. • Cancel —Remains in the configuration session, without committing the configuration changes.

Creating an EEM User Tcl Package Index

Perform this task to create a Tcl package index file that contains a directory of all the Tcl packages and version information contained in a library of Tcl package files. Tcl packages are supported using the Tcl **package** keyword.

Tcl packages are located in either the EEM system library directory or the EEM user library directory. When a **package require** Tcl command is executed, the user library directory is searched first for a `pkgIndex.tcl` file. If the `pkgIndex.tcl` file is not found in the user directory, the system library directory is searched.

In this task, a Tcl package directory—the `pkgIndex.tcl` file—is created in the appropriate library directory using the **pkg_mkIndex** command to contain information about all the Tcl packages contained in the directory along with version information. If the index is not created, the Tcl packages are not found when an EEM policy that contains a **package require** Tcl command is run.

Using the Tcl package support in EEM, users can gain access to packages such as XML_RPC for Tcl. When the Tcl package index is created, a Tcl script can easily make an XML-RPC call to an external entity.



Note Packages implemented in C programming code are not supported in EEM.

SUMMARY STEPS

1. On your workstation (UNIX, Linux, PC, or Mac) create a library directory and copy the Tcl package files into the directory.
2. **tclsh**
3. **pkg_mkindex** *directory_name* *.tcl
4. Copy the Tcl package files from step 1 and the `pkgIndex` file from step 3 to the directory used for storing user library files on the target router.

5. Copy a user-defined EEM policy file written in Tcl to the directory used for storing user-defined EEM policies on the target router.
6. **configure**
7. **event manager directory user library path**
8. **event manager directory user policy path**
9. **event manager policy policy-name username username [persist-time [seconds | infinite] | type [system | user]]**
10. **event manager run policy [argument]**
11. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	On your workstation (UNIX, Linux, PC, or Mac) create a library directory and copy the Tcl package files into the directory.	–
Step 2	tclsh Example: <pre>workstation% tclsh</pre>	Enters the Tcl shell.
Step 3	pkg_mkindex directory_name *.tcl Example: <pre>workstation% pkg_mkindex eem_library *.tcl</pre>	<p>Use the pkg_mkindex command to create the pkgIndex file. The pkgIndex file contains a directory of all the packages contained in the Tcl library files. We recommend that you run the pkg_mkindex command inside a directory, because there can be only a single pkgIndex file in any directory and you may have other Tcl files to be grouped together. Running the pkg_mkindex command in a directory determines which Tcl package file or files are indexed using a specific pkgIndex.</p> <p>The following example pkgIndex is created when some Tcl package files are in a library file directory and the pkg_mkindex command is run:</p> <p>pkgIndex</p> <pre># Tcl package index file, version 1.1 # This file is generated by the "pkg_mkIndex" command # and sourced either when an application starts up or # by a "package unknown" script. It invokes the # "package ifneeded" command to set up package-related # information so that packages will be loaded automatically # in response to "package require" commands. When this # script is sourced, the variable \$dir must contain the # full path name of this file's directory.</pre>

	Command or Action	Purpose
		<pre>package ifneeded xmlrpc 0.3 [list source [file join \$dir xmlrpc.tcl]]</pre>
Step 4	Copy the Tcl package files from step 1 and the pkgIndex file from step 3 to the directory used for storing user library files on the target router.	—
Step 5	Copy a user-defined EEM policy file written in Tcl to the directory used for storing user-defined EEM policies on the target router.	<p>The directory can be the same directory used in step 4.</p> <p>The following example user-defined EEM policy can be used to test the Tcl library support in EEM:</p> <p>packagetest.tcl</p> <pre>::cisco::eem::event_register_none maxrun 1000000.000 # # test if xmlrpc available # # Namespace imports # namespace import ::cisco::eem::* namespace import ::cisco::lib::* # package require xmlrpc puts "Did you get an error?"</pre>
Step 6	<p>configure</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:router# configure</pre>	Enters mode.
Step 7	<p>event manager directory user library path</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:Router(config)# event manager directory user library disk0:/eem_library</pre>	Specifies the EEM user library directory; this is the directory to which the files in step 4 were copied.
Step 8	<p>event manager directory user policy path</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:Router(config)# event manager directory user policy disk0:/eem_policies</pre>	Specifies the EEM user policy directory; this is the directory to which the file in step 5 was copied.
Step 9	<p>event manager policy policy-name username username [persist-time [seconds infinite] type [system user]]</p> <p>Example:</p> <pre>RP/0/RP0/CPU0:Router(config)# event manager policy packagetest.tcl username user_a</pre>	Registers a user-defined EEM policy.

	Command or Action	Purpose
Step 10	event manager run <i>policy</i> [<i>argument</i>] Example: <pre>RP/0/RP0/CPU0:Router(config)# event manager run packagetest.tcl</pre>	Manually runs an EEM policy.
Step 11	Use the commit or end command.	commit —Saves the configuration changes and remains within the configuration session. end —Prompts user to take one of these actions: <ul style="list-style-type: none"> • Yes — Saves configuration changes and exits the configuration session. • No —Exits the configuration session without committing the configuration changes. • Cancel —Remains in the configuration session, without committing the configuration changes.

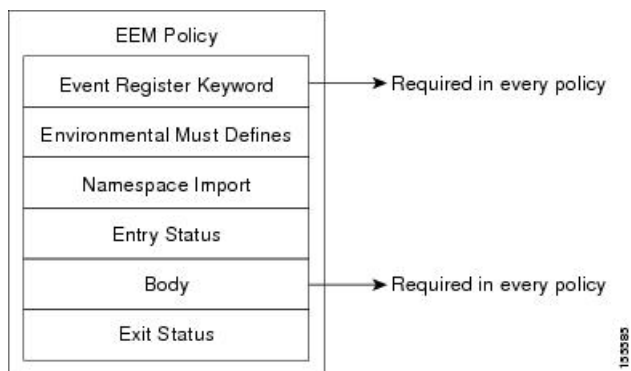
EEM Policies Using TCL: Details

This section provides detailed conceptual information about programming EEM policies using TCL.

Tcl Policy Structure and Requirements

All EEM policies share the same structure, shown in the below figure. There are two parts of an EEM policy that are required: the `event_register` Tcl command extension and the body. The remaining parts of the policy are optional: environmental must defines, namespace import, entry status, and exit status.

Figure 2: Tcl Policy Structure and Requirements



The start of every policy must describe and register the event to detect using an **event_register** Tcl command extension. This part of the policy schedules the running of the policy. The following example Tcl code shows how to register the **event_register_timer** Tcl command extension:

```
::cisco::eem::event_register_timer cron name crontimer2 cron_entry $_cron_entry maxrun 240
```

The following example Tcl code shows how to check for, and define, some environment variables:

```

# Check if all the env variables that we need exist.
# If any of them does not exist, print out an error msg and quit.
if {[info exists _email_server]} {
    set result \
        "Policy cannot be run: variable _email_server has not been set"
    error $result $errorInfo
}
if {[info exists _email_from]} {
    set result \
        "Policy cannot be run: variable _email_from has not been set"
    error $result $errorInfo
}
if {[info exists _email_to]} {
    set result \
        "Policy cannot be run: variable _email_to has not been set"
    error $result $errorInfo
}
)

```

The namespace import section is optional and defines code libraries. The following example Tcl code shows how to configure a namespace import section:

```

namespace import ::cisco::eem:*
namespace import ::cisco::lib:*

```

The body of the policy is a required structure and might contain the following:

- The **event_reqinfo** event information Tcl command extension that is used to query the EEM for information about the detected event.
- The action Tcl command extensions, such as **action_syslog**, that are used to specify actions specific to EEM.
- The system information Tcl command extensions, such as **sys_reqinfo_routername**, that are used to obtain general system information.
- Use of the SMTP library (to send e-mail notifications) or the CLI library (to run CLI commands) from a policy.
- The **context_save** and **context_retrieve** Tcl command extensions that are used to save Tcl variables for use by other policies.

EEM Entry Status

The entry status part of an EEM policy is used to determine if a prior policy has been run for the same event, and to determine the exit status of the prior policy. If the `_entry_status` variable is defined, a prior policy has already run for this event. The value of the `_entry_status` variable determines the return code of the prior policy.

Entry status designations may use one of three possible values:

- 0 (previous policy was successful)
- Not=0 (previous policy failed),
- Undefined (no previous policy was executed).

EEM Exit Status

When a policy finishes running its code, an exit value is set. The exit value is used by the EEM to determine whether or not to apply the default action for this event, if any. A value of zero means that the default action should not be performed. A value of nonzero means that the default action should be performed. The exit status is passed to subsequent policies that are run for the same event.

EEM Policies and Cisco Error Number

Some EEM Tcl command extensions set a Cisco Error Number Tcl global variable known as `_cerno`. Whenever the `_cerno` variable is set, the other Tcl global variables are derived from `_cerno` and are set along with it (`_cerr_sub_num`, `_cerr_sub_err`, and `_cerr_str`).

The `_cerno` variable set by a command can be represented as a 32-bit integer of the following form:

```
XYSSSSSSSSSSSSSEEEEEEEEEPPPPPPPP
```

This 32-bit integer is divided up into the variables shown in this table.

Table 14: `_cerno`: 32-Bit Error Return Value Variables

Variable	Description
XY	The error class (indicates the severity of the error). This variable corresponds to the first two bits in the 32-bit error return value; 10 in the preceding case, which indicates <code>CERR_CLASS_WARNING</code> . See Table 15: Error Class Encodings, on page 53 for the four possible error class encodings specific to this variable.
SSSSSSSSSSSSSS	The subsystem number that generated the most recent error (13 bits = 8192 values). This is the next 13 bits of the 32-bit sequence, and its integer value is contained in <code>\$_cerr_sub_num</code> .
EEEEEEEE	The subsystem specific error number (8 bits = 256 values). This segment is the next 8 bits of the 32-bit sequence, and the string corresponding to this error number is contained in <code>\$_cerr_sub_err</code> .

For example, the following error return value might be returned from an EEM Tcl command extension:

```
862439AE
```

This number is interpreted as the following 32-bit value:

```
10000110001001000011100110101110
```

The variable, `XY`, references the possible error class encodings shown in this table.

Table 15: Error Class Encodings

Error Return Value	Error Class
00	CERR_CLASS_SUCCESS
01	CERR_CLASS_INFO
10	CERR_CLASS_WARNING
11	CERR_CLASS_FATAL

An error return value of zero means SUCCESS.



CHAPTER 6

Implementing IPFIX 315

This module describes the concepts and tasks related to Internet Protocol Flow Information Export (IPFIX) 315. IPFIX is an IETF standard export protocol for sending IP flow information.

Table 16: Feature History for Implementing System Logging

Release	Modification
Release 6.2.1	IPFIX 315 feature was introduced.

- [IP Flow Information Export \(IPFIX\) 315, on page 55](#)
- [IPFIX 315 Implementation Considerations, on page 56](#)
- [Configuring IPFIX 315, on page 57](#)

IP Flow Information Export (IPFIX) 315

Internet Protocol Flow Information Export (IPFIX) is an IETF standard export protocol (RFC 7011) for sending IP flow information. Cisco NCS 5000 Router supports IPFIX 315 format to export flow information. IPFIX 315 format facilitates sending 'n' octets frame information starting from ethernet header till transport header of the traffic flow over the network. IPFIX 315 supports sending variable size packet record with variable payload information such as IPv4, IPv6, MPLS, and Nested packets like OuterIP-GRE-InnerIP etc. The process includes sampling and exporting the traffic flow information. Along with the ethernet frame information, IPFIX 315 format exports information of incoming and outgoing interface of the sampled packet.

The information of the packets flowing through a device is used for variety of purpose including network monitoring, capacity planning, traffic management, etc.



Note Cisco NCS 5000 Router does not support Netflow version 9 format to export flow information.

Sampling and Exporting Information

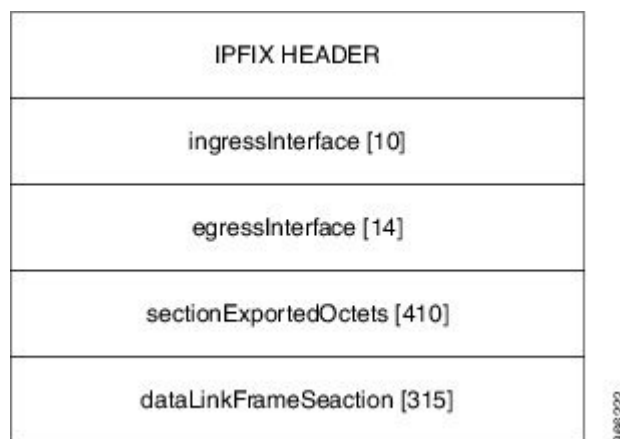
You must configure a sampling map to sample the traffic flow information. The sampler map specifies the rate at which packets (one out of n packets) are sampled. The minimum sampling rate is 1 out of 32,000 packets. Not all packets flowing through a device are exported; packets selected as per sampling rate are considered for exporting.

The size of exported packet depends on the sampled packet size and location of payload L4 header. The exported packet size is:

- 160 bytes, if the sampled packet size is more than 160 bytes and payload L4 header is not obtained within first 160 bytes
- length of the sampled packet until the payload L4 header, if the payload L4 header is obtained within 160 bytes
- equal to length of the packet, if packet size is less than 160 bytes and the payload L4 header is not obtained within first 160 bytes

The below figure *IPFIX 315 Export Packet Format* shows exported packet information.

Figure 3: IPFIX 315 Export Packet Format



A special cache type called Immediate Aging is used while exporting the packets. Immediate Aging ensures that the flows are exported as soon as they are added to the cache. Use the command **cache immediate** in flow monitor map configuration to enable Immediate Aging cache type.

IPFIX 315 Implementation Considerations

Here are few key points to consider before implementing IPFIX 315:

- Supported only in ingress direction.
- Supported on main interface only. The traffic on all sub-interfaces under the main interface is exported.
- Sampling rate for bundles is per member-link and not per bundle interface.
- The outgoing interface information may not be correct incase of packets that are multicasted or broadcasted on multiple ports.
- The incoming and outgoing interface will have information of main interface and not the sub-interface even if the packet is routed via sub-interface. Incase of bundles it will point to bundle main interface.
- IPFIX 315 is not supported on BVI interface.
- Sampling and exporting of the control packets is not supported.

- When you configure **ipfix315-enable**, then you must configure all the ports on that LC with `datalinkframesection flow`.
- When the HQoS profile is enabled, Netflow does not give correct Output Interface. DSP is unique for each sub-interface.
- Netflow on the L2 interface assumes IPv4/IPv6/MPLS traffic, and if the traffic is purely L2 based, then the system ignores that traffic.
- You must remove all v9 configurations before reloading an LC. Else, with the existing v9 configurations on LC reload, you might encounter a few configuration apply error. Or, flow might be seen on an interface even when apply on interface has failed.

Configuring IPFIX 315

Configuring IPFIX 315 involves:

1. Configuring Exporter map
2. Configuring Monitor map
3. Configuring Sampler map
4. Applying the Monitor map and Sampler map to an interface

Configuring Exporter map

```
flow exporter-map ipfix_exp
version ipfix
!
dscp 40
transport udp 9001
source Loopback1
destination 100.10.1.159
!
```

Configuring Monitor map

```
flow monitor-map ipfix_mon
record datalinksectiondump
exporter ipfix_exp
cache immediate
cache entries 1000000
cache timeout rate-limit 1000000
!
```

Configuring Sampler map

```
sampler-map ipfix_sm
random 1 out-of 32000
!
```



Note The default cache size is 65535, hence you can configure sampling rate as 1 out of 65535 packets. However the recommended sampling rate is 1 out of 32000 packets.

Applying the Monitor map to an interface

```
interface HundredGigE 0/0/0/18
    flow datalinkframesection monitor ipfix_mon sampler ipfix_sm ingress
```

Verification

Use the **show flow platform producer statistics location** command to display the sampled and exported flow statistics:

```
RP/0/RP0/CPU0#show flow platform producer statistics location 0/RP0/CPU0
Netflow Platform Producer Counters:|
Ethernet Packets: 1152
Drops (no space): 0
Drops (other): 0
Unknown Ingress Packets: 0
Worker waiting: 0
SPP Packets: 1102
Flow Packets: 1152
Flow Packets per SPP Frame: 1
```

Use the **show flow monitor <monitor-map> cache location** command to check the flow monitor stats. In this example flow statistics for *ipfix_mon* monitor map are displayed:

```
RP/0/RP0/CPU0#show flow monitor ipfix_mon cache location 0/RP0/CPU0

Cache summary for Flow Monitor ipfix_mon:
Cache size: 1000000
Current entries: 0|
Flows added: 1152
Flows not added: 0
Ager Polls: 48
- Active timeout 0
- Inactive timeout 0
- Immediate 1152 /*cache type immediate*/
- TCP FIN flag 0
- Emergency aged 0
- Counter wrap aged 0
- Total 1152
Periodic export:
- Counter wrap 0
- TCP FIN flag 0|
Flows exported 1152
```

Above example shows that there were 1152 flows added to the cache and exported.