System Monitoring Configuration Guide for Cisco 8000 Series Routers, IOS XR Release 7.0.x

First Published: 2020-03-01

Last Modified: 2020-08-01

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

CONTENTS

PREFACE

Preface ix

Changes to this Document ix

Communications, Services, and Additional Information ix

CHAPTER 1

New and Changed System Monitoring Features 1

System Monitoring Features Added or Modified in IOS XR Release 7.0.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 9

PAM Events 9

Disable and Re-enable PAM 11

Data Archiving in PAM 11

Files Collected by PAM Tool 12

CHAPTER 3

Monitoring Alarms and Implementing Alarm Log Correlation 15

Monitoring Alarms and Implementing Alarm Log Correlation 15

Prerequisites for Implementing Alarm Log Correlation 15
Information About Monitoring Alarms and Implementing Alarm Log Correlation 15
Displaying Router Alarms 15
Alarm Logging and Debugging Event Management System 17
Configuring Alarm Log Correlation 19
Configuring Logging Correlation Rules 19
Configuring a Logging Correlation Rule Set 20
Configuring Hierarchical Correlation Rule Flags 20
Configuring Logging Suppression Rules 21
Modifying Logging Events Buffer Settings 21
Modifying Logging Correlation Buffer Settings 21
Enabling Alarm Source Location Display Field for Bistate Alarms 22
Configuring SNMP Correlation Rules 22
Configuring SNMP Correlation Ruleset 23
Alarm Logging Correlation-Details 23

CHAPTER 4 **Onboard Failure Logging** 27
Prerequisites 27
Information About OBFL 28
Monitoring and Maintaining OBFL 29
Clearing OBFL Data 30

CHAPTER 5 **Implementing Performance Management** 31
Prerequisites for Implementing Performance Management 31
Information About Implementing Performance Management 32
PM Functional Overview 32
PM Statistics Server 32
PM Statistics Collector 32
PM Benefits 33
PM Statistics Collection Overview 33
How to Implement Performance Management 34
Configuring an External TFTP Server or Local Disk for PM Statistics Collection 34
Configuring PM Statistics Collection Templates 35
Configuring PM Threshold Monitoring Templates 36

Configuring Instance Filtering by Regular Expression 37
Performance Management: Details 37
Check System Health 49
Restrictions of the Health Check Feature 51
Monitoring Critical System Resources 51
Monitoring Infrastructure Services 54
Monitoring Counters 56

CHAPTER 6 **Configuring and Managing Embedded Event Manager Policies** 59
Prerequisites for Configuring and Managing Embedded Event Manager Policies 60
Information About Configuring and Managing Embedded Event Manager Policies 60
Event Management 60
System Event Processing 60
Embedded Event Manager Scripts 61
Regular Embedded Event Manager Scripts 61
Embedded Event Manager Policy Tcl Command Extension Categories 61
Cisco File Naming Convention for Embedded Event Manager 62
Embedded Event Manager Built-in Actions 63
Application-specific Embedded Event Management 64
Event Detection and Recovery 64
System Manager Event Detector 64
Timer Services Event Detector 65
Syslog Event Detector 66
None Event Detector 66
Distributed Event Detectors 66
Embedded Event Manager Event Scheduling and Notification 66
Reliability Statistics 67
How to Configure and Manage Embedded Event Manager Policies 68
Configuring Environmental Variables 68
Registering Embedded Event Manager Policies 69
How to Write Embedded Event Manager Policies Using Tcl 69
Registering and Defining an EEM Tcl Script 69
Displaying EEM Registered Policies 70
Unregistering EEM Policies 70

Suspending EEM Policy Execution 72
Specifying a Directory for Storing EEM Policies 72
Sample EEM Policies 72
Programming EEM Policies with Tcl 74
Creating an EEM User Tcl Library Index 78
Creating an EEM User Tcl Package Index 81
EEM Policies Using TCL: Details 83
Configuration Examples for Writing Embedded Event Manager Policies Using Tcl 86
EEM Sample Policy Descriptions 86
Registration of Some EEM Policies 86
Basic Configuration Details for All Sample Policies 87
Embedded Event Manager Policy Tcl Command Extension Reference 87
Embedded Event Manager Event Registration Tcl Command Extensions 88
event_register_appl 88
event_register_cli 89
event_register_config 90
event_register_none 91
event_register_oir 91
event_register_process 92
event_register_snmp_notification 93
event_register_syslog 94
event_register_timer 96
event_register_timer_subscriber 99
event_register_track 100
Embedded Event Manager Event Information Tcl Command Extension 101
event_reqinfo 101
Embedded Event Manager Action Tcl Command Extensions 115
action_process 115
action_program 116
action_script 117
action_setnode 118
action_syslog 118
Embedded Event Manager Utility Tcl Command Extensions 119
appl_read 119

appl_reqinfo 120
appl_setinfo 121
counter_modify 122
timer_arm 123
timer_cancel 125
Embedded Event Manager System Information Tcl Command Extensions 126
sys_reqinfo_cpu_all 126
sys_reqinfo_crash_history 127
sys_reqinfo_mem_all 128
sys_reqinfo_proc 129
sys_reqinfo_proc_all 131
sys_reqinfo_proc_version 131
sys_reqinfo_routername 131
sys_reqinfo_syslog_freq 132
sys_reqinfo_syslog_history 133
sys_reqinfo_stat 134
sys_reqinfo_snmp 135
SMTP Library Command Extensions 135
smtp_send_email 136
smtp_subst 137
CLI Library Command Extensions 138
cli_close 138
cli_exec 139
cli_get_ttyname 139
cli_open 139
cli_read 140
cli_read_drain 141
cli_read_line 141
cli_read_pattern 142
cli_write 142
Tcl Context Library Command Extensions 145
context_retrieve 146
context_save 149

Preface

Note 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 8000 Series Routers* preface contains these sections:

- Changes to this Document, on page ix
- Communications, Services, and Additional Information, on page ix

Changes to this Document

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

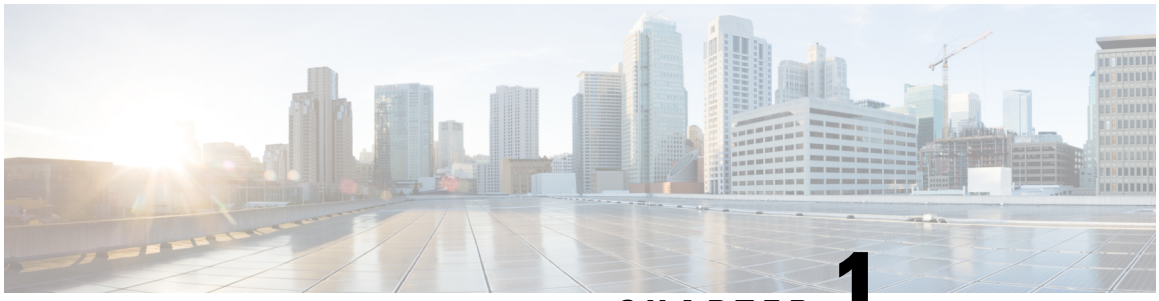
Date	Summary
March 2020	Initial release of this document.
August 2020	Republished for Release 7.0.14

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

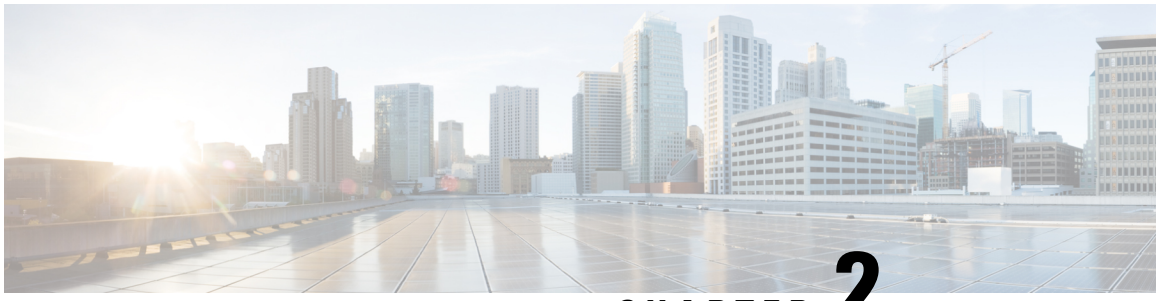
New and Changed System Monitoring Features

This chapter lists all the features that have been added or modified in this guide. The table also contains references to these feature documentation sections.

- System Monitoring Features Added or Modified in IOS XR Release 7.0.x, on page 1

System Monitoring Features Added or Modified in IOS XR Release 7.0.x

Feature	Description	Changed in Release	Where Documented
Health-check	This feature was enhanced in this release.	Release 7.0.14	Check System Health, on page 49



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 7.0.11	This feature was introduced.

• 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.
[pid]	Process ID (pid) of the process that generated the syslog message.
%*message -group -severity -message-code*	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 server 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 send 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 HundredGigE 0/0/0/0 (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 8000 Series Routers*

Related Topics

- Configuring Logging to the Logging Buffer, on page 5

Configuring Logging to Terminal Lines

By default syslog messages will be sent to the console terminal. But, syslog messages can also be send 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 submode.

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 the following issues:

- process crashes
- memory leaks
- CPU hogs
- tracebacks
- disk usage

PAM is enabled by default. 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, 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-cisco8000-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-cisco8000-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-cisco8000-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-cisco8000-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-cisco8000-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-cisco8000-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_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 pam status
Tue Jun 14 17:58:42.791 UTC
PAM is enabled
```

- From router shell prompt:

```
Router# run ps auxw|egrep perl

root     12559  0.0  0.0  57836 17992 ?        S    Apr24   0:00 /usr/bin/perl
/pkg/opt/cisco/pam//pam_plugin.pl
```

Disable PAM

To disable PAM agent systemwide, execute the following command from the XR EXEC mode:

```
Router# disable-pam
```

Re-enable PAM

To re-enable PAM agent systemwide, execute the following command from XR EXEC mode:

```
Router# enable-pam
```

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*.

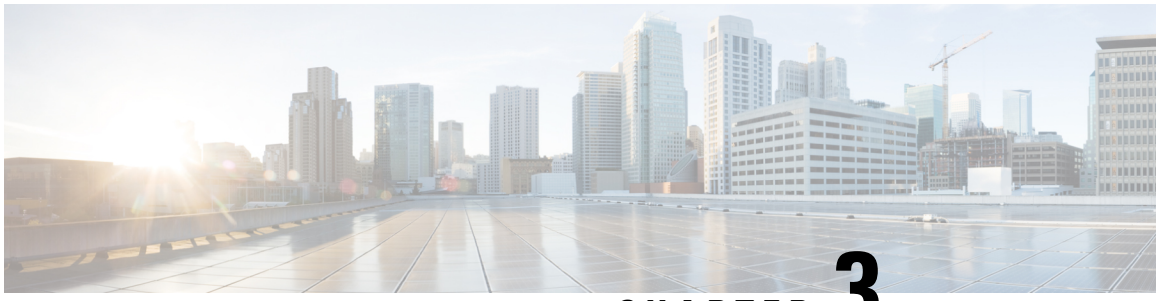
Files Collected by PAM Tool

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	• **show install active** • **show platform** • **show version** • core (gz) file • core.txt file
Process traceback	• **show dll** • **show install active** • **show logging** • **show platform** • **show version**
Memory leak	• **show install active** • **show platform** • **show version** • core (gz) file • dumpcore running • continuous memory usage snapshots
Show logging event	• **show install active** • **show logging** • **show platform** • **show version** • core (gz) file • core.txt file

Event Name	Commands and Files Collected by PAM
CPU hog	• **follow process** • **pstack** • **show dll** • **show install active** • **show platform** • **show version** • **top -H** • core (gz) file • CPU usage snapshots
Disk usage	• **show install active** • **show platform** • **show version** • console log • core (gz) file • Disk usage snapshots



CHAPTER 3

Monitoring Alarms and Implementing Alarm Log Correlation

This module describes the concepts and tasks related to monitoring or displaying router alarms, and 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.

- Monitoring Alarms and Implementing Alarm Log Correlation, on page 15

Monitoring Alarms and 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 monitoring and displaying router alarms, 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 Monitoring Alarms and Implementing Alarm Log Correlation

Displaying Router Alarms

You can view the router alarms in brief and detail.

Execute the command **show alarms brief** to view the router alarms in brief.

```
RP/0/RSP0/CPU0:router#show alarms brief

-------------------------------------------------------------------------
Active Alarms for 1/0
-------------------------------------------------------------------------
Location    Severity   Group     Set time              Description

-------------------------------------------------------------------------
```

```
0/1/CPU0   Critical   Fabric   11/11/2022 10:34:22 IST  LC Bandwidth Insufficient To Support
 Line Rate Traffic
1/0/CPU0    Major    Software  11/11/2022 10:43:36 IST   Optics1/0/0/20 - hw_optics:  RX
LOS LANE-0 ALARM
1/0/CPU0    Major    Software  11/11/2022 10:43:36 IST   Optics1/0/0/20 - hw_optics:  RX
LOS LANE-1 ALARM
------------------------------------------------------------------------------
History Alarms for 1/0
------------------------------------------------------------------------------
No entries.

------------------------------------------------------------------------------
Suppressed Alarms for 1/0
------------------------------------------------------------------------------
No entries.

------------------------------------------------------------------------------
Conditions for 1/0
------------------------------------------------------------------------------
No entries.
```

Execute the command **show alarms detail** to view the router alarms in detail.

```
RP/0/RSP0/CPU0:ddc2-uut#show alarms detail

-------------------------------------------------------
Active Alarms for 1/0
-------------------------------------------------------
Description:            LC Bandwidth Insufficient To Support Line Rate Traffic


Location:               1/0/CPU0

AID:                    XR_FABRIC/SW_MISC_ERR/18

Tag String:             FAM_FAULT_TAG_HW_FIA_LC_BANDWIDTH

Module Name:            N/A

EID:                    MODULE/MSC/1:MODULE/SLICE/1:MODULE/PSE/1

Reporting Agent ID:     524365
Pending Sync:           false
Severity:               Critical
Status:                 Set
Group:                  Fabric
Set Time:               11/16/2022 20:44:44 IST
Clear Time:             -
Service Affecting:      NotServiceAffecting
Transport Direction:    NotSpecified
Transport Source:       NotSpecified
Interface:              N/A

Alarm Name:             LC-BW-DEG
-------------------------------------------------------

History Alarms for 1/0
-------------------------------------------------------
No entries.

-------------------------------------------------------
Suppressed Alarms for 1/0
-------------------------------------------------------
No entries.
```

```
------------------------------------------------------------
Conditions for 1/0
------------------------------------------------------------
No entries.

------------------------------------------------------------
Clients for 1/0
------------------------------------------------------------
Agent Name:              optics_fm.xml

Agent ID:                196678
Agent Location:          1/0/CPU0

Agent Handle:            93827323237168

Agent State:             Registered
Agent Type:              Producer
Agent Filter Display:    false
Agent Subscriber ID:     0
Agent Filter Severity:   Unknown
Agent Filter State:      Unknown
Agent Filter Group:      Unknown
Agent Connect Count:     1
Agent Connect Timestamp: 11/16/2022 20:40:18 IST
Agent Get Count:         0
Agent Subscribe Count:   0
Agent Report Count:      8
------------------------------------------------------------
Statistics for 1/0
------------------------------------------------------------
Alarms Reported:                9
Alarms Dropped:                 0
Active (bi-state set):          9
History (bi-state cleared):     0
Suppressed:                     0
Dropped Invalid AID:            0
Dropped No Memory:              0
Dropped DB Error:               0
Dropped Clear Without Set:      0
Dropped Duplicate:              0
Cache Hit:                      0
Cache Miss:                     0
```

Alarm Logging and Debugging Event Management System

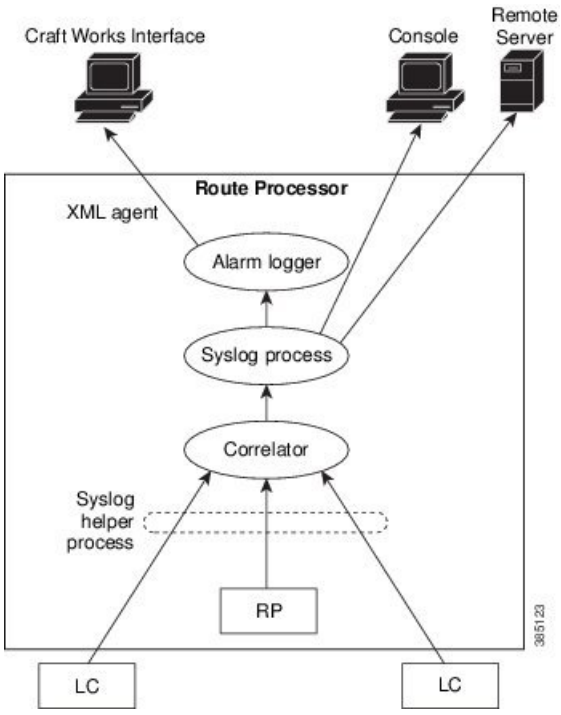
Cisco IOS XR Software Alarm Logging and Debugging Event Management System (ALDEMS) is used to monitor and store alarm messages that are forwarded by system servers and applications. In addition, ALDEMS correlates alarm messages forwarded due to a single root cause.

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.

Figure 1: ALDEMS Component Communications, on page 18 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

By default, the router sends system logging messages to a system logging (syslog) process. Syslog helper processes, that are distributed across the nodes of the router, gather the syslog messages. The system logging process controls the distribution of logging messages to the various destinations, such as the system logging buffer, the console, terminal lines, or a syslog server, depending on the network device configuration.

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 (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.

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 rule1 type stateful
Router(config-corr-rule-st)# timeout 60000
Router(config-corr-rule-st)# exit
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 rule1 type stateful
Router(config-corr-rule-st)# rootcause CAT_BI_1 GROUP_BI_1 CODE_BI_1
Router(config-corr-rule-st)# nonrootcause
Router(config-corr-rule-st-nonrc)# alarm CAT_BI_2 GROUP_BI_2 CODE_BI_2
Router(config)# commit
```

Applying a Logging Correlation Rule

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.

Configuration Example

This example shows how to apply a logging correlation rule.

```
Router# configure
Router(config)# logging correlator apply rule rule1
Router(config-corr-apply-rule)# all-of-router
or
Router(config-corr-apply-rule)# location 0/1/CPU0
or
Router(config-corr-apply-rule)# context HundredGigE_0_0_0_0
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. To configure a ruleset, you should first configure a rule with the same name. The logging correlation ruleset can be applied to the entire router or to a specific context or location.

```
Router# configure
Router(config)# logging correlator ruleset rule1
Router(config-corr-ruleset)# rulename rule1
Router(config-corr-ruleset)# exit
Router(config)# logging correlator apply ruleset rule1
Router(config-corr-apply-rule)# all-of-router
or
Router(config-corr-apply-rule)# location 0/2/CPU0
or
Router(config-corr-apply-rule)# context HundredGigE_0_0_0_0
Router(config)# commit
```

Configuring Hierarchical Correlation Rule Flags

Hierarchical correlation is when a single alarm is both a root cause for one correlation rule and a non-root cause for another rule, and when alarms are generated resulting in a successful correlation associated with both rules. What happens to a non-root-cause alarm depends on the behavior of its correlated root-cause alarm. There are cases in which you want to control the stateful behavior associated with these hierarchies and to implement flags, such as reparenting and reissuing of non-bistate alarms. For detailed information about hierarchical correlation and correlation flags, see Hierarchical Correlation, on page 25

Configuration Example

This example shows how to configure hierarchical correlation rule flags.

```
Router# configure
Router(config)# logging correlator rule rule_stateful type stateful
Router(config-corr-rule-st)# reissue-nonbistate
Router(config-corr-rule-st)# reparent
Router(config-corr-rule-st)# commit
Router(config-corr-rule-st)# exit
Router(config)# exit
Router# show logging correlator rule all (optional)
```

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-suppr-rule)# exit
Router(config)# logging suppress apply rule infobistate
Router(config-suppr-apply-rule)# 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)# commit
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 24

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 1024 bytes. The default value for buffer size is 64KB.

```
Router# configure terminal
Router(config)# snmp-server correlator buffer-size 1024 (optional)
Router(config)# snmp-server correlator rule rule1
Router(config-corr-rule-nonst)# timeout 100
Router(config-corr-rule-nonst)# rootcause 1.3.6.1.2.1.47.1.1
Router(config-corr-rule-nonst-rootvb)# varbind 1.3.6.1.2.1.47.1.2 index regex .*
Router(config-corr-rule-nonst-rootvb)# varbind 1.3.6.1.2.1.47.1.2 value regex .*
Router(config-corr-rule-nonst-rootvb)# exit
Router(config-corr-rule-nonst)# nonrootcause
Router(config-corr-rule-nonst-nonrc)# trap 1.3.6.1.2.1.47.1.1
Router(config-corr-rule-nonst-nonrcvb)# varbind 1.3.6.1.2.1.47.1.3 index regex .*
Router(config-corr-rule-nonst-nonrcvb)# varbind 1.3.6.1.2.1.47.1.3 value regex .*
Router(config-corr-rule-nonst-nonrcvb)# exit
Router(config-corr-rule-nonst-nonrc)# trap 1.3.6.1.2.1.47.1.2
Router(config-corr-rule-nonst-nonrcvb)# varbind 1.3.6.1.2.1.47.1.4 index regex .*
Router(config-corr-rule-nonst-nonrcvb)# varbind 1.3.6.1.2.1.47.1.4 value regex .*
Router(config-corr-rule-nonst-nonrcvb)# exit
Router(config-corr-rule-nonst-nonrc)# exit
Router(config-corr-rule-nonst)# exit
```

```
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.

```
Router# configure terminal
Router(config)# snmp-server correlator ruleset rule1 rulename rule1
Router(config)# snmp-server correlator apply ruleset rule1 host ipv4 address 1.2.3.4
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 (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 HundredGigE 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 HundredGigE 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 HundredGigE 0/0/0/0: Line protocol on Interface HundredGigE 0/0/0/0,
 changed state to Down
```

Hierarchical Correlation

Hierarchical correlation takes effect when the following conditions are true:

- When a single alarm is both a root cause for one rule and a non-root cause for another rule.
- When alarms are generated that result in successful correlations associated with both rules.

The following example illustrates two hierarchical correlation rules:

Rule 1	**Category**	**Group**	**Code**
Root Cause 1	Cat 1	Group 1	Code 1
Non-root Cause 2	Cat 2	Group 2	Code 2
Rule 2			
Root Cause 2	Cat 2	Group 2	Code 2
Non-root Cause 3	Cat 3	Group 3	Code 3

If three alarms are generated for Cause 1, 2, and 3, with all alarms arriving within their respective correlation timeout periods, then the hierarchical correlation appears like this:

Cause 1 -> Cause 2 -> Cause 3

The correlation buffers show two separate correlations: one for Cause 1 and Cause 2 and the second for Cause 2 and Cause 3. However, the hierarchical relationship is implicitly defined.

Note Stateful behavior, such as reparenting and reissuing of alarms, is supported for rules that are defined as stateful; that is, correlations that can change.

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.

Reparent Flag

The reparent flag specifies what happens to non-root-cause alarms in a hierarchical correlation when their immediate root cause clears.

The following example illustrates 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.

In this scenario, if alarm A arrives generated from context 1 and alarm B generated from context 2, then a correlation occurs—regardless of context.

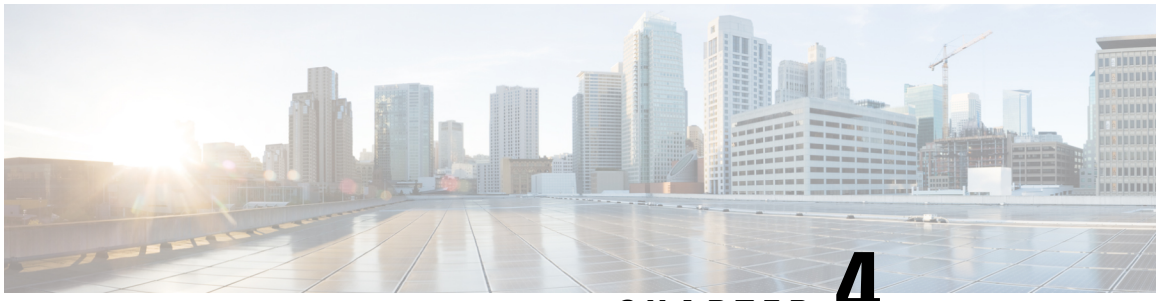
If the context correlation flag is now set on Rule 1 and the same alarms are generated, they are not correlated, because they are from different contexts.

Active-standby consistency check

In a dual RP system, interface state sync library monitors the state between active and standby RPs. An alarm is raised by an RP when its respective port is down and the same port is up for other RP.

Alarm Clearance

The alarm is cleared when there is no inconsistency between both the RP port state.



CHAPTER 4

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.

Note OBFL is activated by default in all cards and cannot be disabled.

Feature History for Implementing OBFL

Release	Modification
Release 7.0.11	This feature was introduced.

- Prerequisites , on page 27
- Information About OBFL, on page 28
- Monitoring and Maintaining OBFL, on page 29
- Clearing OBFL Data, on page 30

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 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:

- Alarms
- Boot time
- Field Programmable Device (FPD) Upgrade data
- FRU part serial number
- Temperature and voltage at boot
- Temperature and voltage history
- Total run time

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 for Baseline Data Collection

Data Type	Details
Current	Information from the current sensors is recorded after boot. The subsequent recordings are specific to variations based on preset thresholds.
Installation	Chassis serial number and slot number are stored at initial boot.
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 15 minutes.
Temperature	Information from the temperature sensors is recorded after boot. The subsequent recordings are specific to variations based on preset thresholds.
Voltage	Information from the voltage sensors is recorded after boot. The subsequent recordings are specific to variations based on preset thresholds.

Event-Driven Data Collection

Event driven data include card failure events. Failure events are card crashes, memory errors, ASIC resets, and similar hardware failure indications.

Table 7: Data Types for Event-Driven Data Collection

Data Type	Details
Alarm	Major and critical alarm state changes.

Data Type	Details
Current	Current sensor information.
FPD	FPD upgrade information.
Inventory	IDPROM information and card state changes.
Temperature	Inlet and hot point temperature value changes beyond the thresholds set in the hardware inventory XML files.
Uptime	Card uptime and location history, including the most recent time the card OBFL disk was cleared.
Voltage	Voltage value changes beyond the thresholds set in the hardware inventory XML files.

Supported Cards and Platform

FRUs that have sufficient nonvolatile memory available for OBFL data storage support OBFL. The following table provides information about the OBFL support for different FRUs on the Cisco 8000 Series router.

Table 8: OBFL Support on Cisco 8000 Series Router

Card Type	Cisco 8000 Series Router
Route processor	Supported
Fabric cards	Supported
Line card	Supported
Power supply cards	Not Supported
Fan tray	Not Supported

Monitoring and Maintaining OBFL

Use the commands described in this section to display the status of OBFL, and the data collected by OBFL. Enter these commands in EXEC mode.

Procedure

	Command or Action	Purpose
Step 1	**show logging onboard** {**alarm** | **current** | **fpd** | **inventory** | **temperature** | **uptime** | **voltage** } [**location** *node-id*] **Example:** `Router# show logging onboard uptime`	Displays stored OBFL data for all nodes or for a specified node. See the *Onboard Failure Logging Commands* module in the *System Monitoring Command Reference for Cisco 8000 Series Routers*.

	Command or Action	Purpose
Step 2	**show process** | **include obfl** **Example:** `Router# show process | include obfl`	Confirms that the OBFL environmental monitor process is operating.
Step 3	**show process obflmgr** **Example:** `Router# show process obflmgr`	Displays details about the OBFL manager process.

Clearing OBFL Data

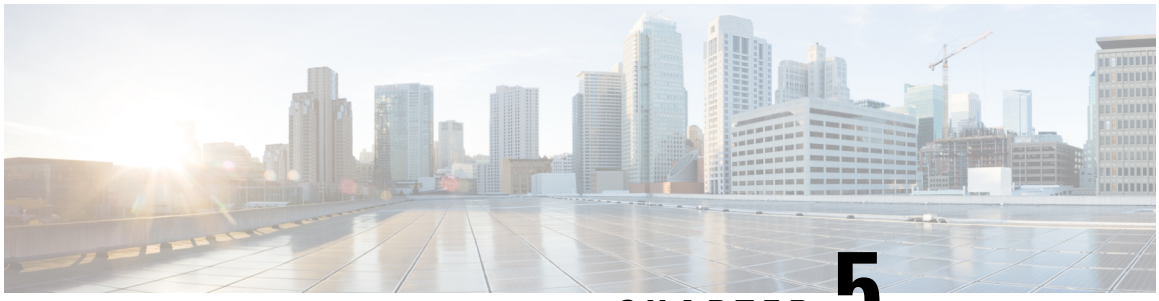
To erase all OBFL data on a specific card, use the following command:

clear logging onboard [**location** *node-id*]

Caution The **clear logging onboard** command permanently deletes all OBFL data for a node. Do not clear the OBFL logs without specific reasons because the OBFL data is used to diagnose and resolve problems in FRUs.

Note The obflmgr process automatically removes old log files to make room for new log files as needed. No manual intervention is required in order to free up OBFL disk space.

For more information, see the *Onboard Failure Logging Commands* module in the *System Monitoring Command Reference for Cisco 8000 Series Routers*.



CHAPTER 5

Implementing Performance Management

Performance management (PM) on the Cisco IOS XR Software provides a framework to perform these tasks:

- Collect and export PM statistics to a TFTP server for data storage and retrieval
- Monitor the system using extensible markup language (XML) queries
- Configure threshold conditions that generate system logging messages when a threshold condition is matched.

The PM system collects data that is useful for graphing or charting system resource utilization, for capacity planning, for traffic engineering, and for trend analysis.

- Prerequisites for Implementing Performance Management , on page 31
- Information About Implementing Performance Management, on page 32
- PM Functional Overview, on page 32
- PM Benefits, on page 33
- PM Statistics Collection Overview, on page 33
- How to Implement Performance Management, on page 34
- Check System Health, on page 49

Prerequisites for Implementing Performance Management

Before implementing performance management in your network operations center (NOC), ensure that these prerequisites are met:

- 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 a TFTP server.

Information About Implementing Performance Management

PM Functional Overview

The Performance Management (PM) frameworks consists of two major components:

- PM statistics server
- PM statistics collectors

PM Statistics Server

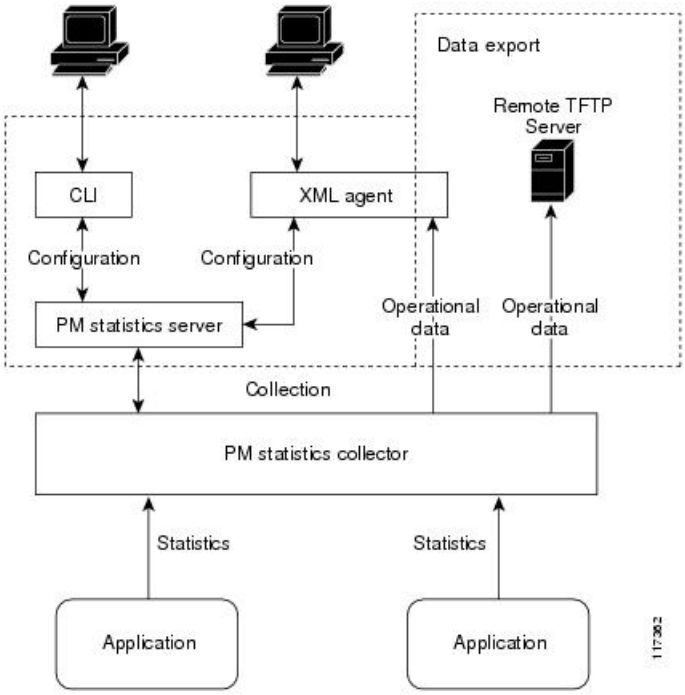
The PM statistics server is the front end for statistic collections, entity instance monitoring collections, and threshold monitoring. All PM statistic collections and threshold conditions configured through the command-line interface (CLI) or through XML schemas are processed by the PM statistics server and distributed among the PM statistics collectors.

PM Statistics Collector

The PM statistics collector collects statistics from entity instances and stores that data in memory. The memory contents are checkpointed so that information is available across process restarts. In addition, the PM statistics collector is responsible for exporting operational data to the XML agent and to the TFTP server.

Figure 2: PM Component Communications, on page 33 illustrates the relationship between the components that constitute the PM system.

Figure 2: PM Component Communications



PM Benefits

The PM system provides these benefits:

- Configurable data collection policies
- Efficient transfer of statistical data in the binary format via TFTP
- Entity instance monitoring support
- Threshold monitoring support
- Data persistency across process restarts and processor failovers

PM Statistics Collection Overview

A PM statistics collection first gathers statistics from all the attributes associated with all the instances of an entity in the PM system. It then exports the statistical data in the binary file format to a TFTP server. For example, a Multiprotocol Label Switching (MPLS) Label Distribution Protocol (LDP) statistics collection gathers statistical data from all the attributes associated with all MPLS LDP sessions on the router.

This table lists the entities and the associated instances in the PM system.

Table 9: Entity Classes and Associated Instances

Entity Classes	Instance
BGP	Neighbors or Peers
Interface Basic Counters	Interfaces
Interface Data Rates	Interfaces
Interface Generic Counters	Interfaces
MPLS LDP	LDP Sessions
Node CPU	Nodes
Node Memory	Nodes
Node Process	Processes
OSPFv2	Processes
OSPFv3	Processes

Note For a list of all attributes associated with the entities that constitute the PM system, see Table 10: Attributes and Values, on page 38.

Note Based on the interface type, the interface either supports the interface generic counters or the interface basic counters. The interfaces that support the interface basic counters do not support the interface data rates.

How to Implement Performance Management

Configuring an External TFTP Server or Local Disk for PM Statistics Collection

You can export PM statistical data to an external TFTP server or dump the data to the local file system. Both the local and TFTP destinations are mutually exclusive and you can configure either one of them at a time.

Configuration Examples

This example configures an external TFTP server for PM statistics collection.

```
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# performance-mgmt resources tftp-server 10.3.40.161 directory
 mypmdata/datafiles
RP/0/RP0/CPU0:Router(config)# commit
```

This example configures a local disk for PM statistics collection.

```
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# performance-mgmt resources dump local
RP/0/RP0/CPU0:Router(config)# commit
```

Configuring PM Statistics Collection Templates

PM statistics collections are configured through PM statistics collection templates. A PM statistics collection template contains the entity, the sample interval, and the number of sampling operations to be performed before exporting the data to a TFTP server. When a PM statistics collection template is enabled, the PM statistics collection gathers statistics for all attributes from all instances associated with the entity configured in the template. You can define multiple templates for any given entity; however, only one PM statistics collection template for a given entity can be enabled at a time.

Guidelines for Configuring PM Statistics Collection Templates

When creating PM statistics collection templates, follow these guidelines:

- You must configure a TFTP server resource or local dump resource if you want to export statistics data onto a remote TFTP server or local disk.
- You can define multiple templates for any given entity, but at a time you can enable only one PM statistics collection template for a given entity.
- When configuring a template, you can designate the template for the entity as the default template using the default keyword or name the template. The default template contains the following default values:
  - A sample interval of 10 minutes.
  - A sample size of five sampling operations.
- The sample interval sets the frequency of the sampling operations performed during the sampling cycle. You can configure the sample interval with the sample-interval command. The range is from 1 to 60 minutes.
- The sample size sets the number of sampling operations to be performed before exporting the data to the TFTP server. You can configure the sample size with the **sample-size** command. The range is from 1 to 60 samples.

Note Specifying a small sample interval increases CPU utilization, whereas specifying a large sample size increases memory utilization. The sample size and sample interval, therefore, may need to be adjusted to prevent system overload.

- The export cycle determines how often PM statistics collection data is exported to the TFTP server. The export cycle can be calculated by multiplying the sample interval and sample size (sample interval x sample size = export cycle).
- Once a template has been enabled, the sampling and export cycles continue until the template is disabled with the no form of the **performance-mgmt apply statistics** command.
- You must specify either a node with the **location** command or enable the PM statistic collections for all nodes using the **location all** command when enabling or disabling a PM statistic collections for the following entities:
  - Node CPU

- Node memory
- Node process

Configuration Example

This example shows how to create and enable a PM statistics collection template.

```
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# performance-mgmt statistics interface generic-counters template
 template 1
RP/0/RP0/CPU0:Router(config)# performance-mgmt statistics interface generic-counters template
 1 sample-size 10
RP/0/RP0/CPU0:Router(config)# performance-mgmt statistics interface generic-counters template
 1 sample-interval 5
RP/0/RP0/CPU0:Router(config)# performance-mgmt apply statistics interface generic-counters
 1
RP/0/RP0/CPU0:Router# commit
```

Configuring PM Threshold Monitoring Templates

The PM system supports the configuration of threshold conditions to monitor an attribute (or attributes) for threshold violations. Threshold conditions are configured through PM threshold monitoring templates. When a PM threshold template is enabled, the PM system monitors all instances of the attribute (or attributes) for the threshold condition configured in the template. If at end of the sample interval a threshold condition is matched, the PM system generates a system logging message for each instance that matches the threshold condition. For the list of attributes and value ranges associated with each attribute for all the entities, see Performance Management: Details, on page 37

Guidelines for Configuring PM Threshold Monitoring Templates

While you configure PM threshold monitoring templates, follow these guidelines:

- Once a template has been enabled, the threshold monitoring continues until the template is disabled with the **no** form of the **performance-mgmt apply thresholds** command.
- Only one PM threshold template for an entity can be enabled at a time.
- You must specify either a node with the **location** command or enable the PM statistic collections for all nodes using the **location all** command when enabling or disabling a PM threshold monitoring template for the following entities:
  - Node CPU
  - Node memory
  - Node process

Configuration Example

This example shows how to create and enable a PM threshold monitoring template. In this example, a PM threshold template is created for the **CurrMemory** attribute of the **node memory** entity. The threshold condition in this PM threshold condition monitors the **CurrMemory** attribute to determine whether the current memory use is greater than 50 percent.

```
Router# conf t
Router(config)# performance-mgmt thresholds node memory template template20
Router(config-threshold-cpu)# CurrMemory gt 50 percent
```

```
Router(config-threshold-cpu)# sample-interval 5
Router(config-threshold-cpu)# exit
Router(config)# performance-mgmt apply thresholds node memory location 0/RP0/CPU0 template20
Router(config)# commit
```

Configuring Instance Filtering by Regular Expression

This task explains defining a regular expression group which can be applied to one or more statistics or threshold templates. You can also include multiple regular expression indices. The benefits of instance filtering using the regular expression group is as follows.

- You can use the same regular expression group that can be applied to multiple templates.
- You can enhance flexibility by assigning the same index values.
- You can enhance the performance by applying regular expressions, which has OR conditions.

Note The Instance filtering by regular-expression is currently supported in interface entities only (Interface basic-counters, generic-counters, data-rates.

Configuration Example

This example shows how to define a regular expression group.

```
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# performance-mgmt regular-expression regexp
RP/0/RP0/CPU0:Router(config-perfmgmt-regex)# index 10 match
RP/0/RP0/CPU0:Router(config)# commit
```

Performance Management: Details

This section contains additional information which will be useful while configuring performance management.

This table describes the attributes and value ranges associated with each attribute for all the entities that constitute the PM system.

Table 10: Attributes and Values

Entity	Attributes	Description	Values
bgp	ConnDropped	Number of times the connection was dropped.	Range is from 0 to 4294967295.
	ConnEstablished	Number of times the connection was established.	Range is from 0 to 4294967295.
	ErrorsReceived	Number of error notifications received on the connection.	Range is from 0 to 4294967295.
	ErrorsSent	Number of error notifications sent on the connection.	Range is from 0 to 4294967295.
	InputMessages	Number of messages received.	Range is from 0 to 4294967295.
	InputUpdateMessages	Number of update messages received.	Range is from 0 to 4294967295.
	OutputMessages	Number of messages sent.	Range is from 0 to 4294967295.
	OutputUpdateMessages	Number of update messages sent.	Range is from 0 to 4294967295.
interface data-rates	Bandwidth	Bandwidth in kbps.	Range is from 0 to 4294967295.
	InputDataRate	Input data rate in kbps.	Range is from 0 to 4294967295.
	InputPacketRate	Input packets per second.	Range is from 0 to 4294967295.
	InputPeakRate	Peak input data rate.	Range is from 0 to 4294967295.
	InputPeakPkts	Peak input packet rate.	Range is from 0 to 4294967295.
	OutputDataRate	Output data rate in kbps.	Range is from 0 to 4294967295.
	OutputPacketRate	Output packets per second.	Range is from 0 to 4294967295.
	OutputPeakPkts	Peak output packet rate.	Range is from 0 to 4294967295.
	OutputPeakRate	Peak output data rate.	Range is from 0 to 4294967295.

Entity	Attributes	Description	Values
interface basic-counters	InPackets	Packets received.	Range is from 0 to 4294967295.
	InOctets	Bytes received.	Range is from 0 to 4294967295.
	OutPackets	Packets sent.	Range is from 0 to 4294967295.
	OutOctets	Bytes sent.	Range is from 0 to 4294967295.
	InputTotalDrops	Inbound correct packets discarded.	Range is from 0 to 4294967295.
	InputQueueDrops	Input queue drops.	Range is from 0 to 4294967295.
	InputTotalErrors	Inbound incorrect packets discarded.	Range is from 0 to 4294967295.
	OutputTotalDrops	Outbound correct packets discarded.	Range is from 0 to 4294967295.
	OutputQueueDrops	Output queue drops.	Range is from 0 to 4294967295.
	OutputTotalErrors	Outbound incorrect packets discarded.	Range is from 0 to 4294967295.

Entity	Attributes	Description	Values
interface generic-counters	InBroadcastPkts	Broadcast packets received.	Range is from 0 to 4294967295.
	InMulticastPkts	Multicast packets received.	Range is from 0 to 4294967295.
	InOctets	Bytes received.	Range is from 0 to 4294967295.
	InPackets	Packets received.	Range is from 0 to 4294967295.
	InputCRC	Inbound packets discarded with incorrect CRC.	Range is from 0 to 4294967295.
	InputFrame	Inbound framing errors.	Range is from 0 to 4294967295.
	InputOverrun	Input overruns.	Range is from 0 to 4294967295.
	InputQueueDrops	Input queue drops.	Range is from 0 to 4294967295.
	InputTotalDrops	Inbound correct packets discarded.	Range is from 0 to 4294967295.
	InputTotalErrors	Inbound incorrect packets discarded.	Range is from 0 to 4294967295.
	InUcastPkts	Unicast packets received.	Range is from 0 to 4294967295.
	InputUnknownProto	Inbound packets discarded with unknown protocol.	Range is from 0 to 4294967295.
	OutBroadcastPkts	Broadcast packets sent.	Range is from 0 to 4294967295.
	OutMulticastPkts	Multicast packets sent.	Range is from 0 to 4294967295.
	OutOctets	Bytes sent.	Range is from 0 to 4294967295.
	OutPackets	Packets sent.	Range is from 0 to 4294967295.
	OutputTotalDrops	Outbound correct packets discarded.	Range is from 0 to 4294967295.
	OutputTotalErrors	Outbound incorrect packets discarded.	Range is from 0 to 4294967295.
	OutUcastPkts	Unicast packets sent.	Range is from 0 to 4294967295.
	OutputUnderrun	Output underruns.	Range is from 0 to 4294967295.

Entity	Attributes	Description	Values
mpls ldp	AddressMsgsRcvd	Address messages received.	Range is from 0 to 4294967295.
	AddressMsgsSent	Address messages sent.	Range is from 0 to 4294967295.
	AddressWithdrawMsgsRcd	Address withdraw messages received.	Range is from 0 to 4294967295.
	AddressWithdrawMsgsSent	Address withdraw messages sent.	Range is from 0 to 4294967295.
	InitMsgsSent	Initial messages sent.	Range is from 0 to 4294967295.
	InitMsgsRcvd	Initial messages received.	Range is from 0 to 4294967295.
	KeepaliveMsgsRcvd	Keepalive messages received.	Range is from 0 to 4294967295.
	KeepaliveMsgsSent	Keepalive messages sent.	Range is from 0 to 4294967295.
	LabelMappingMsgsRcvd	Label mapping messages received.	Range is from 0 to 4294967295.
	LabelMappingMsgsSent	Label mapping messages sent.	Range is from 0 to 4294967295.
	LabelReleaseMsgsRcvd	Label release messages received.	Range is from 0 to 4294967295.
	LabelReleaseMsgsSent	Label release messages sent.	Range is from 0 to 4294967295.
	LabelWithdrawMsgsRcvd	Label withdraw messages received.	Range is from 0 to 4294967295.
	LabelWithdrawMsgsSent	Label withdraw messages sent.	Range is from 0 to 4294967295.
	NotificationMsgsRcvd	Notification messages received.	Range is from 0 to 4294967295.
	NotificationMsgsSent	Notification messages sent.	Range is from 0 to 4294967295.
	TotalMsgsRcvd	Total messages received.	Range is from 0 to 4294967295.
	TotalMsgsSent	Total messages sent.	Range is from 0 to 4294967295.
node cpu	NoProcesses	Number of processes.	Range is from 0 to 4294967295.

Entity	Attributes	Description	Values
node memory	CurrMemory	Current application memory (in bytes) in use.	Range is from 0 to 4294967295.
	PeakMemory	Maximum system memory (in MB) used since bootup.	Range is from 0 to 4194304.
node process	NoThreads	Number of threads.	Range is from 0 to 4294967295.
	PeakMemory	Maximum dynamic memory (in KB) used since startup time.	Range is from 0 to 4194304.

Entity	Attributes	Description	Values
ospf v2protocol	InputPackets	Total number of packets received.	Range is from 0 to 4294967295.
	OutputPackets	Total number of packets sent.	Range is from 0 to 4294967295.
	InputHelloPackets	Number of Hello packets received.	Range is from 0 to 4294967295.
	OutputHelloPackets	Number of Hello packets sent.	Range is from 0 to 4294967295.
	InputDBDs	Number of DBD packets received.	Range is from 0 to 4294967295.
	InputDBDsLSA	Number of LSA received in DBD packets.	Range is from 0 to 4294967295.
	OutputDBDs	Number of DBD packets sent.	Range is from 0 to 4294967295.
	OutputDBDsLSA	Number of LSA sent in DBD packets.	Range is from 0 to 4294967295.
	InputLSRequests	Number of LS requests received.	Range is from 0 to 4294967295.
	InputLSRequestsLSA	Number of LSA received in LS requests.	Range is from 0 to 4294967295.
	OutputLSRequests	Number of LS requests sent.	Range is from 0 to 4294967295.
	OutputLSRequestsLSA	Number of LSA sent in LS requests.	Range is from 0 to 4294967295.
	InputLSAUpdates	Number of LSA updates received.	Range is from 0 to 4294967295.
	InputLSAUpdatesLSA	Number of LSA received in LSA updates.	Range is from 0 to 4294967295.
	OutputLSAUpdates	Number of LSA updates sent.	Range is from 0 to 4294967295.
	OutputLSAUpdatesLSA	Number of LSA sent in LSA updates.	Range is from 0 to 4294967295.
	InputLSAAcks	Number of LSA acknowledgements received.	Range is from 0 to 4294967295.

Entity	Attributes	Description	Values
	InputLSAAcksLSA	Number of LSA received in LSA acknowledgements.	Range is from 0 to 4294967295.
	OutputLSAAcks	Number of LSA acknowledgements sent	Range is from 0 to 4294967295.
	OutputLSAAcksLSA	Number of LSA sent in LSA acknowledgements.	Range is from 0 to 4294967295.
	ChecksumErrors	Number of packets received with checksum errors.	Range is from 0 to 4294967295.

Entity	Attributes	Description	Values
ospf v3protocol	InputPackets	Total number of packets received.	Range is from 0 to 4294967295.
	OutputPackets	Total number of packets sent.	Range is from 0 to 4294967295.
	InputHelloPackets	Number of Hello packets received.	Range is from 0 to 4294967295.
	OutputHelloPackets	Number of Hello packets sent.	Range is from 0 to 4294967295.
	InputDBDs	Number of DBD packets received.	Range is from 0 to 4294967295.
	InputDBDsLSA	Number of LSA received in DBD packets.	Range is from 0 to 4294967295.
	OutputDBDs	Number of DBD packets sent.	Range is from 0 to 4294967295.
	OutputDBDsLSA	Number of LSA sent in DBD packets.	Range is from 0 to 4294967295.
	InputLSRequests	Number of LS requests received.	Range is from 0 to 4294967295.
	InputLSRequestsLSA	Number of LSA received in LS requests.	Range is from 0 to 4294967295.
	OutputLSRequests	Number of LS requests sent.	Range is from 0 to 4294967295.
	OutputLSRequestsLSA	Number of LSA sent in LS requests.	Range is from 0 to 4294967295.
	InputLSAUpdates	Number of LSA updates received.	Range is from 0 to 4294967295.
	InputLSRequestsLSA	Number of LSA received in LS requests.	Range is from 0 to 4294967295.
	OutputLSAUpdates	Number of LSA updates sent.	Range is from 0 to 4294967295.
	OutputLSAUpdatesLSA	Number of LSA sent in LSA updates.	Range is from 0 to 4294967295.
	InputLSAAcks	Number of LSA acknowledgements received.	Range is from 0 to 4294967295.

Entity	Attributes	Description	Values
	InputLSAAcksLSA	Number of LSA received in LSA acknowledgements.	Range is from 0 to 4294967295.
	OutputLSAAcks	Number of LSA acknowledgements sent	Range is from 0 to 4294967295.
	OutputLSAAcksLSA	Number of LSA sent in LSA acknowledgements.	Range is from 0 to 4294967295.

This table describes the commands used to enable entity instance monitoring for different entity instances.

Table 11: Entity Instances and Monitoring Commands

Entity	Command Description
BGP	Use the **performance-mgmt apply monitor bgp** command to enable entity instance monitoring for a BGP entity instance. **Syntax:** **performance-mgmt** **apply monitor** **bgp** *ip-address* *template-name* | **default**} `RP/0/RP0/CPU0:Router(config)# performance-mgmt apply monitor bgp 10.12.0.4 default`
Interface Data Rates	Use the **performance-mgmt apply monitor data-rates** command to enable entity instance monitoring for an interface data rates entity instance. **Syntax:** **performance-mgmt** **apply** **monitor** **interface** **data-rates** *type* *interface-path-id* {*template-name* | **default**} `RP/0/RP0/CPU0:Router(config)# performance-mgmt apply monitor interface data-rates HundredGigE 0/0/0/0 default`

Entity	Command Description

Interface Basic Counters	Use the **performance-mgmt apply monitor interface basic-counters** command to enable entity instance monitoring for an interface basic counters entity instance.
	Syntax:
	performance-mgmt
	apply
	monitor
	interface
	basic-counters
	type
	interface-path-id {*template-name* |
	default}
	`RP/0/RP0/CPU0:Router(config)# performance-mgmt apply monitor interface basic-counters`
	`HundredGigE 0/0/0/0 default`

Interface Generic Counters	Use the **performance-mgmt apply monitor interface generic-counters** command to enable entity instance monitoring for an interface generic counters entity instance.
	Syntax:
	performance-mgmt
	apply
	monitor
	interface
	generic-counters
	type
	interface-path-id {*template-name* |
	default}
	`RP/0/RP0/CPU0:Router(config)# performance-mgmt apply monitor interface generic-counters`
	`HundredGigE 0/0/0/0 default`

MPLS LDP	Use the **performance-mgmt apply monitor mpls ldp** command to enable entity instance monitoring for an MPLS LDP entity instance. **Syntax:** **performance-mgmt** **apply monitor** **mpls** **ldp** *ip-address* {*template-name* | **default**} `RP/0/RP0/CPU0:Router(config)# performance-mgmt apply monitor mpls ldp 10.34.64.154 default`

Entity	Command Description
Node CPU	Use the **performance-mgmt apply monitor node cpu** command to enable entity instance monitoring for a node CPU entity instance. **Syntax:** **performance-mgmt** **apply** **monitor** **node** **cpu** **location** *node-id* {*template-name* | **default**} `RP/0/RP0/CPU0:Router(config)# performance-mgmt apply monitor node cpu location 0/RP0/CPU0 default`
Node Memory	Use the **performance-mgmt apply monitor node memory** command to enable entity instance monitoring for a node memory entity instance. **Syntax:** **performance-mgmt** **apply** **monitor** **node** **memory** **location** *node-id* {*template-name* | **default**} `RP/0/RP0/CPU0:Router(config)# performance-mgmt apply monitor node memory location 0/RP0/CPU0 default`
Node Process	Use the **performance-mgmt apply monitor node process** command to enable entity instance monitoring collection for a node process entity instance. **Syntax:** **performance-mgmt** **apply monitor node** **process** **location** *node-id* *pid* {*template-name* | **default**} `RP/0/RP0/CPU0:Router(config)# performance-mgmt apply monitor node process location p 0/RP0/CPU0 275 default`

Check System Health

Monitoring systems in a network proactively helps prevent potential issues and take preventive actions. This section illustrates how you can monitor the system health using the health check service. This service helps to analyze the system health by monitoring, tracking and analyzing metrics that are critical for functioning of the router.

The system health can be gauged with the values reported by these metrics when the configured threshold values exceed or are nearing the threshold value.

This table describes the significant fields shown in the display.

Table 12: System Health Check Metrics

Metric	Parameter Tracked	Considered Unhealthy When
System Resources	CPU, free memory, file system, shared memory	The respective metric has exceeded the threshold
Infrastructure Services	Field Programmable Device (FPD), fabric health, platform, redundancy	Any component of the service is down or in an error state
Counters	Interface-counters, fabric-statistics, asic-errors	Any specific counter exhibits a consistent increase in drop/error count over the last n runs (n is configurable through CLI, default is 10)

By default, metrics for system resources are configured with preset threshold values. You can customize the metrics to be monitored by disabling or enabling metrics of interest based on your requirement.

Each metric is tracked and compared with that of the configured threshold, and the state of the resource is classified accordingly.

The system resources exhibit one of these states:

- **Normal:** The resource usage is less than the threshold value.
- **Minor:** The resource usage is more than the minor threshold, but less than the severe threshold value.
- **Severe:** The resource usage is more than the severe threshold, but less than the critical threshold value.
- **Critical:** The resource usage is more than the critical threshold value.

The infrastructure services show one of these states:

- **Normal:** The resource operation is as expected.
- **Warning:** The resource needs attention. For example, a warning is displayed when the FPD needs an upgrade.

The health check service is packaged as an optional RPM. This is not part of the base package and you must explicitly install this RPM.

You can configure the metrics and their values using CLI. In addition to the CLI, the service supports NETCONF client to apply configuration (`Cisco-IOS-XR-healthcheck-cfg.yang`) and retrieve operational data (`Cisco-IOS-XR-healthcheck-oper.yang`) using YANG data models. It also supports subscribing to

metrics and their reports to stream telemetry data. For more information about streaming telemetry data, see *Telemetry Configuration Guide for Cisco 8000 Series Routers*.

Here is a sample of Cisco-IOS-XR-health-check-cfg.yang data model to configure the health check metrics:

```
module Cisco-IOS-XR-health-check-cfg {
    namespace http://cisco.com/ns/yang/Cisco-IOS-XR-health-check-cfg;
    prefix "health-check-cfg";

    import Cisco-IOS-XR-types {
        prefix "xr";
    }

    container metric-cfg {
        list cpu {
            key "name";
            description
                "system 15min avg cpu utilization";
            leaf name {
                type string;
            }
            leaf enabled {
                type boolean;
                default "true";
            }
            leaf threshold {
                type uint32 {
                    range "1..100";
                }
                units "percent";
                default "20";
                description
                    "cpu system utilization should be less than threshold";
            }
            leaf node {
                type string;
                default "all";
            }
        }
        list free_memory {
            key "name";
            description
                "system free RAM";
            leaf name {
                type string;
            }
            leaf enabled {
                type boolean;
                default "true";
            }
            leaf threshold {
                type uint32 {
                    range "1..4096";
                }
                units "MB"
                default "1024";
                description
                    "system free memory should be greater than threshold";
            }
            leaf node {
                type string;
                default "all";
            }
```

```
        }
    }
}
```

Associated Commands

- healthcheck
- healthcheck metric
- show healthcheck metric
- show healthcheck report
- show healthcheck status

Restrictions of the Health Check Feature

The following restrictions are applicable for the system health check feature:

- Once the user configures the health check feature through the command line interface, the feature is in **Configured** state in the output of **show healthcheck status**. Enabling the netconf-yang agent is a prerequisite to enable the health check feature. When the user enables netconf-yang agent, the status of the health check feature changes to **Enabled**.
- The system health data is available only when the health check feature is in **Enabled** state. If the user does not configure the netconf-yang agent, the status of the health check feature will remain in **Configured** state.
- If the user disables the netconf-yang agent after enabling the health check feature, the status of the health check feature changes back to the **Configured** state.
- The values reported in the interface-counters, asic-errors and fabric-stats are not accumulated statistics but rather they show the actual number of drops or errors in the collection window. The collection window is the cadence for the collectors.

Monitoring Critical System Resources

This task explains how to check the health of a system using operational data from the network. The data can be queried by both CLI and NETCONF RPC, and can also be streamed using telemetry.

Before you begin

Enable NETCONF-yang agent.

Step 1 Enable NETCONF, SSH and configure the management interface.

Example:

```
configure
line default
exec-timeout 0 0
session-timeout 0
!
```

```
line console
exec-timeout 0 0
session-timeout 0
!

vty-pool default 0 99 line-template default
!

netconf-yang agent ssh
!

ssh server v2
ssh server vrf default
ssh server netconf vrf default
ssh timeout 120
ssh server rate-limit 600
ssh server session-limit 110
!

commit
!

interface MgmtEth 0/RP0/CPU0/0
no shut
ipv4 address dhcp
commit
!

End
```

Step 2 Install the health check RPM.

Example:

```
install source <path-to-repository>/xr-healthcheck-<release-version>.x86_64.rpm
```

For instructions to download optional RPMs, see the *Software Installation Guide for Cisco 8000 Series Routers*.

Step 3 Check the status of all metrics with its associated threshold and configured parameters in the system.

Example:

```
Router#show healthcheck status
Healthcheck status: Enabled

Collector Cadence: 60 seconds

System Resource metrics
  cpu
      Thresholds: Minor: 10%
                  Severe: 20%
                  Critical: 30%

       Tracked CPU utilization: 15 min avg utilization

   free-memory
        Thresholds: Minor: 10%
                    Severe: 8%
                    Critical: 5%

   filesystem
        Thresholds: Minor: 80%
                    Severe: 95%
                    Critical: 99%
```

```
   shared-memory
        Thresholds: Minor: 80%
                    Severe: 95%
                    Critical: 99%

Infra Services metrics
   fpd

   fabric-health
```

Step 4 View the health state for each enabled metric.

Example:

```
Router#show healthcheck report
Healthcheck report for enabled metrics

cpu
  State: Normal

free-momry
   State: Normal

shared-memory
   State: Normal

fpd
   State: Warning
One or more FPDs are in NEED UPGD state

fabric-health
   State: Normal
```

In the above output, the state of the FPD shows a warning message that indicates an FPD upgrade is required.

To further investigate the warning message, check the metric information. Here, for example, check the FPD state.

```
FPD Metric State: Warning
Last Update Time: 17 Feb 18:28:57.917193
FPD Service State: Enabled
Number of Active Nodes: 69

Node Name: 0/0/CPU0
    Card Name: 8800-LC-48H
    FPD Name: Bios
    HW Version: 0.31
    Status: NEED UPGD
    Run Version: 5.01
    Programmed Version: 5.01

-------------Truncated for brevity---------------
```

The `Last Update Time` is the timestamp when the health for that metric was computed. This timestamp gets refreshed with each collector run based on the cadence.

Note With the health check service enabled, no other configuration change is permitted. Before you change a configuration and commit the change, first disable the service.

```
Router#configure
Router(config)#no healthcheck enable
Router(config)#commit
```

Step 5 Customize the health check threshold value for the following parameters:

- **Cadence:** To change the preset cadence, use the command:

```
Router(config)#healthcheck cadence <30 - 1800>
```

- **Metric:** To list the metrics that can be configured, use the command:

```
Router(config)#healthcheck metric ?
  cpu            cpu configurations(cisco-support)
  fabric-health  fabric configurations(cisco-support)
  filesystem     Filesystem usage configurations(cisco-support)
  fpd            FPD configurations(cisco-support)
  free-mem       free memory configurations(cisco-support)
  shared-mem     shared memory configurations(cisco-support)
```

For example, to change the preset value of CPU metric, use the command:

```
Router(config)#healthcheck metric cpu ?
   threshold minor, severe or critical threshold
   avg_cpu_util 1min, 5min or 15min
        ios(config)#healthcheck metric cpu threshold ?
        minor       minor threshold in %
        severe      severe threshold in %
        critical    critical threshold in %
```

- Disable or enable metrics to selectively filter some metrics. By default, all metrics are enabled.

```
Router(config)#[no] healthcheck metric cpu disable
Router(config)#[no] healthcheck metric free-mem disable
```

Monitoring Infrastructure Services

This task explains how to check the health of the infrastructure services of a system. The data can be queried by both CLI and NETCONF RPC, and can also be streamed using telemetry.

Before you begin

Enable NETCONF-yang agent.

Perform steps 1-3 from the section Monitoring Critical System Resources, on page 51

Step 1 Check the health status of the infrastructure metrics in the system. By default, the router software enables the health check for infrastructure services.

Example:

The below example shows how to obtain the health-check status for the platform metric:

```
Router# show healthcheck metric platform
Platform Metric State: Normal ==========> Health of the metric
Last Update Time: 25 Jun 05:17:03.508172 =====> Timestamp at which the metric data was collected
Platform Service State: Enabled =====> Service state of Platform
Number of Racks: 1 ======> Total number of racks in the testbed
Rack Name: 0
```

```
Number of Slots: 12
Slot Name: RP0
Number of Instances: 2
Instance Name: CPU0
Node Name 0/RP0/CPU0
Card Type 8800-RP
Card Redundancy State Active
Admin State NSHUT
Oper State IOS XR RUN
```

To retrieve the health-check information for the platform metric using the Netconf GET operation, use the schema shown below:

```
<health-check xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-healthcheck-oper">
<metric-info>
<platform/>
</metric-info>
</health-check>
```

Example:

The below example shows how to obtain the health-check status for the redundancy metric:

```
Router# show healthcheck metric redundancy
Redundancy Metric State: Normal ==========> Health of the metric
Last Update Time: 25 Jun 05:21:14.562291 =====> Timestamp at which the metric data was collected
Redundancy Service State: Enabled =====> Service state of the metric
Active: 0/RP0/CPU0
Standby: 0/RP1/CPU0
HA State: Node Ready
NSR State: Ready
```

To retrieve the health-check information for the redundancy metric using the Netconf GET operation, use the schema shown below:

```
<health-check xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-healthcheck-oper">
<metric-info>
<redundancy/>
</metric-info>
</health-check>
```

Step 2 Disable health-check of any of the metrics, if required. By default, all metrics are enabled.

Example:

The below example shows how to disable the health-check status for the platform metric:

```
Router(config)# healthcheck metric platform disable
Router(config)# commit
```

To disable the health-check for the platform metric using the Netconf, use the schema shown below:

```
<health-check xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-healthcheck-cfg">
<ord-b>
<metric>
<platform>
<disable/>
</platform>
</metric>
</ord-b>
</health-check>
```

Example:

The below example shows how to disable the health-check status for the redundancy metric:

```
Router(config)# healthcheck metric redundancy disable
Router(config)# commit
```

To disable the health-check for the redundancy metric using the Netconf, use the schema shown below:

```
<health-check xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-healthcheck-cfg">
<ord-b>
<metric>
<redundancy>
<disable/>
</redundancy>
</metric>
</ord-b>
</health-check>
```

Monitoring Counters

This task explains how to check the health of the counters of a system. The counter values that can be monitored are interface-counters, asic-errors and fabric-statistics.

Before you begin

Enable NETCONF-yang agent.

Perform steps 1-3 from the section Monitoring Critical System Resources, on page 51

Step 1 Configure the size of the buffer which stores the history of the counter values as shown in the below examples.

Example:

The below example shows how to configure the buffer-size for the **interface-counters** to store values for the last 5 cadence snapshots:

```
Router(config)# healthcheck metric intf-counters counter-size 5
Router(config)# commit
```

Example:

The below example shows how to configure the buffer-size for the **asic-errors** counters to store values for the last 5 cadence snapshots:

```
Router(config)# healthcheck metric asic-errors counter-size 5
Router(config)# commit
```

Example:

The below example shows how to configure the buffer-size for the **fabric-stats** counters to store values for the last 5 cadence snapshots:

```
Router(config)# healthcheck metric fabric-stats counter-size 5
Router(config)# commit
```

Step 2 Configure the list of interfaces for which the **interface-counters** should be tracked as shown in the below examples. This is possible only for the **interface-counters** metric.

Example:

The below example shows how to configure the list of interfaces for which the **interface-counters** need to be tracked:

```
Router(config)# healthcheck metric intf-counters intf-list MgmtEth0/RP0/CPU0/0 HundredGigE0/0/0/0
Router(config)# commit
```

Example:

The below example shows how to configure all the interfaces so that the **interface-counters** are tracked for them:

```
Router(config)# healthcheck metric intf-counters intf-list all
Router(config)# commit
```

Step 3 By default, the router software enables the health-check for counters. Check the health status of the counters in the system as shown in the below examples.

Example:

The below example shows how to obtain the health-check status for the interface-counters:

```
Router# show healthcheck metric interface-counters summary
Interface-counters Health State: Normal ==========> Health of the metric
Last Update Time: 25 Jun 05:59:33.965851 =====> Timestamp at which the metric data was collected
Interface-counters Service State: Enabled =====> Service state of the metric
Interface MgmtEth0/RP0/CPU0/0 =====> Configured interface for healthcheck monitoring
Counter-Names Count Average Consistently-Increasing
-----------------------------------------------------------------------------------------------
output-buffers-failures 0 0 N
Counter-Names =====> Name of the counters
Count =====> Value of the counter collected at "Last Update Time"
Average =====> Average of all values available in buffer
Consistently-Increasing =====> Trend of the counter values, as per data available in buffer

Router# show healthcheck metric interface-counters detail all
Thu Jun 25 06:02:03.145 UTC
Last Update Time: 25 Jun 06:01:35.217089 =====> Timestamp at which the metric data was collected
Interface MgmtEth0/RP0/CPU0/0 =====> Configured interface for healthcheck monitoring
Following table displays data for last <x=5> values collected in periodic cadence intervals
-----------------------------------------------------------------------------------------------------------
Counter-name Last 5 values
LHS = Earliest RHS = Latest
-----------------------------------------------------------------------------------------------------------
output-buffers-failures 0 0 0 0 0
parity-packets-received 0 0 0 0 0
```

Example:

The below example shows how to obtain the health-check status for the asic-errors:

```
Router# show healthcheck metric asic-errors summary
Asic-errors Health State: Normal ==========> Health of the metric
Last Update Time: 25 Jun 06:20:47.65152 =====> Timestamp at which the metric data was collected
Asic-errors Service State: Enabled =====> Service state of the metric
Node Name: 0/1/CPU0 =====> Node name for healthcheck monitoring

Instance: 0 =====> Instance of the Node

Counter-Names Count Average Consistently-Increasing
-----------------------------------------------------------------------------------------------
Link Errors 0 0 N
Counter-Names =====> Name of the counters
Count =====> Value of the counter collected at "Last Update Time"
Average =====> Average of all values available in buffer
Consistently-Increasing =====> Trend of the counter values, as per data available in buffer

Router# show healthcheck metric asic-errors detail all
Thu Jun 25 06:25:13.778 UTC
Last Update Time: 25 Jun 06:24:49.510525 =====> Timestamp at which the metric data was collected
Node Name: 0/1/CPU0 =====> Node name for healthcheck monitoring
Instance: 0 =====> Instance of the Node
Following table displays data for last <x=5> values collected in periodic cadence intervals
-----------------------------------------------------------------------------------------------------------
Counter-name Last 5 values
LHS = Earliest RHS = Latest
```

```
-------------------------------------------------------------------------------------------------------
Link Errors            0     0     0     0     0
```

Example:

The below example shows how to obtain the health-check status for the fabric-stats:

```
Router# show healthcheck metric fabric-stats summary
Thu Jun 25 06:51:13.154 UTC
Fabric-stats Health State: Normal ==========> Health of the metric
Last Update Time: 25 Jun 06:51:05.669753 =====> Timestamp at which the metric data was collected
Fabric-stats Service State: Enabled =====> Service state of the metric
Fabric plane id 0 =====> Plane ID
Counter-Names Count Average Consistently-Increasing
-------------------------------------------------------------------------------------------
mcast-lost-cells 0 0 N
Counter-Names =====> Name of the counters
Count =====> Value of the counter collected at "Last Update Time"
Average =====> Average of all values available in buffer
Consistently-Increasing =====> Trend of the counter values, as per data available in buffer

Router# show healthcheck metric fabric-stats detail all
Thu Jun 25 06:56:20.944 UTC
Last Update Time: 25 Jun 06:56:08.818528 =====> Timestamp at which the metric data was collected
Fabric Plane id 0 =====> Fabric Plane ID

Following table displays data for last <x=5> values collected in periodic cadence intervals
-------------------------------------------------------------------------------------------------------
Counter-name Last 5 values
LHS = Earliest RHS = Latest
-------------------------------------------------------------------------------------------------------
mcast-lost-cells 0 0 0 0 0
```

Step 4 If required, disable health-check of any of the counters. By default, all counters are enabled.

Example:

The below example shows how to disable the health-check status for the interface-counters:

```
Router(config)# healthcheck metric intf-counters disable
Router(config)# commit
```

Example:

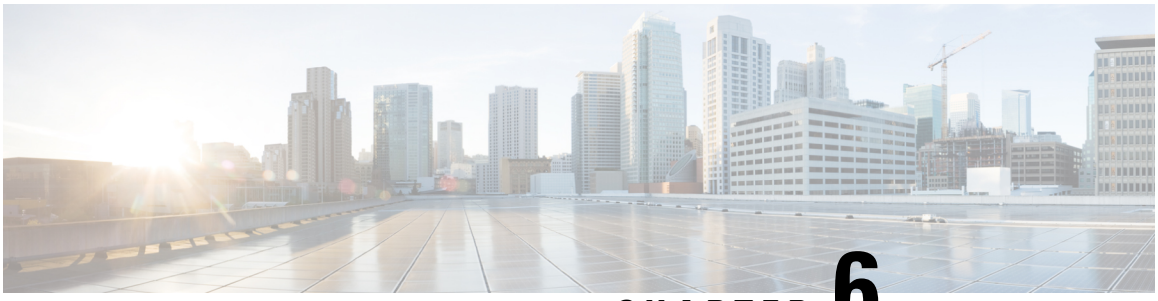
The below example shows how to disable the health-check status for the asic-errors:

```
Router(config)# healthcheck metric asic-errors disable
Router(config)# commit
```

Example:

The below example shows how to disable the health-check status for the fabric-stats:

```
Router(config)# healthcheck metric fabric-stats disable
Router(config)# commit
```



CHAPTER 6

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 60
- Information About Configuring and Managing Embedded Event Manager Policies, on page 60
- How to Configure and Manage Embedded Event Manager Policies, on page 68
- Configuration Examples for Writing Embedded Event Manager Policies Using Tcl , on page 86
- Embedded Event Manager Policy Tcl Command Extension Reference, on page 87

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 61.

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.

Regular Embedded Event Manager Scripts

Regular EEM scripts are used to implement policies when an EEM event is published. 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.

The first executable line of code within an EEM script must be the **eem event register** keyword. This keyword identifies the EEM event for which that script should be scheduled. The keyword is used by the **event manager policy** configuration command to register to handle the specified EEM event.

When an EEM script exits, it is responsible for setting a return code that is used to tell the EEM whether to run the default action for this EEM event (if any) or no other action. If multiple event handlers are scheduled for a given event, the return code from the previous handler is passed into the next handler, which can leave the value as is or update it.

Note An EEM script cannot register to handle an event other than the event that caused it to be scheduled.

Embedded Event Manager Policy Tcl Command Extension Categories

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

Table 13: 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.
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 14: Two-Character Abbreviation Specification

Two-Character Abbreviation	Specification
ap	event_register_appl
no	event_register_none

Two-Character Abbreviation	Specification
oi	event_register_oir
pr	event_register_process
sl	event_register_syslog
tm	event_register_timer
ts	event_register_timer_subscriber

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 15: 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 XR EXEC 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 16: 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 17: 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.
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 XR7 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 XR7 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 XR7 software with minimum change.

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

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 XR7 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 XR7 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.

Displaying EEM Registered Policies

Perform this optional task to display EEM registered policies.

SUMMARY STEPS

1. **show event manager policy registered** [**event-type** *type*] [**system** | **user**] [**time-ordered** | **name-ordered**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**show event manager policy registered** [**event-type** *type*] [**system** | **user**] [**time-ordered** | **name-ordered**] **Example:** `Router# show event manager policy registered system`	Displays information about currently registered policies. • The **event-type** keyword displays the registered policies for a specific event type. • The **time-ordered** keyword displays information about currently registered policies sorted by time. • The **name-ordered** keyword displays the policies in alphabetical order by the policy name.

Unregistering EEM Policies

Perform this task to remove an EEM policy from the running configuration file. Execution of the policy is canceled.

SUMMARY STEPS

1. **show event manager policy registered** [**event-type** *type*] [**system** | **user**] [**time-ordered** | **name-ordered**]
2. **configure**
3. **no event manager policy** *policy-name*
4. Use the **commit** or **end** command.
5. Repeat step 1 to ensure that the policy has been removed.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**show event manager policy registered** [**event-type** *type*] [**system** | **user**] [**time-ordered** | **name-ordered**] **Example:** `Router# show event manager policy registered system`	Displays information about currently registered policies. • The **event-type** keyword displays the registered policies for a specific event type. • The **time-ordered** keyword displays information about currently registered policies sorted by time. • The **name-ordered** keyword displays the policies in alphabetical order by the policy name.
Step 2	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters mode.
Step 3	**no event manager policy** *policy-name* **Example:** `Router(config)# no event manager policy tm_cli_cmd.tcl`	Removes the EEM policy from the configuration, causing the policy to be unregistered.
Step 4	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: • **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 5	Repeat step 1 to ensure that the policy has been removed.	—

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
```

Sample EEM Policies

Cisco IOS XR7 software contains some sample policies in the images that contain the EEM. Developers of EEM policies may modify these policies by customizing the event for which the policy is to be run and the options associated with logging and responding to the event. In addition, developers may select the actions to be implemented when the policy runs.

The Cisco IOS XR7 software includes a set of sample policies (see *Sample EEM Policy Descriptions* table). The sample policies can be copied to a user directory and then modified. Tcl is currently the only scripting language supported by Cisco for policy creation. Tcl policies can be modified using a text editor such as Emacs. Policies must execute within a defined number of seconds of elapsed time, and the time variable can be configured within a policy. The default is 20 seconds.

Sample EEM policies can be seen on the router using the CLI

```
Show event manager policy available system
```

This table describes the sample EEM policies.

Table 18: Sample EEM Policy Descriptions

Name of Policy	Description
periodic_diag_cmds.tcl	This policy is triggered when the _cron_entry_diag cron entry expires.Then, the output of this fixed set is collect for the fixed set of commands and the output is sent by email.
periodic_proc_avail.tcl	This policy is triggered when the _cron_entry_procavail cron entry expires. Then the output of this fixed set is collect for the fixed set of commands and the output is sent by email.
periodic_sh_log.tcl	This policy is triggered when the _cron_entry_log cron entry expires, and collects the output for the show log command and a few other commands. If the environment variable _log_past_hours is configured, it collects the log messages that are generated in the last _log_past_hours hours. Otherwise, it collects the full log.
sl_sysdb_timeout.tcl	This policy is triggered when the script looks for the sysdb timeout ios_msgs and obtains the output of the show commands. The output is written to a file named after the blocking process.
tm_cli_cmd.tcl	This policy runs using a configurable CRON entry. It executes a configurable CLI command and e-mails the results.
tm_crash_hist.tcl	This policy runs at midnight each day and e-mails a process crash history report to a specified e-mail address.

SUMMARY STEPS

1. **show event manager policy available** [**system** | **user**]
2. **configure**
3. **event manager directory user** {**library** *path* | **policy** *path*}
4. **event manager policy** *policy-name* **username** *username* [**persist-time** [*seconds* | **infinite**] | **type** [**system** | **user**]]
5. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**show event manager policy available** [**system**	**user**] **Example:** `Router# show event manager policy available`	Displays EEM policies that are available to be registered.
Step 2	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters mode.

	Command or Action	Purpose
Step 3	**event manager directory user** {**library** *path* | **policy** *path*} **Example:** `Router(config)# event manager directory user library disk0:/user_library`	Specifies a directory to use for storing user library files or user-defined EEM policies.
Step 4	**event manager policy** *policy-name* **username** *username* [**persist-time** [*seconds* | **infinite**] | **type** [**system** | **user**]] **Example:** `Router(config)# event manager policy test.tcl username user_a type user`	Registers the EEM policy to be run when the specified event defined within the policy occurs.
Step 5	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: • **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.

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 83

Procedure

	Command or Action	Purpose
Step 1	**show event manager policy available** [**system** | **user**] **Example:** `RP/0/RP0/CPU0:Router# show event manager policy available`	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:

	Command or Action	Purpose
		• event_register_appl • event_register_oir • event_register_process • event_register_syslog • event_register_timer • event_register_timer_subscriber • event_register_none

Step 4	Add the appropriate namespace under the ::cisco hierarchy.	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:
		• ::cisco::eem
		  • EEM event registration
		  • EEM event information
		  • EEM event publish
		  • EEM action
		  • EEM utility
		  • EEM context library
		  • EEM system information
		  • CLI library
		• ::cisco::lib
		  • SMTP library
		Note Ensure that the appropriate namespaces are imported, or use the qualified command names when using the preceding commands.

Step 5	Program the must defines section to check for each environment variable that is used in this policy.	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

	Command or Action	Purpose
		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 "_". **Note** You can display the Embedded Event Manager environment variables set on your system by using the **show event manager environment** command. 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. • **_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.	In this section of the script, you can define any of 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. • The **context_save** and **context_retrieve** Tcl command extensions that are used to save Tcl variables for use by other policies.

	Command or Action	Purpose
		• 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.	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).
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 (_cerrno) Tcl global variables.	Some EEM Tcl command extensions set a Cisco Error Number Tcl global variable _cerrno. Whenever _cerrno is set, four other Tcl global variables are derived from _cerrno and are set along with it (_cerr_sub_num, _cerr_sub_err, , and _cerr_str).
Step 10	Save the Tcl script with a new filename, and copy the Tcl script to the router.	Embedded Event Manager policy filenames adhere to the following specification: • 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 14: Two-Character Abbreviation Specification, on page 62) for the first event specified, an underscore character part, and a descriptive field part further identifying the policy. • A filename suffix part defined as .tcl. For more details, see the Cisco File Naming Convention for Embedded Event Manager, on page 62. Copy the file to the flash file system on the router—typically disk0:.
Step 11	**configure**	Enters global configuration mode.
Step 12	**event manager directory user** {**library** *path* | **policy** *path*} **Example:** `RP/0/RP0/CPU0:Router(config)# event manager directory user library disk0:/user_library`	Specifies a directory to use for storing user library files or user-defined EEM policies.

	Command or Action	Purpose
Step 13	**event manager policy** *policy-name* **username** *username* [**persist-time** [*seconds* | **infinite**] | **type** [**system** | **user**]] **Example:** `RP/0/RP0/CPU0:Router(config)# event manager policy test.tcl username user_a type user`	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. **end** —Prompts user to take one of these actions: • **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.

Procedure

	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 tclIndex on a workstation running the Tcl shell:
		lib1.tcl
		`proc test1 {} {`
		`  puts "In procedure test1"`
		`}`
		`proc test2 {} {`
		`  puts "In procedure test2"`
		`}`
		lib2.tcl
		`proc test3 {} {`

	Command or Action	Purpose
		`  puts "In procedure test3"` `}`
Step 2	**tclsh** **Example:** `workstation% tclsh`	Enters the Tcl shell.

Step 3	**auto_mkindex** *directory_name* ***.tcl**
	Example:
	`workstation% auto_mkindex eem_library *.tcl`
		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.
		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:
		tclIndex
		`# 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]]`

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.	The directory can be the same directory used in step 4.
		The following example user-defined EEM policy can be used to test the Tcl library support in EEM:
		libtest.tcl
		`::cisco::eem::event_register_none`
		`namespace import ::cisco::eem::*`

	Command or Action	Purpose

		`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"`
		`}`

Step 6	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters mode.
Step 7	**event manager directory user library** *path* **Example:** `RP/0/RP0/CPU0:Router(config)# event manager directory user library disk0:/eem_library`	Specifies the EEM user library directory; this is the directory to which the files in step 4 were copied.
Step 8	**event manager directory user policy** *path* **Example:** `RP/0/RP0/CPU0:Router(config)# event manager directory user policy disk0:/eem_policies`	Specifies the EEM user policy directory; this is the directory to which the file in step 5 was copied.
Step 9	**event manager policy** *policy-name* **username** *username* [**persist-time** [*seconds* | **infinite**] | **type** [**system** | **user**]] **Example:** `RP/0/RP0/CPU0:Router(config)# event manager policy libtest.tcl username user_a`	Registers a user-defined EEM policy.
Step 10	**event manager run** *policy* [*argument*] **Example:** `RP/0/RP0/CPU0:Router(config)# event manager run libtest.tcl`	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: • **Yes** — Saves configuration changes and exits the configuration session.

	Command or Action	Purpose
		• **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.

Procedure

	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:** `workstation% tclsh`	Enters the Tcl shell.
Step 3	**pkg_mkindex** *directory_name* ***.tcl** **Example:** `workstation% pkg_mkindex eem_library *.tcl`	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.

	Command or Action	Purpose

		The following example pkgIndex is created when some Tcl package files are in a library file directory and the pkg_mkindex command is run:
		pkgIndex
		`# 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.`
		`package ifneeded xmlrpc 0.3 [list source [file join $dir xmlrpc.tcl]]`

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.	The directory can be the same directory used in step 4.
		The following example user-defined EEM policy can be used to test the Tcl library support in EEM:
		packagetest.tcl
		`::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?"`

Step 6	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters mode.
Step 7	**event manager directory user library** *path* **Example:**	Specifies the EEM user library directory; this is the directory to which the files in step 4 were copied.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:Router(config)# event manager directory user library disk0:/eem_library`	
Step 8	**event manager directory user policy** *path* **Example:** `RP/0/RP0/CPU0:Router(config)# event manager directory user policy disk0:/eem_policies`	Specifies the EEM user policy directory; this is the directory to which the file in step 5 was copied.
Step 9	**event manager policy** *policy-name* **username** *username* [**persist-time** [*seconds* | **infinite**] | **type** [**system** | **user**]] **Example:** `RP/0/RP0/CPU0:Router(config)# event manager policy packagetest.tcl username user_a`	Registers a user-defined EEM policy.
Step 10	**event manager run** *policy* [*argument*] **Example:** `RP/0/RP0/CPU0:Router(config)# event manager run packagetest.tcl`	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: • **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.

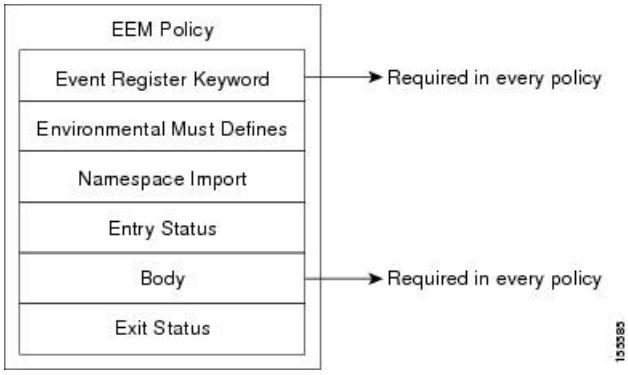
EEM Policies Using TCL: Details

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 3: 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 **con text_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 _cerrno. Whenever the _cerrno variable is set, the other Tcl global variables are derived from _cerrno and are set along with it (_cerr_sub_num, _cerr_sub_err, and _cerr_str).

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

```
XYSSSSSSSSSSSSSEEEEEEEEPPPPPPPP
```

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

Table 19: _cerrno: 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 CERR_CLASS_WARNING: See #unique_97 unique_97_Connect_42_tab_1130225 for the four possible error class encodings specific to this variable.
SSSSSSSSSSSSS	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 $_cerr_sub_num.

Variable	Description
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 $_cerr_sub_err.

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 20: 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.

Configuration Examples for Writing Embedded Event Manager Policies Using Tcl

EEM Sample Policy Descriptions

The configuration example features one sample EEM policy. The tm_cli_cmd.tcl runs using a configurable CRON entry. This policy executes a configurable CLI command and e-mails the results.

Registration of Some EEM Policies

Some EEM policies must be unregistered and then reregistered if an EEM environment variable is modified after the policy is registered. The event_register_ *xxx* statement that appears at the start of the policy contains some of the EEM environment variables, and this statement is used to establish the conditions under which the policy is run. If the environment variables are modified after the policy has been registered, the conditions may become invalid. To avoid any errors, the policy must be unregistered and then reregistered. The following variables are affected:

- _cron_entry in the tm_cli_cmd.tcl policy
- _syslog_pattern in the sl_intf_down.tcl policy

Basic Configuration Details for All Sample Policies

To allow e-mail to be sent from the Embedded Event Manager (EEM), the **hostname** and **domain-name** commands must be configured. The EEM environment variables must also be set. After a Cisco IOS XR7 software image has been booted, use the following initial configuration, substituting appropriate values for your network:

```
hostname cpu
example.com
event manager environment _email_server ms.example.net
event manager environment _email_to username@example.net
event manager environment _email_from engineer@example.net
event manager environment _email_cc projectgroup@example.net
event manager environment _cron_entry 0-59/2 0-23/1 * * 0-7
event manager environment _show_cmd show event manager policy registered
event manager environment _syslog_pattern .*UPDOWN.*FastEthernet0/0
event manager environment _config_cmd1 interface Ethernet1/0
event manager environment _config_cmd2 no shutdown
event manager environment _crash_reporter_debug 1
event manager environment _crash_reporter_url
http://www.example.com/fm/interface_tm.cgi
end
```

Embedded Event Manager Policy Tcl Command Extension Reference

This section documents the following EEM policy Tcl command extension categories:

Note For all EEM Tcl command extensions, if there is an error, the returned Tcl result string contains the error information.

Note Arguments for which no numeric range is specified take an integer from -2147483648 to 2147483647, inclusive.

The following conventions are used for the syntax documented on the Tcl command extension pages:

- An optional argument is shown within square brackets, for example:

  ```
  [type ?]
  ```

- A question mark ? represents a variable to be entered.
- Choices between arguments are represented by pipes, for example:

```
[queue_priority low|normal|high]
```

Embedded Event Manager Event Registration Tcl Command Extensions

The following EEM event registration Tcl command extensions are supported:

event_register_appl

Registers for an application event. Use this Tcl command extension to run a policy when an application event is triggered following another policy's execution of an event_publish Tcl command extension; the event_publish command extension publishes an application event.

To register for an application event, a subsystem must be specified. Either a Tcl policy or the internal EEM API can publish an application event. If the event is being published by a policy, the *sub_system* argument that is reserved for a policy is 798.

Syntax

```
event_register_appl [sub_system ?] [type ?] [queue_priority low|normal|high] [maxrun ?]
[nice 0|1]
```

Arguments

sub_system	(Optional) Number assigned to the EEM policy that published the application event. The number is set to 798, because all other numbers are reserved for Cisco use. If this argument is not specified, all components are matched.
type	(Optional) Event subtype within the specified event. The *sub_system* and *type* arguments uniquely identify an application event. If this argument is not specified, all types are matched. If you specify this argument, you must choose an integer between 1 and 4294967295, inclusive. There must be a match of component and type between the **event_publish** command extension and the **event_register_appl** command extension for the publishing and registration to work.
queue_priority	(Optional) Priority level at which the script will be queued; normal priority is greater than low priority but less than high priority. The priority here is not execution priority, but queuing priority. If this argument is not specified, the default priority is normal.
maxrun	(Optional) Maximum run time of the script (specified in SSSSSSSSSS[.MMM] format, where SSSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the default 20-second run-time limit is used.
nice	(Optional) Policy run-time priority setting. When the *nice* argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.

If multiple conditions exist, the application event is raised when all the conditions are satisfied.

Result String

None

Set _cerrno

No

event_register_cli

Registers for a CLI event. Use this Tcl command extension to run a policy when a CLI command of a specific pattern is entered based on pattern matching performed against an expanded CLI command. This will be implemented as a new process in IOS-XR which will be dlrsc_tracker. This ED will not do pattern match on admin commands of XR.

Note You can enter an abbreviated CLI command, such as **sh mem summary**, and the parser will expand the command to **show memory summary** to perform the matching. The functionality provided in the CLI event detector only allows a regular expression pattern match on a valid XR CLI command itself. This does not include text after a pipe character when redirection is used.

Syntax

```
event_register_cli [tag ?]
[occurs ?] [period ?] pattern ? [default ?] [queue_priority low|normal|high|last] [maxrun
?] [nice 0|1]
```

Arguments

tag	(Optional) String identifying a tag that can be used with the trigger Tcl command extension to support multiple event statements within a Tcl script.
occurs	(Optional) The number of occurrences before the event is raised. If this argument is not specified, the event is raised on the first occurrence. If this argument is specified, it must be an integer between 1 and 4294967295, inclusive.
period	(Optional) Specifies a backward looking time window in which all CLI events must occur (the occurs clause must be satisfied) in order for an event to be published (specified in SSSSSSSSSS[.MMM] format, where SSSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the most recent event is used.
pattern	(Mandatory) Specifies the regular expression used to perform the CLI command pattern match.
default	(Optional) The time period during which the CLI event detector waits for the policy to exit (specified in SSSSSSSSSS[.MMM] format, where SSSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If the default time period expires before the policy exits, the default action will be executed. The default action is to run the command. If this argument is not specified, the default time period is set to 30 seconds.

If multiple conditions are specified, the CLI event will be raised when all the conditions are matched.

Result String

None

Set _cerrno

No

event_register_config

Registers for a change in running configuration. Use this Tcl command extension to trigger a policy when there is any configuration change. This will be implemented as a new process in IOS-XR which will be dlrsc_tracker. This ED will not check for admin config changes in XR.

Syntax

```
event_register_config
[queue_priority low|normal|high|last]
[maxrun ?] [nice 0|1]
```

Arguments

queue_priority	(Optional) Priority level at which the script will be queued: • queue_priority low-Specifies that the script is to be queued at the lowest of the three priority levels. • queue_priority normal-Specifies that the script is to be queued at a priority level greater than low priority but less than high priority. • queue_priority high-Specifies that the script is to be queued at the highest of the three priority levels. • queue_priority last-Specifies that the script is to be queued at the lowest priority level. If more than one script is registered with the "queue_priority_last" argument set, these scripts will execute in the order in which the events are published. **Note** The queue_priority argument specifies the queuing priority, but not the execution priority, of the script being registered. If this argument is not specified, the default queuing priority is normal.
maxrun	(Optional) Maximum run time of the script (specified in SSSSSSSSSS[.MMM] format, where SSSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the default 20-second run-time limit is used.
nice	(Optional) Policy run-time priority setting. When the nice argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.

If multiple conditions are specified, the syslog event will be raised when all the conditions are matched.

Result String

None

Set _cerrno

No

event_register_none

Registers for an event that is triggered by the event manager run command. These events are handled by the None event detector that screens for this event.

Syntax

```
event_register_none [queue_priority low|normal|high] [maxrun ?] [nice 0|1]
```

Arguments

queue_priority	(Optional) Priority level at which the script will be queued; normal priority is greater than low priority but less than high priority. The priority here is not execution priority, but queuing priority. If this argument is not specified, the default priority is normal.
maxrun	(Optional) Maximum run time of the script (specified in SSSSSSSSSS[.MMM] format, where SSSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the default 20-second run-time limit is used.
nice	(Optional) Policy run-time priority setting. When the *nice* argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.

Result String

None

Set _cerrno

No

event_register_oir

Registers for an online insertion and removal (OIR) event. Use this Tcl command extension to run a policy on the basis of an event raised when a hardware card OIR occurs. These events are handled by the OIR event detector that screens for this event.

Syntax

```
event_register_oir [queue_priority low|normal|high] [maxrun ?] [nice 0|1]
```

Arguments

queue_priority	(Optional) Priority level at which the script will be queued; normal priority is greater than low priority but less than high priority. The priority here is not execution priority, but queuing priority. If this argument is not specified, the default priority is normal.
maxrun	(Optional) Maximum run time of the script (specified in SSSSSSSSSS[.MMM] format, where SSSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the default 20-second run-time limit is used.

nice	(Optional) Policy run-time priority setting. When the *nice* argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.

Result String

None

Set _cerrno

No

event_register_process

Registers for a process event. Use this Tcl command extension to run a policy on the basis of an event raised when a Cisco IOS XR7 software modularity process starts or stops. These events are handled by the system manager event detector that screens for this event. This Tcl command extension is supported only in software modularity images.

Syntax

```
event_register_process abort|term|start
[job_id ?] [instance ?] [path ?] [node ?]
[queue_priority low|normal|high] [maxrun ?] [nice 0|1] [tag?]
```

Arguments

abort	(Mandatory) Abnormal process termination. Process may terminate because of exiting with a nonzero exit status, receiving a kernel-generated signal, or receiving a SIGTERM or SIGKILL signal that is not sent because of user request.
term	(Mandatory) Normal process termination.
start	(Mandatory) Process start.
job_id	(Optional) Number assigned to the EEM policy that published the process event. Number is set to 798, because all other numbers are reserved for Cisco use.
instance	(Optional) Process instance ID. If specified, this argument must be an integer between 1 and 4294967295, inclusive.
path	(Optional) Process pathname (regular expression string).
node	(Optional) The node name is a string that consists of the word "node" followed by two fields separated by a slash (/), using the following format: node<slot-number>/<cpu-number> The slot-number is the hardware slot number. The cpu-number is the hardware CPU number. For example, the SP CPU in a Supervisor card on a Cisco Catalyst 6500 series switch located in slot 0 would be specified as node0/0. The RP CPU in a Supervisor card on a Cisco Catalyst 6500 series switch located in slot 0 would be addressed as node0/1. If the *node* argument is not specified, the default node specification is always the regular expression pattern match of * representing all applicable nodes.

queue_priority	(Optional) Priority level at which the script will be queued; normal priority is greater than low priority but less than high priority. The priority here is not execution priority, but queuing priority. If this argument is not specified, the default priority is normal.
maxrun	(Optional) Maximum run time of the script (specified in SSSSSSSSS[.MMM] format, where SSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the default 20-second run-time limit is used.
nice	(Optional) Policy run-time priority setting. When the *nice* argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.
tag	Tag is acceptable but ignored. Cisco IOS EEM scripts with the tag option can run in an Cisco IOS XR7 software environment without any error. Since Cisco IOS XR7 software does not support multiple events, the tag has no effect.

If an optional argument is not specified, the event matches all possible values of the argument. If multiple arguments are specified, the process event will be raised when all the conditions are matched.

Result String

None

Set _cerrno

No

event_register_snmp_notification

Registers for a Simple Network Management Protocol (SNMP) notification trap event. Use this Tcl command extension to run a policy when an SNMP trap with the specified SNMP object ID (oid) is encountered on a specific interface or address. The **snmp-server manager** CLI command must be enabled for the SNMP notifications to work using Tcl policies.

Syntax

```
event_register_snmp_notification [tag ?] oid ? oid_val ?
op {gt|ge|eq|ne|lt|le}
[src_ip_address ?]
[dest_ip_address ?]
[queue_priority {normal|low|high|last}]
[maxrun ?]
[nice {0|1}]
[default ?]
[direction {incoming|outgoing}]
[msg_op {drop|send}]
```

Argument

tag	(Optional) String identifying a tag that can be used with the trigger Tcl command extension to support multiple event statements within a Tcl script.
oid	(Mandatory) OID number of the data element in SNMP dot notation (for example, 1.3.6.1.2.1.2.1.0). If the specified OID ends with a dot (.), then all OIDs that start with the OID number before the dot are matched. It supports all OID supported by SNMP in XR.

oid_val	(Mandatory) OID value with which the current OID data value should be compared to decide if the SNMP event should be raised.
op	(Mandatory) Comparison operator used to compare the current OID data value with the SNMP Protocol Data Unit (PDU) OID data value; if this is true, an event is raised.
src_ip_address	(Optional) Source IP address where the SNMP notification trap originates. The default is all; it is set to receive SNMP notification traps from all IP addresses. This option will not be supported in XR as src_ip_address is only for incoming trap which is not supported in EEM XR.
dest_ip_address	(Optional) Destination IP address where the SNMP notification trap is sent. The default is all; it is set to receive SNMP traps from all destination IP addresses.
default	(Optional) Specifies the time period in seconds during which the snmp notification event detector waits for the policy to exit. Thetime periodis specified in sssssssss[.mmm] format, where sssssssss must be an integer representing seconds between 0 and 4294967295 and mmm must be an integer representing milliseconds between 0 and 999
direction	(Optional) The direction of the incoming or outgoing SNMP trap or inform PDU to filter. The default value is outgoing. For XR direction incoming will not be supported and policy registration will fail if user provides direction as incoming.
msg_op	(Optional) The action to be taken on the SNMP PDU (drop it or send it) once the event is triggered. The default value is send. For XR msg_op drop will not be supported and policy registration will fail if user provides msg_op as drop.

Result String

None

Set _cerrno

No

event_register_syslog

Registers for a syslog event. Use this Tcl command extension to trigger a policy when a syslog message of a specific pattern is logged after a certain number of occurrences during a certain period of time.

Syntax

```
event_register_syslog [occurs ?] [period ?] pattern ?
[priority all|emergencies|alerts|critical|errors|warnings|notifications|
informational|debugging|0|1|2|3|4|5|6|7]
[queue_priority low|normal|high]
[severity_fatal] [severity_critical] [severity_major]
[severity_minor] [severity_warning] [severity_notification]
[severity_normal] [severity_debugging]
[maxrun ?] [nice 0|1]
```

Arguments

occurs	(Optional) Number of occurrences before the event is raised; if not specified, the event is raised on the first occurrence. If specified, the value must be greater than 0.
period	(Optional) Time interval, in seconds and milliseconds, during which the one or more occurrences must take place in order to raise an event (specified in SSSSSSSSSS[.MMM] format where SSSSSSSSSS must be an integer number representing seconds between 0 and 4294967295, inclusive, and where MMM represents milliseconds and must be an integer number between 0 and 999). If this argument is not specified, no period check is applied.
pattern	(Mandatory) Regular expression used to perform syslog message pattern match. This argument is what the policy uses to identify the logged syslog message.
priority	(Optional) Message priority to be screened. If this argument is specified, only messages that are at the specified logging priority level, or lower, are screened. If this argument is not specified, the default priority is 0.
queue_priority	(Optional) Priority level at which the script will be queued; normal priority is greater than low priority but less than high priority. The priority here is not execution priority, but queuing priority. If this argument is not specified, the default priority is normal.
maxrun	(Optional) Maximum run time of the script (specified in SSSSSSSSSS[.MMM] format, where SSSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the default 20-second run-time limit is used.
nice	(Optional) Policy run-time priority setting. When the *nice* argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.

If multiple conditions are specified, the syslog event is raised when all the conditions are matched.

Table 21: Severity Level Mapping For Syslog Events

Severity Keyword	Syslog Priority	Description
severity_fatal	LOG_EMERG (0)	System is unusable.
severity_critical	LOG_ALERT (1)	Critical conditions, immediate attention required.
severity_major	LOG_CRIT (2)	Major conditions.
severity_minor	LOG_ERR (3)	Minor conditions.
severity_warning	LOG_WARNING (4)	Warning conditions.
severity_notification	LOG_NOTICE (5)	Basic notification, informational messages.
severity_normal	LOG_INFO (6)	Normal event, indicates returning to a normal state.
severity_debugging	LOG_DEBUG (7)	Debugging messages.

Result String

None

Set _cerrno

No

event_register_timer

Creates a timer and registers for a timer event as both a publisher and a subscriber. Use this Tcl command extension when there is a need to trigger a policy that is time specific or timer based. This event timer is both an event publisher and a subscriber. The publisher part indicates the conditions under which the named timer is to go off. The subscriber part identifies the name of the timer to which the event is subscribing.

Note Both the CRON and absolute time specifications work on local time.

Syntax

```
event_register_timer watchdog|countdown|absolute|cron
[name ?] [cron_entry ?]
[time ?]
[queue_priority low|normal|high] [maxrun ?]
[nice 0|1]
```

Arguments

watchdog	(Mandatory) Watchdog timer.
countdown	(Mandatory) Countdown timer.
absolute	(Mandatory) Absolute timer.
cron	(Mandatory) CRON timer.
name	(Optional) Name of the timer.

cron_entry	(Optional) Entry must be specified if the CRON timer type is specified. Must not be specified if any other timer type is specified. A cron_entry is a partial UNIX crontab entry (the first five fields) as used with the UNIX CRON daemon. A cron_entry specification consists of a text string with five fields. The fields are separated by spaces. The fields represent the time and date when CRON timer events will be triggered. The fields are described in Table 22: Time and Date When CRON Events Will Be Triggered , on page 98. Ranges of numbers are allowed. Ranges are two numbers separated with a hyphen. The specified range is inclusive. For example, 8-11 for an hour entry specifies execution at hours 8, 9, 10, and 11. A field may be an asterisk (*), which always stands for "first-last." Lists are allowed. A list is a set of numbers (or ranges) separated by commas. Examples: "1,2,5,9" and "0-4,8-12". Step values can be used in conjunction with ranges. Following a range with "/<number>" specifies skips of the number's value through the range. For example, "0-23/2" is used in the hour field to specify an event that is triggered every other hour. Steps are also permitted after an asterisk, so if you want to say "every two hours", use "*/2". Names can also be used for the month and the day of week fields. Use the first three letters of the particular day or month (case does not matter). Ranges or lists of names are not allowed. The day on which a timer event is triggered can be specified by two fields: day of month and day of week. If both fields are restricted (that is, are not *), an event will be triggered when either field matches the current time. For example, "30 4 1,15 * 5" would cause an event to be triggered at 4:30 a.m. on the 1st and 15th of each month, plus every Friday. Instead of the first five fields, one of seven special strings may appear. These seven special strings are described in Table 23: Special Strings for cron_entry, on page 98 Example 1: "0 0 1,15 * 1" would trigger an event at midnight on the 1st and 15th of each month, as well as on every Monday. To specify days by only one field, the other field should be set to *; "0 0 * * 1" would trigger an event at midnight only on Mondays. Example 2: "15 16 1 * *" would trigger an event at 4:15 p.m. on the first day of each month. Example 3: "0 12 * * 1-5" would trigger an event at noon on Monday through Friday of each week. Example 4: "@weekly" would trigger an event at midnight once a week on Sunday.
time	(Optional) Time must be specified if a timer type other than CRON is specified. Must not be specified if the CRON timer type is specified. For watchdog and countdown timers, the number of seconds and milliseconds until the timer expires; for the absolute timer, the calendar time of the expiration time. Time is specified in SSSSSSSSSS[.MMM] format, where SSSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999. An absolute expiration date is the number of seconds and milliseconds since January 1, 1970. If the date specified has already passed, the timer expires immediately.
queue_priority	(Optional) Priority level at which the script will be queued; normal priority is greater than low priority but less than high priority. The priority here is not execution priority, but queuing priority. If this argument is not specified, the default priority is normal.

maxrun	(Optional) Maximum run time of the script (specified in SSSSSSSSSS[.MMM] format, where SSSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the default 20-second run-time limit is used.
nice	(Optional) Policy run-time priority setting. When the *nice* argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.

Table 22: Time and Date When CRON Events Will Be Triggered

Field	Allowed Values
minute	0-59
hour	0-23
day of month	1-31
month	1-12 (or names, see Table 23: Special Strings for cron_entry, on page 98)
day of week	0-7 (0 or 7 is Sun, or names; see Table 23: Special Strings for cron_entry, on page 98)

Table 23: Special Strings for cron_entry

String	Meaning
@yearly	Trigger once a year, "0 0 1 1 *".
@annually	Same as @yearly.
@monthly	Trigger once a month, "0 0 1 * *".
@weekly	Trigger once a week, "0 0 * * 0".
@daily	Trigger once a day, "0 0 * * *".
@midnight	Same as @daily.
@hourly	Trigger once an hour, "0 * * * *".

Result String

None

Set _cerrno

No

See Also

#unique_113

event_register_timer_subscriber

Registers for a timer event as a subscriber. Use this Tcl command extension to identify the name of the timer to which the event timer, as a subscriber, wants to subscribe. The event timer depends on another policy or another process to actually manipulate the timer. For example, let policyB act as a timer subscriber policy, but policyA (although it does not need to be a timer policy) uses register_timer, timer_arm, or timer_cancel Tcl command extensions to manipulate the timer referenced in policyB.

Syntax

```
event_register_timer_subscriber watchdog|countdown|absolute|cron
name ? [queue_priority low|normal|high] [maxrun ?] [nice 0|1]
```

Arguments

watchdog	(Mandatory) Watchdog timer.
countdown	(Mandatory) Countdown timer.
absolute	(Mandatory) Absolute timer.
cron	(Mandatory) CRON timer.
name	(Mandatory) Name of the timer.
queue_priority	(Optional) Priority level at which the script will be queued; normal priority is greater than low priority but less than high priority. The priority here is not execution priority, but queuing priority. If this argument is not specified, the default priority is normal.
maxrun	(Optional) Maximum run time of the script (specified in SSSSSSSSSS[.MMM] format, where SSSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the default 20-second run-time limit is used.
nice	(Optional) Policy run-time priority setting. When the nice argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.

Note An EEM policy that registers for a timer event or a counter event can act as both publisher and subscriber.

Result String

None

Set _cerrno

No

See Also

event_register_timer, on page 96

event_register_track

Registers for a report event from the Object Tracking component in XR. Use this Tcl command extension to trigger a policy on the basis of a Object Tracking component report for a specified track. This will be implemented as a new process in IOS-XR which will be dlrsc_tracker. Please note that the manageability package should be installed for the track ED to be functional.

Syntax

```
event_register_track ? [tag ?] [state up|down|any] [queue_priority low|normal|high|last]
[maxrun ?]
[nice 0|1]
```

Arguments

? (represents a string)	(Mandatory) Tracked object name.
tag	(Optional) String identifying a tag that can be used with the trigger Tcl command extension to support multiple event statements within a Tcl script.
state	(Optional) Specifies that the tracked object transition will cause an event to be raised. If **up** is specified, an event will be raised when the tracked object transitions from a down state to an up state. If **down** is specified, an event will be raised when the tracked object transitions from an up state to a down state. If **any** is specified, an event will be raised when the tracked object transitions to or from any state.
queue_priority	(Optional) Priority level at which the script will be queued: • queue_priority low-Specifies that the script is to be queued at the lowest of the three priority levels. • queue_priority normal-Specifies that the script is to be queued at a priority level greater than low priority but less than high priority. • queue_priority high-Specifies that the script is to be queued at the highest of the three priority levels. • queue_priority last-Specifies that the script is to be queued at the lowest priority level. If more than one script is registered with the "queue_priority_last" argument set, these scripts will execute in the order in which the events are published. **Note** The queue_priority argument specifies the queuing priority, but not the execution priority, of the script being registered. If this argument is not specified, the default queuing priority is normal.
maxrun	(Optional) Maximum run time of the script (specified in SSSSSSSSSS[.MMM] format, where SSSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the default 20-second run-time limit is used.
nice	(Optional) Policy run-time priority setting. When the nice argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.

If an optional argument is not specified, the event matches all possible values of the argument.

Result String

None

Set _cerrno

No

Embedded Event Manager Event Information Tcl Command Extension

The following EEM Event Information Tcl Command Extensions are supported:

event_reqinfo

Queries information for the event that caused the current policy to run.

Syntax

```
event_reqinfo
```

Arguments

None

Result String

If the policy runs successfully, the characteristics for the event that triggered the policy will be returned. The following sections show the characteristics returned for each event detector.

For EEM_EVENT_APPLICATION

```
"event_id %u event_type %u event_type_string {%s} event_pub_sec %u event_pub_msec %u"
"sub_system 0x%x type %u data1 {%s} data2 {%s} data3 {%s} data4 {%s}"
```

Event Type	Description
event_id	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
event_type	Type of event.
event_type_string	ASCII string that represents the name of the event for this event type.
event_pub_secevent_pub_msec	The time, in seconds and milliseconds, when the event was published to the Embedded Event Manager.
sub_system	Number assigned to the EEM policy that published the application event. Number is set to 798 because all other numbers are reserved for Cisco use.

Event Type	Description
type	Event subtype within the specified component.
data1data2data3data4	Argument data that is passed to the application-specific event when the event is published. The data is character text, an environment variable, or a combination of the two.

For EEM_EVENT_COUNTER

```
"event_id %u event_type %u event_type_string {%s} %u event_pub_sec %u event_pub_msec %u"
"name {%s}"
```

Event Type	Description
event_id	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
event_type	Type of event.
event_type_string	ASCII string that represents the name of the event for this event type.
event_pub_secevent_pub_msec	The time, in seconds and milliseconds, when the event was published to the Embedded Event Manager.
name	Counter name.

For EEM_EVENT_NONE

```
"event_id %u event_type %u event_type_string {%s} event_pub_sec %u event_pub_msec %u"
```

Event Type	Description
event_id	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
event_type	Type of event.
event_type_string	ASCII string that represents the name of the event for this event type.
event_pub_secevent_pub_msec	Time, in seconds and milliseconds, when the event was published to the Embedded Event Manager.

For EEM_EVENT_OIR

```
"event_id %u event_type %u event_type_string {%s} event_pub_sec %u event_pub_msec %u"
"slot %u event %s"
```

Event Type	Description
event_id	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event ID.
event_type	Type of event.
event_type_string	ASCII string that represents the name of the event for this event type.
event_pub_secevent_pub_msec	Time, in seconds and milliseconds, when the event was published to the Embedded Event Manager.
slot	Slot number for the affected card.
event	Indicates a string, removed or online, that represents either an OIR removal event or an OIR insertion event.

For EEM_EVENT_PROCESS (Software Modularity Only)

```
"event_id %u event_type %u event_type_string {%s} event_pub_sec %u event_pub_msec %u"
"sub_system 0x%x instance %u process_name {%s} path {%s} exit_status 0x%x"
"respawn_count %u last_respawn_sec %ld last_respawn_msec %ld fail_count %u"
"dump_count %u node_name {%s}"
```

Event Type	Description
event_id	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
event_type	Type of event.
event_type_string	ASCII string that represents the name of the event for this event type.
event_pub_secevent_pub_msec	Time, in seconds and milliseconds, when the event was published to the Embedded Event Manager.
sub_system	Number assigned to the EEM policy that published the application-specific event. Number is set to 798 because all other numbers are reserved for Cisco use.
instance	Process instance ID.
process_name	Process name.
path	Process absolute name including path.
exit_status	Process last exit status.
respawn_count	Number of times that the process was restarted.
last_respawn_seclast_respawn_msec	Calendar time when the last restart occurred.

Event Type	Description
fail_count	Number of restart attempts of the process that failed. This count will be reset to 0 when the process is successfully restarted.
Event Type	**Description**
dump_count	Number of core dumps taken of the process.
node_name	Name of the node that the process is on. The node name is a string that consists of the word “node” followed by two fields separated by a slash character using the following format: node<slot-number>/<cpu-number> The slot-number is the hardware slot number. The cpu-number is the hardware CPU number.

For EEM_EVENT_RF

```
"event_id %u event_type %u event_type_string {%s} %u event_pub_sec %u event_pub_msec %u"
"event {%s}"
```

Event Type	Description
event_id	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
event_type	Type of event.
event_type_string	ASCII string that represents the name of the event for this event type.
event_pub_secevent_pub_msec	Time, in seconds and milliseconds, when the event was published to the Embedded Event Manager.
event	RF progression or status event notification that caused this event to be published.

For EEM_EVENT_SYSLOG_MSG

```
"event_id %u event_type %u event_type_string {%s} event_pub_sec %u event_pub_msec %u"
"msg {%s}"
```

Event Type	Description
event_id	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
event_type	Type of event.

Event Type	Description
event_type_string	ASCII string that represents the name of the event for this event type.
event_pub_secevent_pub_msec	Time, in seconds and milliseconds, when the event was published to the Embedded Event Manager.
msg	Last syslog message that matches the pattern.

For EEM_EVENT_TIMER_ABSOLUTE

EEM_EVENT_TIMER_COUNTDOWN

EEM_EVENT_TIMER_WATCHDOG

```
"event_id %u event_type %u event_type_string {%s} event_pub_sec %u event_pub_msec %u"
"timer_type %s timer_time_sec %ld timer_time_msec %ld"
"timer_remain_sec %ld timer_remain_msec %ld"
```

Event Type	Description
event_id	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
event_type	Type of event.
event_type_string	ASCII string that represents the name of the event for this event type.
event_pub_secevent_pub_msec	Time, in seconds and milliseconds, when the event was published to the Embedded Event Manager.
timer_type	Type of the timer. Can be one of the following: • watchdog • countdown • absolute
timer_time_sectimer_time_msec	Time when the timer expired.
timer_remain_sectimer_remain_msec	Remaining time before the next expiration.

For EEM_EVENT_TIMER_CRON

```
"event_id %u event_type %u event_type_string {%s} event_pub_sec %u event_pub_msec %u"
"timer_type {%s} timer_time_sec %ld timer_time_msec %ld"
```

Event Type	Description
event_id	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
event_type	Type of event.
event_type_string	ASCII string that represents the name of the event for this event type.
event_pub_secevent_pub_msec	Time, in seconds and milliseconds, when the event was published to the Embedded Event Manager.
timer_type	Type of the timer.
timer_time_sectimer_time_msec	Time when the timer expired.

For EEM_EVENT_TRACK

```
"event_id %u event_type %u event_type_string {%s} %u event_pub_sec %u event_pub_msec %u"
"track_number {%u} track_state {%s}"
```

Event Type	Description
event_id	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event ID.
event_type	Type of event.
event_type_string	ASCII string that represents the name of the event for this event type.
event_pub_secevent_pub_msec	Time, in seconds and milliseconds, when the event was published to the Embedded Event Manager.
track_number	Number of the tracked object that caused the event to be triggered.
track_state	State of the tracked object when the event was triggered; valid states are up or down.

For EEM_EVENT_WDSYSMON

```
"event_id %u event_type %u event_type_string {%s} %u event_pub_sec %u event_pub_msec %u"
"num_subs %u"
```

Event Type	Description
event_id	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.

Event Type	Description
event_type	Type of event.
event_type_string	ASCII string that represents the name of the event for this event type.
event_pub_secevent_pub_msec	Time, in seconds and milliseconds, when the event was published to the Embedded Event Manager.
num_subs	Subevent number.

Where the subevent info string is for a deadlock subevent:

```
"{type %s num_entries %u entries {entry 1, entry 2, ...}}"
```

Subevent Type	Description
type	Type of wdsysmon subevent.
num_entries	Number of processes and threads in the deadlock.
entries	Information of processes and threads in the deadlock.

Where each entry is:

```
"{node {%s} procname {%s} pid %u tid %u state %s b_node %s b_procname %s b_pid %u
b_tid %u}"
```

Assume that the entry describes the scenario in which Process A thread m is blocked on process B thread n:

Subevent Type	Description
node	Name of the node that process A thread m is on.
procname	Name of process A.
pid	Process ID of process A.
tid	Thread ID of process A thread m.

Subevent Type	Description
state	Thread state of process A thread m. Can be one of the following: • STATE_CONDVAR • STATE_DEAD • STATE_INTR • STATE_JOIN • STATE_MUTEX • STATE_NANOSLEEP • STATE_READY • STATE_RECEIVE • STATE_REPLY • STATE_RUNNING • STATE_SEM • STATE_SEND • STATE_SIGSUSPEND • STATE_SIGWAITINFO • STATE_STACK • STATE_STOPPED • STATE_WAITPAGE • STATE_WAITTHREAD
b_node	Name of the node that process B thread is on.
b_procname	Name of process B.
b_pid	Process ID of process B.
b_tid	Thread ID of process B thread n; 0 means that process A thread m is blocked on all threads of process B.

For dispatch_mgr Subevent

```
"{type %s node {%s} procname {%s} pid %u value %u sec %ld msec %ld}"
```

Subevent Type	Description
type	Type of wdsysmon subevent.
node	Name of the node that the POSIX process is on.
procname	POSIX process name for this subevent.
pid	POSIX process ID for this subevent. **Note** The three preceding fields describe the owner process of this dispatch manager.
value	If the sec and msec variables are specified as 0 or are unspecified in the event registration Tcl command extension, the number of events processed by the dispatch manager is in the latest sample. If a time window is specified and is greater than zero in the event registration Tcl command extension, the total number of events processed by this dispatch manager is in the given time window.
secmsec	If the sec and msec variables are specified as 0 or are unspecified in the event registration Tcl command extension, they are both 0. If a time window is specified and is greater than zero in the event registration Tcl command extension, the sec and msec variables are the actual time difference between the time stamps of the oldest and latest samples in this time window.

For cpu_proc Subevent

```
"{type %s node {%s} procname {%s} pid %u value %u sec %ld msec %ld}"
```

Subevent Type	Description
type	Type of wdsysmon subevent.
node	Name of the node that the POSIX process is on.
procname	POSIX process name for this subevent.
pid	POSIX process ID for this subevent. **Note** The three preceding fields describe the process whose CPU utilization is being monitored.
value	If the sec and msec variables are specified as 0 or are unspecified in the event registration Tcl command extension, the process CPU utilization is in the latest sample. If a time window is specified and is greater than zero in the event registration Tcl command extension, the averaged process CPU utilization is in the given time window.

Subevent Type	Description
secmsec	If the sec and msec variables are specified as 0 or are unspecified in the event registration Tcl command extension, they are both 0. If a time window is specified and is greater than zero in the event registration Tcl command extension, the sec and msec variables are the actual time difference between the time stamps of the oldest and latest samples in this time window.

For cpu_tot Subevent

```
"{type %s node {%s} value %u sec %ld msec %ld}"
```

Subevent Type	Description
type	Type of wdsysmon subevent.
node	Name of the node on which the total CPU utilization is being monitored.
value	If the sec and msec variables are specified as 0 or are unspecified in the event registration Tcl command extension, the total CPU utilization is in the latest sample. If a time window is specified and is greater than zero in the event registration Tcl command extension, the averaged total CPU utilization is in the given time window.
secmsec	If the sec and msec variables are specified as 0 or are unspecified in the event registration Tcl command extension, they are both 0. If a time window is specified and is greater than zero in the event registration Tcl command extension, the sec and msec variables are the actual time difference between the time stamps of the oldest and latest samples in this time window.

For mem_proc Subevent

```
"{type %s node {%s} procname {%s} pid %u is_percent %s value %u diff %d sec %ld msec %ld}"
```

Subevent Type	Description
type	Type of wdsysmon subevent.
node	Name of the node that the POSIX process is on.
procname	POSIX process name for this subevent.
pid	POSIX process ID for this subevent. **Note** The three preceding fields describe the process whose memory usage is being monitored.
is_percent	Can be either TRUE or FALSE. TRUE means that the value is a percentage value; FALSE means that the value is an absolute value (may be an averaged value).

Subevent Type	Description
value	If the sec and msec variables are specified as 0 or are unspecified in the event registration Tcl command extension, the process used memory is in the latest sample. If a time window is specified and is greater than zero in the event registration Tcl command extension, the averaged process used memory utilization is in the given time window.
Subevent Type	**Description**
diff	If the sec and msec variables are specified as 0 or are unspecified in the event registration Tcl command extension, the diff is the percentage difference between the first process used memory sample ever collected and the latest process used memory sample. If a time window is specified and is greater than zero in the event registration Tcl command extension, the diff is the percentage difference between the oldest and latest process used memory utilization in the specified time window.
secmsec	If the sec and msec variables are specified as 0 or are unspecified in the event registration Tcl command extension, they are both 0. If a time window is specified and is greater than zero in the event registration Tcl command extension, the sec and msec variables are the actual time difference between the time stamps of the oldest and latest samples in this time window.

If the *is_percent* argument is FALSE, and the *sec* and *msec* arguments are specified as 0 or are unspecified in the event registration Tcl command extension:

- *value* is the process used memory in the latest sample.
- *diff* is 0.
- *sec* and *msec* are both 0.

If the *is_percent* argument is FALSE, and a time window is specified as greater than zero in the event registration Tcl command extension:

- *value* is the averaged process used memory sample value in the specified time window.
- *diff* is 0.
- *sec* and *msec* are both the actual time difference between the time stamps of the oldest and latest samples in this time window.

If the *is_percent* argument is TRUE, and a time window is specified as greater than zero in the event registration Tcl command extension:

- value is 0.
- *diff* is the percentage difference between the oldest and latest process used memory samples in the specified time window.
- *sec* and **msec** are the actual time difference between the time stamps of the oldest and latest process used memory samples in this time window.

If the *is_percent* argument is TRUE, and the *sec* and *msec* arguments are specified as 0 or are unspecified in the event registration Tcl command extension:

- *value* is 0.

- *diff* is the percentage difference between the first process used memory sample ever collected and the latest process used memory sample.
- *sec* and *msec* are the actual time difference between the time stamps of the first process used memory sample ever collected and the latest process used memory sample.

For mem_tot_avail Subevent

```
"{type %s node {%s} is_percent %s used %u avail %u diff %d sec %ld msec %ld}"
```

Subevent Type	Description
type	Type of wdsysmon subevent.
node	Name of the node for which the total available memory is being monitored.
is_percent	Can be either TRUE or FALSE. TRUE means that the value is a percentage value; FALSE means that the value is an absolute value (may be an averaged value).
used	If the sec and msec variables are specified as 0 or are unspecified in the event registration Tcl command extension, the total used memory is in the latest sample. If a time window is specified and is greater than zero in the event registration Tcl command extension, the averaged total used memory utilization is in the given time window.
avail	If the sec and msec variables are specified as 0 or are unspecified in the event registration Tcl command extension, the avail is in the latest total available memory sample. If a time window is specified and is greater than zero in the event registration Tcl command extension, the avail is the total available memory utilization in the specified time window.
diff	If the sec and msec variables are specified as 0 or are unspecified in the event registration Tcl command extension, the diff is the percentage difference between the first total available memory sample ever collected and the latest total available memory sample. If a time window is specified and is greater than zero in the event registration Tcl command extension, the diff is the percentage difference between the oldest and latest total available memory utilization in the specified time window.
secmsec	If the sec and msec variables are specified as 0 or are unspecified in the event registration Tcl command extension, they are both 0. If a time window is specified and is greater than zero in the event registration Tcl command extension, they are the actual time difference between the time stamps of the oldest and latest samples in this time window.

If the *is_percent* argument is FALSE, and the sec and msec arguments are specified as 0 or are unspecified in the event registration Tcl command extension:

- *used* is the total used memory in the latest sample.
- *avail* is the total available memory in the latest sample.
- *diff* is 0.
- *sec* and **msec** are both 0.

If the *is_percent* argument is FALSE, and a time window is specified as greater than zero in the event registration Tcl command extension:

- *used* is 0.
- *avail* is the averaged total available memory sample value in the specified time window.
- *diff* is 0.
- *sec* and *msec* are both the actual time difference between the time stamps of the oldest and latest total available memory samples in this time window.

If the *is_percent* argument is TRUE, and a time window is specified as greater than zero in the event registration Tcl command extension:

- *used* is 0.
- *avail* is 0.
- *diff* is the percentage difference between the oldest and latest total available memory samples in the specified time window.
- *sec* and *msec* are both the actual time difference between the time stamps of the oldest and latest total available memory samples in this time window.

If the *is_percent* argument is TRUE, and the *sec* and *msec* arguments are specified as 0 or are unspecified in the event registration Tcl command extension:

- *used* is 0.
- *avail* is 0.
- *diff* is the percentage difference between the first total available memory sample ever collected and the latest total available memory sample.
- *sec* and msec are the actual time difference between the time stamps of the first total available memory sample ever collected and the latest total available memory sample.

For mem_tot_used Subevent

```
"{type %s node {%s} is_percent %s used %u avail %u diff %d sec %ld msec %ld}"
```

Subevent Type	Description
type	Type of wdsysmon subevent.
node	Name of the node for which the total used memory is being monitored.
is_percent	Can be either TRUE or FALSE. TRUE means that the value is a percentage value; FALSE means that the value is an absolute value (may be an averaged value).

Subevent Type	Description
used	If the sec and msec variables are specified as 0 or are unspecified in the event registration Tcl command extension, the total used memory is in the latest sample. If a time window is specified and is greater than zero in the event registration Tcl command extension, the averaged total used memory utilization is in the given time window.
avail	If the sec and msec variables are specified as 0 or are unspecified in the event registration Tcl command extension, the avail is in the latest total used memory sample. If a time window is specified and is greater than zero in the event registration Tcl command extension, the avail is the total used memory utilization in the specified time window.
diff	If the sec and msec variables are specified as 0 or are unspecified in the event registration Tcl command extension, the diff is the percentage difference between the first total used memory sample ever collected and the latest total used memory sample. If a time window is specified and is greater than zero in the event registration Tcl command extension, the diff is the percentage difference between the oldest and latest total used memory utilization in the specified time window.
secmsec	If the sec and msec variables are specified as 0 or are unspecified in the event registration Tcl command extension, they are both 0. If a time window is specified and is greater than zero in the event registration Tcl command extension, the sec and msec variables are the actual time difference between the time stamps of the oldest and latest samples in this time window.

If the *is_percent* argument is FALSE, and the *sec* and *msec* arguments are specified as 0 or are unspecified in the event registration Tcl command extension:

- *used* is the total used memory in the latest sample,
- *avail* is the total available memory in the latest sample,
- *diff* is 0,
- *sec* and *msec* are both 0,

If the *is_percent* argument is FALSE, and a time window is specified as greater than zero in the event registration Tcl command extension:

- *used* is the averaged total used memory sample value in the specified time window,
- *avail* is 0,
- *diff* is 0,
- *sec* and *msec* are both the actual time difference between the time stamps of the oldest and latest total used memory samples in this time window,

If the *is_percent* argument is TRUE, and a time window is specified as greater than zero in the event registration Tcl command extension:

- *used* is 0.
- *avail* is 0.

- *diff* is the percentage difference between the oldest and latest total used memory samples in the specified time window.
- *sec* and *msec* are both the actual time difference between the time stamps of the oldest and latest total used memory samples in this time window.

If the *is_percent* argument is TRUE, and the sec and msec arguments are specified as 0 or are unspecified in the event registration Tcl command extension:

- *used* is 0.
- *avail* is 0.
- *diff* is the percentage difference between the first total used memory sample ever collected and the latest total used memory sample.
- *sec* and *msec* are the actual time difference between the time stamps of the first total used memory sample ever collected and the latest total used memory sample.

Set _cerrno

Yes

Embedded Event Manager Action Tcl Command Extensions

action_process

Starts, restarts, or kills a Software Modularity process. This Tcl command extension is supported only in Software Modularity images.

Syntax

```
action_process start|restart|kill [job_id ?]
[process_name ?] [instance ?]
```

Arguments

start	(Mandatory) Specifies that a process is to be started.
restart	(Mandatory) Specifies that a process is to be restarted.
kill	(Mandatory) Specifies that a process is to be stopped (killed).
job_id	(Optional) System manager assigned job ID for the process. If you specify this argument, it must be an integer between 1 and 4294967295, inclusive.
process_name	(Optional) Process name. Either job_id must be specified or process_name and instance must be specified.
instance	(Optional) Process instance ID. If you specify this argument, it must be an integer between 1 and 4294967295, inclusive.

Result String

None

Set _cerrno

Yes

```
(_cerr_sub_err = 14)    FH_ENOSUCHACTION  (unknown action type)
```

This error means that the action command requested was unknown.

```
(_cerr_sub_num = 425, _cerr_sub_err = 1) SYSMGR_ERROR_INVALID_ARGS  (Invalid arguments
passed)
```

This error means that the arguments passed in were invalid.

```
(_cerr_sub_num = 425, _cerr_sub_err = 2) SYSMGR_ERROR_NO_MEMORY  (Could not allocate required
 memory)
```

This error means that an internal SYSMGR request for memory failed.

```
(_cerr_sub_num = 425, _cerr_sub_err = 5) SYSMGR_ERROR_NO_MATCH  (This process is not known
 to sysmgr)
```

This error means that the process name was not known.

```
(_cerr_sub_num = 425, _cerr_sub_err = 14) SYSMGR_ERROR_TOO_BIG  (outside the valid limit)
```

This error means that an object size exceeded its maximum.

```
(_cerr_sub_num = 425, _cerr_sub_err = 15) SYSMGR_ERROR_INVALID_OP  (Invalid operation for
this process)
```

This error means that the operation was invalid for the process.

action_program

Allows a Tcl script to run a POSIX process (program), optionally with a given argument string, environment string, Standard Input (stdin) pathname, Standard Output (stdout) pathname, or Standard Error (stderr) pathname. This Tcl command extension is supported only in Software Modularity images.

Syntax

```
action_program path ? [argv ?] [envp ?] [stdin ?] [stdout ?] [stderr ?]
```

Arguments

path	(Mandatory) Pathname of a program to run.

argv	(Optional) Argument string of the program.
envp	(Optional) Environment string of the program.
stdin	(Optional) Pathname for stdin.
stdout	(Optional) Pathname for stdout.
stderr	(Optional) Pathname for stderr.

Result String

None

Set _cerrno

Yes

```
(_cerr_sub_err =  2)    FH_ESYSERR  (generic/unknown error from OS/system)
```

This error means that the operating system reported an error. The POSIX errno value that is reported with the error should be used to determine the cause of the operating system error.

```
(_cerr_sub_err = 14)    FH_ENOSUCHACTION  (unknown action type)
```

This error means that the action command requested was unknown.

```
(_cerr_sub_err = 34)    FH_EMAXLEN  (maximum length exceeded)
```

This error means that the object length or number exceeded the maximum.

action_script

Allows a Tcl script to enable or disable the execution of all Tcl scripts (enables or disables the script scheduler).

Syntax

```
action_script [status enable|disable]
```

Arguments

status	(Optional) Flag to indicate script execution status. If this argument is set to enable, script execution is enabled; if this argument is set to disable, script execution is disabled.

Result String

None

Set _cerrno

Yes

```
(_cerr_sub_err =  2)    FH_ESYSERR  (generic/unknown error from OS/system)
```

This error means that the operating system reported an error. The POSIX errno value that is reported with the error should be used to determine the cause of the operating system error.

```
(_cerr_sub_err = 14)    FH_ENOSUCHACTION  (unknown action type)
```

This error means that the action command requested was unknown.

```
(_cerr_sub_err = 52)    FH_ECONFIG  (configuration error)
```

This error means that a configuration error has occurred.

action_setnode

Switches to the given node to enable subsequent EEM commands to be performed on that node. The following EEM commands use action_setnode to set their target node:

- action_process
- sys_reqinfo_proc
- sys_reqinfo_proc_all
- sys_reqinfo_crash_history
- sys_reqinfo_proc_version

Syntax

```
action_setnode [node ?]
```

Arguments

node	(Mandatory) Name of the node.

Result String

None

Set _cerrno

Yes

action_syslog

Logs a message.

Syntax

```
action_syslog [priority emerg|alert|crit|err|warning|notice|info|debug]
[msg ?]
```

Arguments

priority	(Optional) Action_syslog message facility level. If this argument is not specified, the default priority is LOG_INFO.
msg	(Optional) Message to be logged.

Result String

None

Set _cerrno

Yes

```
(_cerr_sub_err = 14)    FH_ENOSUCHACTION  (unknown action type)
```

This error means that the action command requested was unknown.

Embedded Event Manager Utility Tcl Command Extensions

appl_read

Reads Embedded Event Manager (EEM) application volatile data. This Tcl command extension provides support for reading EEM application volatile data. EEM application volatile data can be published by a Cisco IOS XR7 software process that uses the EEM application publish API. EEM application volatile data cannot be published by an EEM policy.

Note Currently there are no Cisco IOS XR software processes that publish application volatile data.

Syntax

```
appl_read name ? length ?
```

Arguments

name	(Mandatory) Name of the application published string data.
length	(Mandatory) Length of the string data to read. Must be an integer number between 1 and 4294967295, inclusive.

Result String

```
data %s
```

Where data is the application published string data to be read.

Set _cerrno

Yes

```
(_cerr_sub_err =  2)    FH_ESYSERR  (generic/unknown error from OS/system)
```

This error means that the operating system reported an error. The POSIX errno value that is reported with the error should be used to determine the cause of the operating system error.

```
(_cerr_sub_err =  7)    FH_ENOSUCHKEY  (could not find key)
```

This error means that the application event detector info key or other ID was not found.

```
(_cerr_sub_err =  9)    FH_EMEMORY  (insufficient memory for request)
```

This error means that an internal EEM request for memory failed.

appl_reqinfo

Retrieves previously saved information from the Embedded Event Manager (EEM). This Tcl command extension provides support for retrieving information from EEM that has been previously saved with a unique key, which must be specified in order to retrieve the information. Note that retrieving the information deletes it from EEM. It must be resaved if it is to be retrieved again.

Syntax

```
appl_reqinfo key ?
```

Arguments

key	(Mandatory) String key of the data.

Result String

```
data %s
```

Where data is the application string data to be retrieved.

Set _cerrno

Yes

```
(_cerr_sub_err =  2)    FH_ESYSERR  (generic/unknown error from OS/system)
```

This error means that the operating system reported an error. The POSIX errno value that is reported with the error should be used to determine the cause of the operating system error.

```
(_cerr_sub_err =  7)    FH_ENOSUCHKEY  (could not find key)
```

This error means that the application event detector info key or other ID was not found.

appl_setinfo

Saves information in the EEM. This Tcl command extension provides support for saving information in the EEM that can be retrieved later by the same policy or by another policy. A unique key must be specified. This key allows the information to be retrieved later.

Syntax

```
appl_setinfo key ? data ?
```

Arguments

key	(Mandatory) String key of the data.
data	(Mandatory) Application string data to save.

Result String

None

Set _cerrno

Yes

```
(_cerr_sub_err =  2)    FH_ESYSERR  (generic/unknown error from OS/system)
```

This error means that the operating system reported an error. The POSIX errno value that is reported with the error should be used to determine the cause of the operating system error.

```
(_cerr_sub_err =  8)    FH_EDUPLICATEKEY  (duplicate appl info key)
```

This error means that the application event detector info key or other ID was a duplicate.

```
(_cerr_sub_err =  9)    FH_EMEMORY  (insufficient memory for request)
```

This error means that an internal EEM request for memory failed.

```
(_cerr_sub_err = 34)    FH_EMAXLEN  (maximum length exceeded)
```

This error means that the object length or number exceeded the maximum.

```
(_cerr_sub_err = 43)    FH_EBADLENGTH  (bad API length)
```

This error means that the API message length was invalid.

counter_modify

Modifies a counter value.

Syntax

```
counter_modify event_id ? val ? op nop|set|inc|dec
```

Arguments

event_id	(Mandatory) Counter event ID returned by the **register_counter** Tcl command extension. Must be an integer between 0 and 4294967295, inclusive.
val	(Mandatory) • If op is set, this argument represents the counter value that is to be set. • If op is inc, this argument is the value by which to increment the counter. • If op is dec, this argument is the value by which to decrement the counter.
op	(Mandatory) • nop—Retrieves the current counter value. • set—Sets the counter value to the given value. • inc—Increments the counter value by the given value. • dec—Decrements the counter value by the given value.

Result String

```
val_remain %d
```

Where val_remain is the current value of the counter.

Set _cerrno

Yes

```
(_cerr_sub_err =  2)    FH_ESYSERR  (generic/unknown error from OS/system)
```

This error means that the operating system reported an error. The POSIX errno value that is reported with the error should be used to determine the cause of the operating system error.

```
(_cerr_sub_err = 11)    FH_ENOSUCHESID  (unknown event specification ID)
```

This error means that the event specification ID could not be matched when the event was being registered or that an event detector internal event structure is corrupt.

```
(_cerr_sub_err = 22)    FH_ENULLPTR  (event detector internal error - ptr is null)
```

This error means that an internal EEM event detector pointer was null when it should have contained a value.

```
(_cerr_sub_err = 30)    FH_ECTBADOPER  (bad counter threshold operator)
```

This error means that the counter event detector set or modify operator was invalid.

timer_arm

Arms a timer. The type could be CRON, watchdog, countdown, or absolute.

Syntax

```
timer_arm event_id ? cron_entry ?|time ?
```

Arguments

event_id	(Mandatory)Timer event ID returned by the **register_timer** command extension. Must be an integer between 0 and 4294967295, inclusive.
cron_entry	(Mandatory) Must exist if the timer type is CRON. Must not exist for other types of timer. CRON timer specification uses the format of the CRON table entry.
time	(Mandatory) Must exist if the timer type is not CRON. Must not exist if the timer type is CRON. For watchdog and countdown timers, the number of seconds and milliseconds until the timer expires; for an absolute timer, the calendar time of the expiration time (specified in SSSSSSSSSS[.MMM] format, where SSSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). An absolute expiration date is the number of seconds and milliseconds since January 1, 1970. If the date specified has already passed, the timer expires immediately.

Result String

```
sec_remain %ld msec_remain %ld
```

Where sec_remain and msec_remain are the remaining time before the next expiration of the timer.

Note A value of 0 is returned for the sec_remain and msec_remain arguments if the timer type is CRON.

Set _cerrno

Yes

```
(_cerr_sub_err =  2)    FH_ESYSERR  (generic/unknown error from OS/system)
```

This error means that the operating system reported an error. The POSIX errno value that is reported with the error should be used to determine the cause of the operating system error.

```
(_cerr_sub_err =  6)    FH_EBADEVENTTYPE  (unknown EEM event type)
```

This error means that the event type specified in the internal event specification was invalid.

```
(_cerr_sub_err =  9)    FH_EMEMORY  (insufficient memory for request)
```

This error means that an internal EEM request for memory failed.

```
(_cerr_sub_err = 11)    FH_ENOSUCHESID  (unknown event specification ID)
```

This error means that the event specification ID could not be matched when the event was being registered or that an event detector internal event structure is corrupt.

```
(_cerr_sub_err = 12)    FH_ENOSUCHEID  (unknown event ID)
```

This error means that the event ID could not be matched when the event was being registered or that an event detector internal event structure is corrupt.

```
(_cerr_sub_err = 22)    FH_ENULLPTR  (event detector internal error - ptr is null)
```

This error means that an internal EEM event detector pointer was null when it should have contained a value.

```
(_cerr_sub_err = 27)    FH_ETMDELAYZR  (zero delay time)
```

This error means that the time specified to arm a timer was zero.

```
(_cerr_sub_err = 42)    FH_ENOTREGISTERED  (request for event spec that is unregistered)
```

This error means that the event was not registered.

```
(_cerr_sub_err = 54)    FH_EFDUNAVAIL  (connection to event detector unavailable)
```

This error means that the event detector was unavailable.

```
(_cerr_sub_err = 56)    FH_EFDCONNERR  (event detector connection error)
```

This error means that the EEM event detector that handles this request is not available.

timer_cancel

Cancels a timer.

Syntax

```
timer_cancel event_id ?
```

Arguments

event_id	(Mandatory) Timer event ID returned by the **register_timer** command extension. Must be an integer between 0 and 4294967295, inclusive.

Result String

```
sec_remain %ld msec_remain %ld
```

Where sec_remain and msec_remain are the remaining time before the next expiration of the timer.

Note A value of 0 will be returned for sec_remain and msec_remain if the timer type is CRON.

Set _cerrno

Yes

```
(_cerr_sub_err =  2)    FH_ESYSERR  (generic/unknown error from OS/system)
```

This error means that the operating system reported an error. The POSIX errno value that is reported with the error should be used to determine the cause of the operating system error.

```
(_cerr_sub_err =  6)    FH_EBADEVENTTYPE  (unknown EEM event type)
```

This error means that the event type specified in the internal event specification was invalid.

```
(_cerr_sub_err =  7)    FH_ENOSUCHKEY  (could not find key)
```

This error means that the application event detector info key or other ID was not found.

```
(_cerr_sub_err = 11)    FH_ENOSUCHESID  (unknown event specification ID)
```

This error means that the event specification ID could not be matched when the event was being registered or that an event detector internal event structure is corrupt.

```
(_cerr_sub_err = 12)    FH_ENOSUCHEID  (unknown event ID)
```

This error means that the event ID could not be matched when the event was being registered or that an event detector internal event structure is corrupt.

```
(_cerr_sub_err = 22)    FH_ENULLPTR  (event detector internal error - ptr is null)
```

This error means that an internal EEM event detector pointer was null when it should have contained a value.

```
(_cerr_sub_err = 54)    FH_EFDUNAVAIL  (connection to event detector unavailable)
```

This error means that the event detector was unavailable.

```
(_cerr_sub_err = 56)    FH_EFDCONNERR  (event detector connection error)
```

This error means that the EEM event detector that handles this request is not available.

Embedded Event Manager System Information Tcl Command Extensions

Note All EEM system information commands—**sys_reqinfo _xxx**—have the Set _cerrno section set to **yes**.

sys_reqinfo_cpu_all

Queries the CPU utilization of the top processes (both POSIX processes and IOS processes) during a specified time period and in a specified order. This Tcl command extension is supported only in Software Modularity images.

Syntax

```
sys_reqinfo_cpu_all order cpu_used [sec ?] [msec ?] [num ?]
```

Arguments

order	(Mandatory) Order used for sorting the CPU utilization of processes.
cpu_used	(Mandatory) Specifies that the average CPU utilization, for the specified time window, will be sorted in descending order.
secmsec	(Optional) Time period, in seconds and milliseconds, during which the average CPU utilization is calculated. Must be integers in the range from 0 to 4294967295. If not specified, or if both sec and msec are specified as 0, the most recent CPU sample is used.
num	(Optional) Number of entries from the top of the sorted list of processes to be displayed. Must be an integer in the range from 1 to 4294967295. Default value is 5.

Result String

```
rec_list {{process CPU info string 0},{process CPU info string 1}, ...}
```

Where each process CPU info string is:

```
pid %u name {%s} cpu_used %u
```

rec_list	Marks the start of the process CPU information list.
pid	Process ID.
name	Process name.
cpu_used	Specifies that if sec and msec are specified with a number greater than zero, the average percentage is calculated from the process CPU utilization during the specified time period. If sec and msec are both zero or not specified, the average percentage is calculated from the process CPU utilization in the latest sample.

Set _cerrno

Yes

sys_reqinfo_crash_history

Queries the crash information of all processes that have ever crashed. This Tcl command extension is supported only in Software Modularity images.

Syntax

```
sys_reqinfo_crash_history
```

Arguments

None

Result String

```
rec_list {{crash info string 0},{crash info string 1}, ...}

Where each crash info string is:

job_id %u name {%s} respawn_count %u fail_count %u dump_count %u
inst_id %d exit_status 0x%x exit_type %d proc_state {%s} component_id 0x%x
crash_time_sec %ld crash_time_msec %ld
```

job_id	System manager assigned job ID for the process. An integer between 1 and 4294967295, inclusive.
name	Process name.
respawn_count	Total number of restarts for the process.
fail_count	Number of restart attempts of the process. This count is reset to zero when the process is successfully restarted.
dump_count	Number of core dumps performed.
inst_id	Process instance ID.

exit_status	Last exit status of the process.
exit_type	Last exit type.
proc_state	Sysmgr process states. One of the following: error, forced_stop, hold, init, ready_to_run, run, run_rnode, stop, waitEOltimer, wait_rnode, wait_spawntimer, wait_tpl.
component_id	Version manager assigned component ID for the component to which the process belongs.
crash_time_sec crash_time_msec	Seconds and milliseconds since January 1, 1970, which represent the last time the process crashed.

Set _cerrno

Yes

sys_reqinfo_mem_all

Queries the memory usage of the top processes (both POSIX and IOS) during a specified time period and in a specified order. This Tcl command extension is supported only in Software Modularity images.

Syntax

```
sys_reqinfo_mem_all order allocates|increase|used [sec ?] [msec ?] [num ?]
```

Arguments

order	(Mandatory) Order used for sorting the memory usage of processes.
allocates	(Mandatory) Specifies that the memory usage is sorted by the number of process allocations during the specified time window, and in descending order.
increase	(Mandatory) Specifies that the memory usage is sorted by the percentage of process memory increase during the specified time window, and in descending order.
used	(Mandatory) Specifies that the memory usage is sorted by the current memory used by the process.
secmsec	(Optional) Time period, in seconds and milliseconds, during which the process memory usage is calculated. Must be integers in the range from 0 to 4294967295. If both sec and msec are specified and are nonzero, the number of allocations is the difference between the number of allocations in the oldest and latest samples collected in the time period. The percentage is calculated as the the percentage difference between the memory used in the oldest and latest samples collected in the time period. If not specified, or if both sec and msec are specified as 0, the first sample ever collected is used as the oldest sample; that is, the time period is set to be the time from startup until the current moment.
num	(Optional) Number of entries from the top of the sorted list of processes to be displayed. Must be an integer in the range from 1 to 4294967295. Default value is 5.

Result String

```
rec_list {{process mem info string 0},{process mem info string 1}, ...}
```

Where each process mem info string is:

```
pid %u name {%s} delta_allocs %d initial_alloc %u current_alloc %u percent_increase %d
```

rec_list	Marks the start of the process memory usage information list.
pid	Process ID.
name	Process name.
delta_allocs	Specifies the difference between the number of allocations in the oldest and latest samples collected in the time period.
initial_alloc	Specifies the amount of memory, in kilobytes, used by the process at the start of the time period.
current_alloc	Specifies the amount of memory, in kilobytes, currently used by the process.
percent_increase	Specifies the percentage difference between the memory used in the oldest and latest samples collected in the time period. The percentage difference can be expressed as current_alloc minus initial_alloc times 100 and divided by initial_alloc.

Set _cerrno

Yes

sys_reqinfo_proc

Queries the information about a single POSIX process. This Tcl command extension is supported only in Software Modularity images.

Syntax

```
sys_reqinfo_proc job_id ?
```

Arguments

job_id	(Mandatory) System manager assigned job ID for the process. Must be an integer between 1 and 4294967295, inclusive.

Result String

```
job_id %u component_id 0x%x name {%s} helper_name {%s} helper_path {%s} path {%s}
node_name {%s} is_respawn %u is_mandatory %u is_hold %u dump_option %d
max_dump_count %u respawn_count %u fail_count %u dump_count %u
last_respawn_sec %ld last_respawn_msec %ld inst_id %u proc_state %s
```

```
level %d exit_status 0x%x exit_type %d
```

job_id	System manager assigned job ID for the process. An integer between 1 and 4294967295, inclusive.
component_id	Version manager assigned component ID for the component to which the process belongs.
name	Process name.
helper_name	Helper process name.
helper_path	Executable path of the helper process.
path	Executable path of the process.
node_name	System manager assigned node name for the node to which the process belongs.
is_respawn	Flag that specifies that the process can be respawned.
is_mandatory	Flag that specifies that the process must be alive.
is_hold	Flag that specifies that the process is spawned until called by the API.
dump_option	Core dumping options.
max_dump_count	Maximum number of core dumping permitted.
respawn_count	Total number of restarts for the process.
fail_count	Number of restart attempts of the process. This count is reset to zero when the process is successfully restarted.
dump_count	Number of core dumps performed.
last_respawn_seclast_respawn_msec	Seconds and milliseconds in POSIX timer units since January 1, 1970, which represent the last time the process was started.
inst_id	Process instance ID.
proc_state	Sysmgr process states. One of the following: error, forced_stop, hold, init, ready_to_run, run, run_rnode, stop, waitEOltimer, wait_rnode, wait_spawntimer, wait_tpl.
level	Process run level.
exit_status	Last exit status of the process.
exit_type	Last exit type.

Set _cerrno

Yes

sys_reqinfo_proc_all

Queries the information of all POSIX processes. This Tcl command extension is supported only in Software Modularity images.

Syntax

```
sys_reqinfo_proc_all
```

Arguments

None

Result String

```
rec_list {{process info string 0}, {process info string 1},...}
```

Where each process info string is the same as the result string of the **sysreq_info_proc** Tcl command extension.

Set _cerrno

Yes

sys_reqinfo_proc_version

Queries the version of the given process.

Syntax

```
sys_reqinfo_proc_version [job_id ?]
```

Arguments

job_id	(Mandatory) System manager assigned job ID for the process. The integer number must be inclusively between 1 and 2147483647.

Result String

```
version_id %02d.%02d.%04d
```

Where version_id is the version manager that is assigned the version number of the process.

Set _cerrno

Yes

sys_reqinfo_routername

Queries the router name.

Syntax

```
sys_reqinfo_routername
```

Arguments

None

Result String

```
routername %s
```

Where routername is the name of the router.

Set _cerrno

Yes

sys_reqinfo_syslog_freq

Queries the frequency information of all syslog events.

Syntax

```
sys_reqinfo_syslog_freq
```

Arguments

None

Result String

```
rec_list {{event frequency string 0}, {log freq str 1}, ...}
```

Where each event frequency string is:

```
time_sec %ld time_msec %ld match_count %u raise_count %u occurs %u
period_sec %ld period_msec %ld pattern {%s}
```

time_sectime_msec	Seconds and milliseconds in POSIX timer units since January 1, 1970, which represent the time the last event was raised.
match_count	Number of times that a syslog message matches the pattern specified by this syslog event specification since event registration.
raise_count	Number of times that this syslog event was raised.
occurs	Number of occurrences needed in order to raise the event; if not specified, the event is raised on the first occurrence.
period_secperiod_msec	Number of occurrences must occur within this number of POSIX timer units in order to raise the event; if not specified, the period check does not apply.

pattern	Regular expression used to perform syslog message pattern matching.

Set _cerrno

Yes

```
(_cerr_sub_err =  2)    FH_ESYSERR  (generic/unknown error from OS/system)
```

This error means that the operating system reported an error. The POSIX errno value that is reported with the error should be used to determine the cause of the operating system error.

```
(_cerr_sub_err =  9)    FH_EMEMORY  (insufficient memory for request)
```

This error means that an internal EEM request for memory failed.

```
(_cerr_sub_err = 22)    FH_ENULLPTR  (event detector internal error - ptr is null)
```

This error means that an internal EEM event detector pointer was null when it should have contained a value.

```
(_cerr_sub_err = 45)    FH_ESEQNUM  (sequence or workset number out of sync)
```

This error means that the event detector sequence or workset number was invalid.

```
(_cerr_sub_err = 46)    FH_EREGEMPTY  (registration list is empty)
```

This error means that the event detector registration list was empty.

```
(_cerr_sub_err = 54)    FH_EFDUNAVAIL  (connection to event detector unavailable)
```

This error means that the event detector was unavailable.

sys_reqinfo_syslog_history

Queries the history of the specified syslog message.

Syntax

```
sys_reqinfo_syslog_history
```

Arguments

None

Result String

```
rec_list {{log hist string 0}, {log hist str 1}, ...}
```

Where each log hist string is:

```
time_sec %ld time_msec %ld msg {%s}
```

time_sec time_msec	Seconds and milliseconds since January 1, 1970, which represent the time the message was logged.
msg	Syslog message.

Set _cerrno

Yes

```
(_cerr_sub_err =  2)    FH_ESYSERR  (generic/unknown error from OS/system)
```

This error means that the operating system reported an error. The POSIX errno value that is reported with the error should be used to determine the cause of the operating system error.

```
(_cerr_sub_err = 22)    FH_ENULLPTR  (event detector internal error - ptr is null)
```

This error means that an internal EEM event detector pointer was null when it should have contained a value.

```
(_cerr_sub_err = 44)    FH_EHISTEMPTY  (history list is empty)
```

This error means that the history list was empty.

```
(_cerr_sub_err = 45)    FH_ESEQNUM  (sequence or workset number out of sync)
```

This error means that the event detector sequence or workset number was invalid.

```
(_cerr_sub_err = 54)    FH_EFDUNAVAIL  (connection to event detector unavailable)
```

This error means that the event detector was unavailable.

sys_reqinfo_stat

Queries the value of the statistic entity that is specified by name, and optionally the first modifier and the second modifier.

Syntax

```
sys_reqinfo_stat [name ?][mod1 ?][mod2 ?]
```

Arguments

name	(Mandatory) Statistics data element name.
mod_1	(Optional) Statistics data element modifier 1.

mod_2	(Optional) Statistics data element modifier 2.

Result String

```
name %s value %s
```

name	Statistics data element name.
value	Value string of the statistics data element.

Set _cerrno

Yes

sys_reqinfo_snmp

Queries the value of the entity specified by a Simple Network Management Protocol (SNMP) object ID.

Syntax

```
sys_reqinfo_snmp oid ? get_type exact|next
```

Arguments

oid	(Mandatory) SNMP OID in dot notation (for example, 1.3.6.1.2.1.2.1.0).
get_type	(Mandatory) Type of SNMP get operation that needs to be applied to the specified oid. If the get_type is "exact," the value of the specified oid is retrieved; if the get_type is "next," the value of the lexicographical successor to the specified oid is retrieved.

Result String

```
oid {%s} value {%s}
```

oid	SNMP OID.
value	Value string of the associated SNMP data element.

SMTP Library Command Extensions

All Simple Mail Transfer Protocol (SMTP) library command extensions belong to the ::cisco::lib namespace.

To use this library, the user needs to provide an e-mail template file. The template file can include Tcl global variables so that the e-mail service and the e-mail text can be configured through the **event manager environment** Cisco IOS XR7 software command-line interface (CLI) configuration command. There are commands in this library to substitute the global variables in the e-mail template file and to send the desired e-mail context with the To address, CC address, From address, and Subject line properly configured using the configured e-mail server.

E-Mail Template

The e-mail template file has the following format:

```
Mailservername:<space><the list of candidate SMTP server addresses>
From:<space><the e-mail address of sender>
To:<space><the list of e-mail addresses of recipients>
Cc:<space><the list of e-mail addresses that the e-mail will be copied to>
Subject:<subject line>
<a blank line>
<body>
```

Note The template normally includes Tcl global variables to be configured.

The following is a sample e-mail template file:

```
Mailservername: $_email_server
From: $_email_from
To: $_email_to
Cc: $_email_cc
Subject: From router $routername: Process terminated

process name: $process_name
subsystem: $sub_system
exit status: $exit_status
respawn count: $respawn_count
```

Exported Tcl Command Extensions

smtp_send_email

Given the text of an e-mail template file with all global variables already substituted, sends the e-mail out using Simple Mail Transfer Protocol (SMTP). The e-mail template specifies the candidate mail server addresses, To addresses, CC addresses, From address, subject line, and e-mail body.

Note A list of candidate e-mail servers can be provided so that the library will try to connect the servers on the list one by one until it can successfully connect to one of them.

Syntax

```
smtp_send_email text
```

Arguments

text	(Mandatory) Text of an e-mail template file with all global variables already substituted.

Result String

None

Set _cerrno

- Wrong 1st line format—Mailservername:list of server names.
- Wrong 2nd line format—From:from-address.
- Wrong 3rd line format—To:list of to-addresses.
- Wrong 4th line format—CC:list of cc-addresses.
- Error connecting to mail server:—$sock closed by remote server (where $sock is the name of the socket opened to the mail server).
- Error connecting to mail server:—$sock reply code is $k instead of the service ready greeting (where $sock is the name of the socket opened to the mail server; $k is the reply code of $sock).
- Error connecting to mail server:—cannot connect to all the candidate mail servers.
- Error disconnecting from mail server:—$sock closed by remote server (where $sock is the name of the socket opened to the mail server).

Sample Scripts

After all needed global variables in the e-mail template are defined:

```
if [catch {smtp_subst [file join $tcl_library email_template_sm]} result] {
    puts stderr $result
    exit 1
}
if [catch {smtp_send_email $result} result] {
    puts stderr $result
    exit 1
}
```

smtp_subst

Given an e-mail template file e-mail_template, substitutes each global variable in the file by its user-defined value. Returns the text of the file after substitution.

Syntax

```
smtp_subst e-mail_template
```

Arguments

e-mail_template	(Mandatory) Name of an e-mail template file in which global variables need to be substituted by a user-defined value. An example filename could be /disk0://example.template which represents a file named example.template in a top-level directory on an ATA flash disk in slot 0.

Result String

The text of the e-mail template file with all the global variables substituted.

Set _cerrno

- cannot open e-mail template file
- cannot close e-mail template file

CLI Library Command Extensions

All command-line interface (CLI) library command extensions belong to the ::cisco::eem namespace.

This library provides users the ability to run CLI commands and get the output of the commands in Tcl. Users can use commands in this library to spawn an exec and open a virtual terminal channel to it, write the command to execute to the channel so that the command will be executed by exec, and read back the output of the command.

There are two types of CLI commands: interactive commands and non-interactive commands.

For interactive commands, after the command is entered, there will be a "Q&A" phase in which the router will ask for different user options, and the user is supposed to enter the answer for each question. Only after all the questions have been answered properly will the command run according to the user's options until completion.

For noninteractive commands, once the command is entered, the command will run to completion. To run different types of commands using an EEM script, different CLI library command sequences should be used, which are documented in the Using the CLI Library to Run a Noninteractive Command, on page 143 and in the Using the CLI Library to Run an Interactive Command, on page 144.

Exported Tcl Command Extensions

cli_close

Closes the exec process and releases the VTY and the specified channel handler connected to the command-line interface (CLI).

Syntax

```
cli_close fd tty_id
```

Arguments

fd	(Mandatory) The CLI channel handler.
tty_id	(Mandatory) The TTY ID returned from the **cli_open** command extension.

Result String

None

Set _cerrno

Cannot close the channel.

cli_exec

Writes the command to the specified channel handler to execute the command. Then reads the output of the command from the channel and returns the output.

Syntax

```
cli_exec fd cmd
```

Arguments

fd	(Mandatory) The command-line interface (CLI) channel handler.
cmd	(Mandatory) The CLI command to execute.

Result String

The output of the CLI command executed.

Set _cerrno

Error reading the channel.

cli_get_ttyname

Returns the real and pseudo tty names for a given TTY ID.

Syntax

```
cli_get_ttyname tty_id
```

Arguments

tty_id	(Mandatory) The TTY ID returned from the **cli_open** command extension.

Result String

```
pty %s tty %s
```

Set _cerrno

None

cli_open

Allocates a vty, creates an EXEC command-line interface (CLI) session, and connects the vty to a channel handler. Returns an array including the channel handler.

Note Each call to **cli_open** initiates a Cisco IOS XR7 software EXEC session that allocates a Cisco IOS XR7 software vty. The vty remains in use until the cli_close routine is called. Vtys are allocated from the pool of vtys that are configured using the **line vty vty-pool** CLI configuration command. Be aware that the cli_open routine fails when two or fewer vtys are available, preserving the remaining vtys for Telnet use.

Syntax

```
cli_open
```

Arguments

None

Result String

```
"tty_id {%s} pty {%d} tty {%d} fd {%d}"
```

Event Type	Description
tty_id	TTY ID.
pty	PTY device name.
tty	TTY device name.
fd	CLI channel handler.

Set _cerrno

- Cannot get pty for EXEC.
- Cannot create an EXEC CLI session.
- Error reading the first prompt.

cli_read

Reads the command output from the specified command-line interface (CLI) channel handler until the pattern of the router prompt occurs in the contents read. Returns all the contents read up to the match.

Syntax

```
cli_read fd
```

Arguments

fd	(Mandatory) CLI channel handler.

Result String

All the contents read.

Set _cerrno

Cannot get router name.

Note This Tcl command extension blocks waiting for the router prompt to show up in the contents read.

cli_read_drain

Reads and drains the command output of the specified command-line interface (CLI) channel handler. Returns all the contents read.

Syntax

```
cli_read_drain fd
```

Arguments

fd	(Mandatory) The CLI channel handler.

Result String

All the contents read.

Set _cerrno

None

cli_read_line

Reads one line of the command output from the specified command-line interface (CLI) channel handler. Returns the line read.

Syntax

```
cli_read_line fd
```

Arguments

fd	(Mandatory) CLI channel handler.

Result String

The line read.

Set _cerrno

None

Note This Tcl command extension blocks waiting for the end of line to show up in the contents read.

cli_read_pattern

Reads the command output from the specified command-line interface (CLI) channel handler until the pattern that is to be matched occurs in the contents read. Returns all the contents read up to the match.

Note The pattern matching logic attempts a match by looking at the command output data as it is delivered from the Cisco IOS XR7 software command. The match is always done on the most recent 256 characters in the output buffer unless there are fewer characters available, in which case the match is done on fewer characters. If more than 256 characters in the output buffer are required for the match to succeed, the pattern will not match.

Syntax

```
cli_read_pattern fd ptn
```

Arguments

fd	(Mandatory) CLI channel handler.
ptn	(Mandatory) Pattern to be matched when reading the command output from the channel.

Result String

All the contents read.

Set _cerrno

None

Note This Tcl command extension blocks waiting for the specified pattern to show up in the contents read.

cli_write

Writes the command that is to be executed to the specified CLI channel handler. The CLI channel handler executes the command.

Syntax

```
cli_write fd cmd
```

Arguments

fd	(Mandatory) The CLI channel handler.
cmd	(Mandatory) The CLI command to execute.

Result String

None

Set _cerrno

None

Sample Usage

As an example, use configuration CLI commands to bring up Ethernet interface 1/0:

```
if [catch {cli_open} result] {
puts stderr $result
exit 1
} else {
array set cli1 $result
}
if [catch {cli_exec $cli1(fd) "config t"} result] {
puts stderr $result
exit 1
}
if [catch {cli_exec $cli1(fd) "interface Ethernet1/0"} result] {
puts stderr $result
exit 1
}
if [catch {cli_exec $cli1(fd) "no shut"} result] {
puts stderr $result
exit 1
}
if [catch {cli_exec $cli1(fd) "end"} result] {
puts stderr $result
exit 1
}
if [catch {cli_close $cli1(fd) $cli1(tty_id)} } result] {
puts stderr $result
exit 1
```

Using the CLI Library to Run a Noninteractive Command

To run a noninteractive command, use the **cli_exec** command extension to issue the command, and then wait for the complete output and the router prompt. For example, the following shows the use of configuration CLI commands to bring up Ethernet interface 1/0:

```
if [catch {cli_open} result] {
error $result $errorInfo
} else {
```

```
set fd $result
}
if [catch {cli_exec $fd "config t"} result] {
error $result $errorInfo
}
if [catch {cli_exec $fd "interface Ethernet1/0"} result] {
error $result $errorInfo
}
if [catch {cli_exec $fd "no shut"} result] {
error $result $errorInfo
}
if [catch {cli_exec $fd "end"} result] {
error $result $errorInfo
}
if [catch {cli_close $fd} result] {
error $result $errorInfo
}
```

Using the CLI Library to Run an Interactive Command

To run interactive commands, three phases are needed:

- Phase 1: Issue the command using the **cli_write** command extension.
- Phase 2: Q&A Phase. Use the **cli_read_pattern** command extension to read the question (the regular pattern that is specified to match the question text) and the **cli_write** command extension to write back the answers alternately.
- Phase 3: Noninteractive phase. All questions have been answered, and the command will run to completion. Use the **cli_read** command extension to wait for the complete output of the command and the router prompt.

For example, use CLI commands to do squeeze bootflash: and save the output of this command in the Tcl variable cmd_output.

```
if [catch {cli_open} result] {
error $result $errorInfo
} else {
array set cli1 $result
}

# Phase 1: issue the command
if [catch {cli_write $cli1(fd) "squeeze bootflash:"} result] {
error $result $errorInfo
}

# Phase 2: Q&A phase
# wait for prompted question:
# All deleted files will be removed. Continue? [confirm]
if [catch {cli_read_pattern $cli1(fd) "All deleted"} result] {
error $result $errorInfo
}
# write a newline character
if [catch {cli_write $cli1(fd) "\n"} result] {
error $result $errorInfo
}
# wait for prompted question:
# Squeeze operation may take a while. Continue? [confirm]
if [catch {cli_read_pattern $cli1(fd) "Squeeze operation"} result] {
error $result $errorInfo
}
```

```
# write a newline character
if [catch {cli_write $cli1(fd) "\n"} result] {
error $result $errorInfo
}

# Phase 3: noninteractive phase
# wait for command to complete and the router prompt
if [catch {cli_read $cli1(fd) } result] {
error $result $errorInfo
} else {
set cmd_output $result
}
if [catch {cli_close $cli1(fd) $cli1(tty_id)} result] {
error $result $errorInfo
}
```

The following example causes a router to be reloaded using the CLI **reload** command. Note that the EEM **action_reload** command accomplishes the same result in a more efficient manner, but this example is presented to illustrate the flexibility of the CLI library for interactive command execution.

```
# 1. execute the reload command
if [catch {cli_open} result] {
    error $result $errorInfo
} else {
    array set cli1 $result
}
if [catch {cli_write $cli1(fd) "reload"} result] {
    error $result $errorInfo
} else {
    set cmd_output $result
}
if [catch {cli_read_pattern $cli1(fd) ".*(System configuration has been modified. Save\\\?
 \\\[yes/no\\\]: )"} result] {
    error $result $errorInfo
} else {
    set cmd_output $result
}
if [catch {cli_write $cli1(fd) "no"} result] {
    error $result $errorInfo
} else {
    set cmd_output $result
}
if [catch {cli_read_pattern $cli1(fd) ".*(Proceed with reload\\\? \\\[confirm\\\])"} result]
 {
    error $result $errorInfo
} else {
    set cmd_output $result
}
if [catch {cli_write $cli1(fd) "y"} result] {
    error $result $errorInfo
} else {
    set cmd_output $result
}
if [catch {cli_close $cli1(fd) $cli1(tty_id)} result] {
    error $result $errorInfo
}
```

Tcl Context Library Command Extensions

All the Tcl context library command extensions belong to the ::cisco::eem namespace.

Exported Commands

context_retrieve

Retrieves Tcl variable(s) identified by the given context name, and possibly the scalar variable name, the array variable name, and the array index. Retrieved information is automatically deleted.

Note Once saved information is retrieved, it is automatically deleted. If that information is needed by another policy, the policy that retrieves it (using the **context_retrieve** command extension) should also save it again (using the **context_save** command extension).

Syntax

```
context_retrieve ctxt [var] [index_if_array]
```

Arguments

ctxt	(Mandatory) Context name.
var	(Optional) Scalar variable name or array variable name. Defaults to a null string if this argument is not specified.
index_if_array	(Optional) Array index.

Note The *index_if_array* argument is ignored when the *var* argument is a scalar variable.

If *var* is unspecified, retrieves the whole variable table saved in the context.

If *var* is specified and *index_if_array* is not specified, or if *index_if_array* is specified but *var* is a scalar variable, retrieves the value of *var*.

If *var* is specified, and *index_if_array* is specified, and *var* is an array variable, retrieves the value of the specified array element.

Result String

Resets the Tcl global variables to the state that they were in when the save was performed.

Set _cerrno

- A string displaying _cerrno, _cerr_sub_num, _cerr_sub_err, _cerr_posix_err, _cerr_str due to appl_reqinfo error.
- Variable is not in the context.

Sample Usage

The following examples show how to use the **context_save** and **context_retrieve** command extension functionality to save and retrieve data. The examples are shown in save and retrieve pairs.

Example 1: Save

If var is unspecified or if a pattern if specified, saves multiple variables to the context.

```
::cisco::eem::event_register_none

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

set testvara 123
set testvarb 345
set testvarc 789
if {[catch {context_save TESTCTX "testvar*"} errmsg]} {
      action_syslog msg "context_save failed: $errmsg"
} else {
      action_syslog msg "context_save succeeded"
}
```

Example 1: Retrieve

If var is unspecified, retrieves multiple variables from the context.

```
::cisco::eem::event_register_none

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

if {[catch {foreach {var value} [context_retrieve TESTCTX] {set $var $value}} errmsg]} {
      action_syslog msg "context_retrieve failed: $errmsg"
} else {
      action_syslog msg "context_retrieve succeeded"
}
if {[info exists testvara]} {
      action_syslog msg "testvara exists and is $testvara"
} else {
      action_syslog msg "testvara does not exist"
}
if {[info exists testvarb]} {
      action_syslog msg "testvarb exists and is $testvarb"
} else {
      action_syslog msg "testvarb does not exist"
}
if {[info exists testvarc]} {
      action_syslog msg "testvarc exists and is $testvarc"
} else {
      action_syslog msg "testvarc does not exist"
}
```

Example 2: Save

If var is specified, saves the value of var.

```
::cisco::eem::event_register_none
```

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

set testvar 123
if {[catch {context_save TESTCTX testvar} errmsg]} {
      action_syslog msg "context_save failed: $errmsg"
} else {
      action_syslog msg "context_save succeeded"
}
```

Example 2: Retrieve

If var is specified and index_if_array is not specified, or if index_if_array is specified but var is a scalar variable, retrieves the value of var.

```
::cisco::eem::event_register_none

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

if {[catch {set testvar [context_retrieve TESTCTX testvar]} errmsg]} {
      action_syslog msg "context_retrieve failed: $errmsg"
} else {
      action_syslog msg "context_retrieve succeeded"
}
if {[info exists testvar]} {
      action_syslog msg "testvar exists and is $testvar"
} else {
      action_syslog msg "testvar does not exist"
}
```

Example 3: Save

If var is specified, saves the value of var even if it is an array.

```
::cisco::eem::event_register_none

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

array set testvar "testvar1 ok testvar2 not_ok"
if {[catch {context_save TESTCTX testvar} errmsg]} {
      action_syslog msg "context_save failed: $errmsg"
} else {
      action_syslog msg "context_save succeeded"
}
```

Example 3: Retrieve

If var is specified, and index_if_array is not specified, and var is an array variable, retrieves the entire array.

```
::cisco::eem::event_register_none

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

```
if {[catch {array set testvar [context_retrieve TESTCTX testvar]} errmsg]} {
      action_syslog msg "context_retrieve failed: $errmsg"
} else {
      action_syslog msg "context_retrieve succeeded"
}
if {[info exists testvar]} {
      action_syslog msg "testvar exists and is [array get testvar]"
} else {
      action_syslog msg "testvar does not exist"
}
```

Example 4: Save

If var is specified, saves the value of var even if it is an array.

```
::cisco::eem::event_register_none

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

array set testvar "testvar1 ok testvar2 not_ok"
if {[catch {context_save TESTCTX testvar} errmsg]} {
      action_syslog msg "context_save failed: $errmsg"
} else {
      action_syslog msg "context_save succeeded"
}
```

Example 4: Retrieve

If var is specified, and index_if_array is specified, and var is an array variable, retrieves the specified array element value.

```
::cisco::eem::event_register_none

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

if {[catch {set testvar [context_retrieve TESTCTX testvar testvar1]} errmsg]} {
      action_syslog msg "context_retrieve failed: $errmsg"
} else {
      action_syslog msg "context_retrieve succeeded"
}
if {[info exists testvar]} {
      action_syslog msg "testvar exists and is $testvar"
} else {
      action_syslog msg "testvar doesn't exist"
}
```

context_save

Saves Tcl variables that match a given pattern in current and global namespaces with the given context name as identification. Use this Tcl command extension to save information outside of a policy. Saved information can be retrieved by a different policy using the **context_retrieve** command extension.

Note Once saved information is retrieved, it is automatically deleted. If that information is needed by another policy, the policy that retrieves it (using the **context_retrieve** command extension) should also save it again (using the **context_save** command extension).

Syntax

```
context_save ctxt [pattern]
```

Arguments

ctxt	(Mandatory) Context name.
pattern	(Optional) Glob-style pattern as used by the **string match** Tcl command. If this argument is not specified, the pattern defaults to the wildcard *. There are three constructs used in glob patterns: • * = all characters • ? = 1 character • [abc] = match one of a set of characters

Result String

None

Set _cerrno

A string displaying _cerrno, _cerr_sub_num, _cerr_sub_err, _cerr_str due to appl_setinfo error.

Sample Usage

For examples showing how to use the **context_save** and **context_retrieve** command extension functionality to save and retrieve data, see the Sample Usage, on page 147.