CONTENTS

PREFACE

Preface vii
Changes to this Document vii
Communications, Services, and Additional Information vii

CHAPTER 1
New and Changed Feature Information 1
System Monitoring Features Added or Modified in IOS XR Release 6.6.x 1

CHAPTER 2
Implementing System Logging 3
Implementing System Logging 3
Prerequisites for Configuring System Logging 4
Configuring System Logging 5
Configuring Logging to the Logging Buffer 5
Configuring Logging to a Remote Server 5
Configuring Logging to Terminal Lines 6
Modifying Logging to Console Terminal 7
Modifying Time Stamp Format 7
Suppressing Duplicate Syslog Messages 7
Archiving System Logging Messages to a Local Storage Device 7
Platform Automated Monitoring 8
PAM Events 9
Disable and Re-enable PAM 11
Data Archiving in PAM 12
Files Collected by PAM Tool 12

CHAPTER 3
Implementing Alarm Log Correlation 15
Implementing Alarm Log Correlation 15
Enabling PM Entity Instance Monitoring  37
Configuring PM Threshold Monitoring Templates  37
Configuring Instance Filtering by Regular Expression  38
Performance Management: Details  38

CHAPTER 6
Prerequisites for Configuring and Managing Embedded Event Manager Policies  51
Information About Configuring and Managing Embedded Event Manager Policies  52
Event Management  52
System Event Processing  52
Embedded Event Manager Scripts  53
Embedded Event Manager Policy Tcl Command Extension Categories  53
Cisco File Naming Convention for Embedded Event Manager  54
Embedded Event Manager Built-in Actions  54
Application-specific Embedded Event Management  55
Event Detection and Recovery  56
System Manager Event Detector  56
Timer Services Event Detector  57
Syslog Event Detector  57
None Event Detector  58
Watchdog System Monitor Event Detector  58
Distributed Event Detectors  59
Embedded Event Manager Event Scheduling and Notification  59
Reliability Statistics  59
How to Configure and Manage Embedded Event Manager Policies  61
Configuring Environmental Variables  61
Registering Embedded Event Manager Policies  61
How to Write Embedded Event Manager Policies Using Tcl  62
Registering and Defining an EEM Tcl Script  62
Suspending EEM Policy Execution  63
Specifying a Directory for Storing EEM Policies  63
Programming EEM Policies with Tcl  63
Creating an EEM User Tcl Library Index  68
Creating an EEM User Tcl Package Index  70
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implementing IP Service Level Agreements</td>
<td>77</td>
</tr>
<tr>
<td>IP Service Level Agreements Technology Overview</td>
<td>77</td>
</tr>
<tr>
<td>Service Level Agreements</td>
<td>78</td>
</tr>
<tr>
<td>Benefits of IP Service Level Agreements</td>
<td>79</td>
</tr>
<tr>
<td>Prerequisites for Implementing IP Service Level Agreements</td>
<td>79</td>
</tr>
<tr>
<td>Restrictions for Implementing IP Service Level Agreements</td>
<td>80</td>
</tr>
<tr>
<td>Measuring Network Performance with IP Service Level Agreements</td>
<td>80</td>
</tr>
<tr>
<td>IP SLA Responder and IP SLA Control Protocol</td>
<td>81</td>
</tr>
<tr>
<td>Response Time Computation for IP SLA</td>
<td>81</td>
</tr>
<tr>
<td>IP SLA Operation Scheduling</td>
<td>82</td>
</tr>
<tr>
<td>Two-Way Active Measurement Protocol (TWAMP)</td>
<td>82</td>
</tr>
<tr>
<td>The TWAMP Entities</td>
<td>82</td>
</tr>
<tr>
<td>TWAMP Protocols</td>
<td>83</td>
</tr>
<tr>
<td>Restrictions of TWAMP on the Router</td>
<td>84</td>
</tr>
<tr>
<td>Configuring TWAMP on the Router</td>
<td>84</td>
</tr>
<tr>
<td>Verification of TWAMP</td>
<td>84</td>
</tr>
<tr>
<td>TWAMP-Light</td>
<td>85</td>
</tr>
<tr>
<td>Restrictions of TWAMP-Light on the Cisco NCS 5500 Series Router</td>
<td>85</td>
</tr>
<tr>
<td>Configuring TWAMP-Light on the NCS 5500 Series Router</td>
<td>85</td>
</tr>
<tr>
<td>Verification of TWAMP-Light on the NCS 5500 Series Router</td>
<td>86</td>
</tr>
</tbody>
</table>
Preface

The System Monitoring Configuration Guide for Cisco NCS 5500 Series Routers preface contains these sections:

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

Changes to this Document

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

<table>
<thead>
<tr>
<th>Date</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>May 2019</td>
<td>Initial release of this document.</td>
</tr>
</tbody>
</table>

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.
### System Monitoring Features Added or Modified in IOS XR Release 6.6.x

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Changed in Release</th>
<th>Where Documented</th>
</tr>
</thead>
<tbody>
<tr>
<td>TWAMP-Light</td>
<td>This feature was introduced in this release.</td>
<td>Release 6.6.25</td>
<td>TWAMP-Light, on page 85</td>
</tr>
</tbody>
</table>

This table summarizes the new and changed feature information for the *System Monitoring Configuration Guide for Cisco NCS 5500 Series Routers*, and tells you where they are documented.

- System Monitoring Features Added or Modified in IOS XR Release 6.6.x, on page 1
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

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1.2</td>
<td>Platform Automated Monitoring (PAM) tool was introduced for all Cisco IOS XR 64-bit platforms.</td>
</tr>
</tbody>
</table>

- 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:


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

Table 1: Format of Syslog Messages

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>node-id</td>
<td>Node from which the syslog message originated.</td>
</tr>
</tbody>
</table>
### Prerequisites for Configuring System Logging

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

#### Field Description

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

### 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

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

---

System Monitoring Configuration Guide for Cisco NCS 5500 Series Routers, IOS XR Release 6.6.x
• 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, 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**

<table>
<thead>
<tr>
<th>Features</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNIX system log facility</td>
<td>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 <code>logging facility</code> command.</td>
</tr>
</tbody>
</table>
### Features

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hostname prefix logging</strong></td>
</tr>
<tr>
<td>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 <strong>logging hostname</strong> command to append a hostname prefix to syslog messages sent to syslog servers.</td>
</tr>
<tr>
<td><strong>Syslog source address logging</strong></td>
</tr>
<tr>
<td>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 <strong>logging source-interface</strong> command to set all syslog messages to contain the same IP address, regardless of which interface the syslog message uses to exit the router.</td>
</tr>
</tbody>
</table>

### Configuration Example

This example shows the configuration for sending syslog messages to an external syslog server. The ip address 10.3.32.154 is configured as the syslog server and the logging trap command is used to limit the syslog messages sent to syslog servers based on severity.

```
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# logging 10.3.32.154
(Optional) RP/0/RP0/CPU0:Router(config)# logging 10.3.32.155 vrf vrf2
RP/0/RP0/CPU0:Router(config)# logging trap warnings
RP/0/RP0/CPU0:Router(config)# logging facility kern (optional)
RP/0/RP0/CPU0:Router(config)# logging hostnameprefix 123.12.35.7 (optional)
RP/0/RP0/CPU0:Router(config)# logging source-interface HundredGigE 0/0/0/3 (optional)
RP/0/RP0/CPU0:Router(config)# commit
```

### Related Topics

- Configuring Logging to the Logging Buffer, on page 5
- Configuring Logging to Terminal Lines, on page 6

### 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
```
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.

```plaintext
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.

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

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.

```plaintext
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

<table>
<thead>
<tr>
<th>Features</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>archive-length</td>
<td>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.</td>
</tr>
<tr>
<td>archive-size</td>
<td>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.</td>
</tr>
<tr>
<td>device</td>
<td>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.</td>
</tr>
<tr>
<td>file-size</td>
<td>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.</td>
</tr>
<tr>
<td>frequency</td>
<td>Specifies if logs are collected on a daily or weekly basis.</td>
</tr>
<tr>
<td>severity</td>
<td>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.</td>
</tr>
</tbody>
</table>

**Configuration Example**

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

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

Platform Automated Monitoring

Platform Automated Monitoring (PAM) is a system monitoring tool integrated into Cisco IOS XR software image to monitor issues such as process crash, memory leak, CPU hog, tracebacks, syslog and disk usage. PAM is enabled by default on all Cisco IOS XR 64 bit platforms. When the PAM tool detects any of these issues, it can automatically generate alerts or take corrective actions.
system issues, it collects the required data to troubleshoot the issue, and generates a syslog message stating the issue. The auto-collected troubleshooting information is then stored as a separate file in harddisk:/cisco_support/ or in /misc/disk1/cisco_support/ directory.

**PAM Events**

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

The format of the .tgz file is: PAM-<platform>-<PAM event>-<node-name>-<PAM process>-<YYYYMMDD>-<checksum>.tgz. For example, PAM-ncs5500-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-ncs5500-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-ncs5500-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-ncs5500-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-ncs5500-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-ncs5500-disk_usage-xr_0_1_CPU0-2016Jun20-135904.tgz
(Please copy tgz file out of the router and send to Cisco support. This tgz file will be removed after 14 days.)
```
Disable and Re-enable PAM

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

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

• From Cisco IOS XR Command Line Interface:

```
Router# show processes pam_manager location all
Tue Jun 14 17:58:42.791 UTC
node: node0_RP0_CPU0
    Job Id: 317
    PID: 14070
    Executable path: /opt/cisco/XR/packages/iosxr-infra.rp-6.1.1.17I/bin/pam_manager
    Instance #: 1
    Version ID: 00.00.0000
    Respawn: ON
    Respawn count: 4
    Last started: Mon Jun 13 23:08:43 2016
    Process state: Run
    Package state: Normal
        core: MAINMEM
        Max. core: 0
        Level: 999
        Placement: None
    startup_path:
        /opt/cisco/XR/packages/iosxr-infra.rp-6.1.1.17I/startup/pam_manager.startup
    Ready: 0.166s
    Process cpu time: 0.200 user, 0.310 kernel, 0.510 total

<table>
<thead>
<tr>
<th>JID</th>
<th>TID</th>
<th>Stack</th>
<th>pri</th>
<th>state</th>
<th>NAME</th>
<th>rt_pri</th>
</tr>
</thead>
<tbody>
<tr>
<td>317</td>
<td>14070</td>
<td>OK</td>
<td>20</td>
<td>Sleeping</td>
<td>pam_manager</td>
<td>0</td>
</tr>
<tr>
<td>317</td>
<td>14071</td>
<td>OK</td>
<td>20</td>
<td>Sleeping</td>
<td>lwm_debug_threa</td>
<td>0</td>
</tr>
<tr>
<td>317</td>
<td>14076</td>
<td>OK</td>
<td>20</td>
<td>Sleeping</td>
<td>pam_manager</td>
<td>0</td>
</tr>
<tr>
<td>317</td>
<td>14077</td>
<td>OK</td>
<td>20</td>
<td>Sleeping</td>
<td>lwm_service_thr</td>
<td>0</td>
</tr>
<tr>
<td>317</td>
<td>14078</td>
<td>OK</td>
<td>20</td>
<td>Sleeping</td>
<td>qsm_service_thr</td>
<td>0</td>
</tr>
<tr>
<td>317</td>
<td>14080</td>
<td>OK</td>
<td>20</td>
<td>Sleeping</td>
<td>pam_manager</td>
<td>0</td>
</tr>
</tbody>
</table>
```

• From router shell prompt:

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

Disable PAM

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

For local RP:
```
Router# process shutdown pam_manager
```

For all RPs:
```
Router# process shutdown pam_manager location all
```
Re-enable PAM

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

For local RP:
\begin{verbatim}
Router# process start pam_manager
\end{verbatim}

For all RPs:
\begin{verbatim}
Router# process start pam_manager location all
\end{verbatim}

To start PAM on all locations, the \textit{pam\_manager} process should be restarted on all nodes by using the \texttt{location all} option in the \texttt{process start pam\_manager} command.

\begin{itemize}
  \item 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 \texttt{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 \texttt{<process name>\_pid\_by\_user:\<yyyyymmdd>-<hhmmss>\_<node>\_<checksum>.core.gz}.

\end{itemize}

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.

\begin{tabular}{|l|l|}
\hline
\textbf{Event Name} & \textbf{Commands and Files Collected by PAM} \\
\hline
Process crash & \begin{itemize}
  \item show install active
  \item show platform
  \item show version
  \item core (gz) file
  \item core.txt file
\end{itemize} \\
\hline
\end{tabular}
<table>
<thead>
<tr>
<th>Event Name</th>
<th>Commands and Files Collected by PAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process traceback</td>
<td>• show dll</td>
</tr>
<tr>
<td></td>
<td>• show install active</td>
</tr>
<tr>
<td></td>
<td>• show logging</td>
</tr>
<tr>
<td></td>
<td>• show platform</td>
</tr>
<tr>
<td></td>
<td>• show version</td>
</tr>
<tr>
<td>Memory leak</td>
<td>• show install active</td>
</tr>
<tr>
<td></td>
<td>• show platform</td>
</tr>
<tr>
<td></td>
<td>• show version</td>
</tr>
<tr>
<td></td>
<td>• core (gz) file</td>
</tr>
<tr>
<td></td>
<td>• dumpcore running</td>
</tr>
<tr>
<td></td>
<td>• continuous memory usage snapshots</td>
</tr>
<tr>
<td>Show logging event</td>
<td>• show install active</td>
</tr>
<tr>
<td></td>
<td>• show logging</td>
</tr>
<tr>
<td></td>
<td>• show platform</td>
</tr>
<tr>
<td></td>
<td>• show version</td>
</tr>
<tr>
<td></td>
<td>• core (gz) file</td>
</tr>
<tr>
<td></td>
<td>• core.txt file</td>
</tr>
<tr>
<td>CPU hog</td>
<td>• follow process</td>
</tr>
<tr>
<td></td>
<td>• pstack</td>
</tr>
<tr>
<td></td>
<td>• show dll</td>
</tr>
<tr>
<td></td>
<td>• show install active</td>
</tr>
<tr>
<td></td>
<td>• show platform</td>
</tr>
<tr>
<td></td>
<td>• show version</td>
</tr>
<tr>
<td></td>
<td>• top -H</td>
</tr>
<tr>
<td></td>
<td>• core (gz) file</td>
</tr>
<tr>
<td></td>
<td>• CPU usage snapshots</td>
</tr>
<tr>
<td>Event Name</td>
<td>Commands and Files Collected by PAM</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------------------------------------------</td>
</tr>
<tr>
<td>Disk usage</td>
<td>• show install active</td>
</tr>
<tr>
<td></td>
<td>• show platform</td>
</tr>
<tr>
<td></td>
<td>• show version</td>
</tr>
<tr>
<td></td>
<td>• console log</td>
</tr>
<tr>
<td></td>
<td>• core (gz) file</td>
</tr>
<tr>
<td></td>
<td>• Disk usage snapshots</td>
</tr>
</tbody>
</table>
CHAPTER 3

Implementing Alarm Log Correlation

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

- Implementing Alarm Log Correlation, on page 15

Implementing Alarm Log Correlation

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

Prerequisites for Implementing Alarm Log Correlation

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

Information About Implementing Alarm Log Correlation

Alarm Logging and Debugging Event Management System

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

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

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

Figure 1: ALDEMS Component Communications, on page 16 illustrates the relationship between the components that constitute ALDEMS.
Correlator

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

System Logging Process

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

Alarm Logger

The alarm logger is the final destination for system logging messages forwarded on the router. The alarm logger stores alarm messages in the logging events buffer. The logging events buffer is circular; that is, when full, it overwrites the oldest messages in the buffer.
Alarms are prioritized in the logging events buffer. When it is necessary to overwrite an alarm record, the logging events buffer overwrites messages in the following order: nonbistate alarms first, then bistate alarms in the CLEAR state, and, finally, bistate alarms in the SET state.

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

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

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

Configuring Alarm Log Correlation

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

Configuring Logging Correlation Rules

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

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

Configuration Example

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

```
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# logging correlator rule rule1 type stateful
RP/0/RP0/CPU0:Router(config-corr-rule-st)# timeout 60000
RP/0/RP0/CPU0:Router(config)# logging correlator apply-rule rule1
RP/0/RP0/CPU0:Router(config-corr-apply-rule)# all-of-router
or
RP/0/RP0/CPU0:Router(config-corr-apply-rule)# location 0/1/CPU0
or
RP/0/RP0/CPU0:Router(config-corr-apply-rule)# context HundredGigE_0_0_0_0
RP/0/RP0/CPU0:Router(config)# commit
```
Configuring a Logging Correlation Rule Set

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

**Configuration Example**

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

```
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# logging correlator ruleset ruleset1
RP/0/RP0/CPU0:Router(config-corr-ruleset)# rulename stateful_rule1
RP/0/RP0/CPU0:Router(config-corr-ruleset)# rulename stateful_rule2
RP/0/RP0/CPU0:Router(config)# logging correlator apply ruleset ruleset1
RP/0/RP0/CPU0:Router(config-corr-apply-rule)# all-of-router
or
RP/0/RP0/CPU0:Router(config-corr-apply-rule)# location 0/2/CPU0
or
RP/0/RP0/CPU0:Router(config-corr-apply-rule)# context HundredGigE_0_0_0_0
RP/0/RP0/CPU0: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.

```
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# logging correlator rule rule_stateful type stateful
RP/0/RP0/CPU0:Router(config-corr-rule-st)# rootcause CAT_BI_1 GROUP_BI_1 CODE_BI_1
RP/0/RP0/CPU0:Router(config-corr-rule-st)# nonrootcause
RP/0/RP0/CPU0:Router(config-corr-rule-st-nonrc)# alarm CAT_BI_2 GROUP_BI_2 CODE_BI_2
RP/0/RP0/CPU0: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 22](#).

**Configuration Example**

This example shows how to configure hierarchical correlation rule flags.

```
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# logging correlator rule rule_nonstateful type nonstateful
RP/0/RP0/CPU0:Router(config-corr-rule-st)# reissue-nonbistate
RP/0/RP0/CPU0:Router(config-corr-rule-st)# reparent
RP/0/RP0/CPU0:Router(config)# commit
RP/0/RP0/CPU0: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.

```
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# logging suppress rule infobistate
RP/0/RP0/CPU0:Router(config-suppr-rule)# alarm MBGL COMMIT SUCCEEDED
RP/0/RP0/CPU0:Router(config)# logging suppress apply rule infobistate
RP/0/RP0/CPU0:Router(config-suppr-apply-rule)# all-of-router
RP/0/RP0/CPU0:Router(config)# commit
```

Modifying Logging Events Buffer Settings

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

**Configuration Example**

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

```
RP/0/RP0/CPU0:Router# configure terminal
RP/0/RP0/CPU0:Router(config)# logging events buffer-size 50000
RP/0/RP0/CPU0:Router(config)# logging events threshold 85
RP/0/RP0/CPU0:Router(config)# logging events level warnings
RP/0/RP0/CPU0: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.

```
RP/0/RP0/CPU0:Router# configure terminal
RP/0/RP0/CPU0:Router(config)# logging correlator buffer-size 100000
RP/0/RP0/CPU0:Router(config)# exit
RP/0/RP0/CPU0:Router# clear logging correlator delete 48 49 50 (optional)
RP/0/RP0/CPU0: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 22.

Configuration Example

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

```
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# logging events display-location
RP/0/RP0/CPU0:Router(config)# commit
```

Configuring SNMP Correlation Rules

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

Configuration Example

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

```
RP/0/RP0/CPU0:Router# configure terminal
RP/0/RP0/CPU0:Router(config)# snmp-server correlator buffer-size 600 (optional)
RP/0/RP0/CPU0:Router(config)# snmp-server correlator rule test rootcause A varbind A1 value regex RA1 nonrootcause trap B varbind B1 index regex RB1
RP/0/RP0/CPU0:Router(config)# snmp-server correlator apply rule test host ipv4 address 1.2.3.4
RP/0/RP0/CPU0:Router(config)# commit
```

Configuring SNMP Correlation Ruleset

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

Configuration Example

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

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

Alarm Logging Correlation-Details

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

**Correlation Rules**

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

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

- Message category
- Message group
- Message code

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

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

**Alarm Severity Level and Filtering**

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

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

*Table 5: Alarm Severity Levels for Event Logging*

<table>
<thead>
<tr>
<th>Severity Level</th>
<th>System Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Emergencies</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>Rule 1</th>
<th>Category</th>
<th>Group</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root Cause 1</td>
<td>Cat 1</td>
<td>Group 1</td>
<td>Code 1</td>
</tr>
<tr>
<td>Non-root Cause 2</td>
<td>Cat 2</td>
<td>Group 2</td>
<td>Code 2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rule 2</th>
<th>Category</th>
<th>Group</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root Cause 2</td>
<td>Cat 2</td>
<td>Group 2</td>
<td>Code 2</td>
</tr>
<tr>
<td>Non-root Cause 3</td>
<td>Cat 3</td>
<td>Group 3</td>
<td>Code 3</td>
</tr>
</tbody>
</table>

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.
Onboard Failure Logging

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

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

The Onboard Failure Logging (OBFL) functionality is enhanced to provide a generic library that can be used by different clients to log string messages.

Caution

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

Prerequisites

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:

- FRU part serial number
- OS version
- Boot time
• Total run time
• Temperature and voltage at boot
• Temperature and voltage history

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

**Baseline Data Collection**

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

*Table 6: Data Types*

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

**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*

<table>
<thead>
<tr>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environmental Factors</td>
</tr>
<tr>
<td>Environmental Factors</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
### Onboard Failure Logging

#### Information About OBFL

<table>
<thead>
<tr>
<th>Details</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calendar Time</td>
<td>The time when OBFL logging was disabled.</td>
</tr>
<tr>
<td>Disabled</td>
<td></td>
</tr>
<tr>
<td>Cleared</td>
<td>The time when OBFL logging was cleared.</td>
</tr>
<tr>
<td>Reset to 0</td>
<td>The time when total line card runtime is reset to zero.</td>
</tr>
</tbody>
</table>

---

#### 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 NCS 5500 Series router.

**Table 8: OBFL Support on Cisco NCS 5500 Series Router**

<table>
<thead>
<tr>
<th>Card Type</th>
<th>Cisco NCS 5500 Series Router</th>
</tr>
</thead>
<tbody>
<tr>
<td>Route processor (RP)</td>
<td>Supported</td>
</tr>
<tr>
<td>Fabric cards</td>
<td>Supported</td>
</tr>
<tr>
<td>Line card</td>
<td>Supported</td>
</tr>
<tr>
<td>Power supply cards</td>
<td>Not Supported</td>
</tr>
<tr>
<td>Fan tray</td>
<td>Supported</td>
</tr>
<tr>
<td>System Controller</td>
<td>Supported</td>
</tr>
</tbody>
</table>
Information About OBFL
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

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

• You must install and activate the Package Installation Envelope (PIE) for the manageability software.

• 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 31 illustrates the relationship between the components that constitute the PM system.
**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

<table>
<thead>
<tr>
<th>Entity Classes</th>
<th>Instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>BGP</td>
<td>Neighbors or Peers</td>
</tr>
<tr>
<td>Interface Basic Counters</td>
<td>Interfaces</td>
</tr>
<tr>
<td>Interface Data Rates</td>
<td>Interfaces</td>
</tr>
<tr>
<td>Interface Generic Counters</td>
<td>Interfaces</td>
</tr>
<tr>
<td>MPLS LDP</td>
<td>LDP Sessions</td>
</tr>
<tr>
<td>Node CPU</td>
<td>Nodes</td>
</tr>
<tr>
<td>Node Memory</td>
<td>Nodes</td>
</tr>
<tr>
<td>Node Process</td>
<td>Processes</td>
</tr>
<tr>
<td>OSPFv2</td>
<td>Processes</td>
</tr>
<tr>
<td>OSPFv3</td>
<td>Processes</td>
</tr>
</tbody>
</table>

For a list of all attributes associated with the entities that constitute the PM system, see Table 12: Attributes and Values, on page 39.

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.

Binary File Format for Exporting PM Statistics

This sample describes the binary file format:

Version : 4 Bytes
NoOf Entities : 1 Byte (e.g. 4)
Entity Identifier : 1 Byte (e.g NODE=1, Interface=2, BGP=3)
Options : 2 Bytes
NoOf SubEntities : 1 Byte (2)
SubEntity Identifier : 1 Byte (e.g BGP-PEERS)
Time Stamp 4 Bytes (Reference Time : Start Ref Time)
No Of Instances : 2 Byte (e.g 100)
  Key Instance : Variable
    NoOfSamples: 1 Byte (e.g 10 Samples)
    SampleNo : 1 Byte (e.g Sample No 1)
Time Stamp 4 Bytes (Sample Time)
  StatCounterName : 1 Byte (PeerSessionsEst=1)
**Binary File ID Assignments for Entity, Subentity, and StatsCounter Names**

This table describes the assignment of various values and keys which is present in the binary file.

*Table 10: Binary Format Values and Keys*

<table>
<thead>
<tr>
<th>Entity</th>
<th>Subentity</th>
<th>Key</th>
<th>StatsCounters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node (1)</td>
<td>CPU (1)</td>
<td>CPU Key &lt;Node ID&gt;</td>
<td>See Table 11: Supported StatsCounters for Entities and Subentites, on page 34</td>
</tr>
<tr>
<td></td>
<td>Memory (2)</td>
<td>Memory Key &lt;Node ID&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Process (3)</td>
<td>Node Process Key &lt;NodeProcessID&gt;</td>
<td></td>
</tr>
<tr>
<td>Interface (2)</td>
<td>Generic Counters (1)</td>
<td>Generic Counters Key &lt;ifName&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Data Rate Counters (2)</td>
<td>Data Rate Counters Key &lt;ifName&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Basic Counters (3)</td>
<td>Basic Counters Key &lt;ifName&gt;</td>
<td></td>
</tr>
<tr>
<td>BGP (3)</td>
<td>Peer (1)</td>
<td>Peer Key &lt;IpAddress&gt;</td>
<td></td>
</tr>
<tr>
<td>MPLS (4)</td>
<td>Reserved (1)</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reserved (2)</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LDP (4)</td>
<td>LDP Session Key &lt;IpAddress&gt;</td>
<td></td>
</tr>
<tr>
<td>OSPF (5)</td>
<td>v2protocol (1)</td>
<td>Instance &lt;process_instance&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>v3protocol (2)</td>
<td>Instance &lt;process_instance&gt;</td>
<td></td>
</tr>
</tbody>
</table>

---

**Note**

- `<ifName>`—The length is variable. The first two bytes contain the size of the Instance ID; this is followed by the Instance ID string (that is, an Interface name).
- `<IpAddress>`—4 bytes that contain the IP address.
- `<NodeProcessID>`—64-bit Instance ID. The first 32 bits contain the node ID, and the second 32 bits contain the process ID.
- `<NodeID>`—32-bit instance ID that contains the Node ID.
- `<process_instance>`—The length is variable. The first two bytes contain the size of Instance ID followed by Instance ID string (that is, a process name).
The numbers in parenthesis (the numbers that are associated with each entity and subentity in Table 10: Binary Format Values and Keys, on page 33) denote the entity and subEntity IDs that are displayed in the TFTP File.

This table describes the supported statistics counters that are collected in the binary file for entities and subentities.

**Table 11: Supported StatsCounters for Entities and Subentites**

<table>
<thead>
<tr>
<th>Entity</th>
<th>Subentity</th>
<th>StatsCounters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node (1)</td>
<td>CPU (1)</td>
<td>AverageCPUUsed, NoProcesses</td>
</tr>
<tr>
<td></td>
<td>Memory (2)</td>
<td>CurrMemory, PeakMemory</td>
</tr>
<tr>
<td></td>
<td>Process (3)</td>
<td>PeakMemory, AverageCPUUsed, NoThreads</td>
</tr>
<tr>
<td></td>
<td>Data Rate Counters (2)</td>
<td>InputDataRate, InputPacketRate, OutputDataRate, OutputPacketRate, InputPeakRate, InputPeakPkts, OutputPeakRate, OutputPeakPkts, Bandwidth</td>
</tr>
<tr>
<td></td>
<td>Basic Counters</td>
<td>InPackets, InOctets, OutPackets, OutOctets, InputTotalDrops, InputQueueDrops, InputTotalErrors, OutputTotalErrors, OutputQueueDrops, OutputTotalErrors</td>
</tr>
<tr>
<td>BGP (3)</td>
<td>Peer (1)</td>
<td>InputMessages, OutputMessages, InputUpdateMessages, OutputUpdateMessages, ConnEstablished, ConnDropped, ErrorsReceived, ErrorsSent</td>
</tr>
<tr>
<td>OSPF (5)</td>
<td>v2protocol (1)</td>
<td>InputPackets, OutputPackets, InputHelloPackets, OutputHelloPackets, InputDBDs, InputDBDsLSA, OutputDBDs, OutputDBDsLSA, InputLSRequests, InputLSRequestsLSA, OutputLSRequests, OutputLSRequestsLSA, InputLSAUpdates, InputLSAUpdatesLSA, OutputLSAUpdates, OutputLSAUpdatesLSA, InputLSAACKs, InputLSAACKsLSA, OutputLSAACKs, OutputLSAACKsLSA, ChecksumErrors</td>
</tr>
</tbody>
</table>
Filenaming Convention Applied to Binary Files

These filenaming convention is applied to PM statistics collections that are sent to the directory location configured on the TFTP server:

<LR_NAME>_<EntityName>_<SubentityName>_<TimeStamp>

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 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
Enabling PM Entity Instance Monitoring

Entity instance monitoring gathers statistics from attributes associated with a specific entity instance. When an entity instance is enabled for monitoring, the PM system gathers statistics from only attributes associated with the specified entity instance. The PM system uses the sampling cycle that is configured in the PM statistics collection template for the entity being monitored. Entity instance monitoring, however, is a separate process from that of the PM statistics collection; therefore, it does not interfere with PM statistics collection. Furthermore, the data from entity instance monitoring collection is independent of PM statistics collection. Unlike PM statistics collection, the data from entity instance monitoring is not exported to the TFTP server. For more information about all the attributes associated with each entity instance and commands, see Performance Management: Details, on page 38.

Configuration Example

This example shows how to enable entity instance monitoring for a node CPU entity instance.

```
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# performance-mgmt apply monitor node cpu location 0/RP0/CPU0 default
RP/0/RP0/CPU0:Router(config)# 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 38

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 AverageCpuUsed attribute of the node CPU entity. The threshold condition in this PM threshold condition monitors the AverageCpuUsed attribute to determine whether the average CPU use is greater than 25 percent.
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-regexp)# 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 12: Attributes and Values**

<table>
<thead>
<tr>
<th>Entity</th>
<th>Attributes</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>bgp</td>
<td>ConnDropped</td>
<td>Number of times the connection was dropped.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>ConnEstablished</td>
<td>Number of times the connection was established.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>ErrorsReceived</td>
<td>Number of error notifications received on the connection.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>ErrorsSent</td>
<td>Number of error notifications sent on the connection.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>InputMessages</td>
<td>Number of messages received.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>InputUpdateMessages</td>
<td>Number of update messages received.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>OutputMessages</td>
<td>Number of messages sent.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>OutputUpdateMessages</td>
<td>Number of update messages sent.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td>interface data-rates</td>
<td>Bandwidth</td>
<td>Bandwidth in kbps.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>InputDataRate</td>
<td>Input data rate in kbps.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>InputPacketRate</td>
<td>Input packets per second.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>InputPeakRate</td>
<td>Peak input data rate.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>InputPeakPkts</td>
<td>Peak input packet rate.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>OutputDataRate</td>
<td>Output data rate in kbps.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>OutputPacketRate</td>
<td>Output packets per second.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>OutputPeakPkts</td>
<td>Peak output packet rate.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>OutputPeakRate</td>
<td>Peak output data rate.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td>Entity</td>
<td>Attributes</td>
<td>Description</td>
<td>Values</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------</td>
<td>----------------------------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>interface</td>
<td>InPackets</td>
<td>Packets received.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td>basic-counters</td>
<td>InOctets</td>
<td>Bytes received.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>OutPackets</td>
<td>Packets sent.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>OutOctets</td>
<td>Bytes sent.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>InputTotalDrops</td>
<td>Inbound correct packets discarded.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>InputQueueDrops</td>
<td>Input queue drops.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>InputTotalErrors</td>
<td>Inbound incorrect packets discarded.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>OutputTotalDrops</td>
<td>Outbound correct packets discarded.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>OutputQueueDrops</td>
<td>Output queue drops.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>OutputTotalErrors</td>
<td>Outbound incorrect packets discarded.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
</tbody>
</table>
## Performance Management: Details

<table>
<thead>
<tr>
<th>Entity</th>
<th>Attributes</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>interface generic-counters</td>
<td>InBroadcastPkts</td>
<td>Broadcast packets received.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>InMulticastPkts</td>
<td>Multicast packets received.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>InOctets</td>
<td>Bytes received.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>InPackets</td>
<td>Packets received.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>InputCRC</td>
<td>Inbound packets discarded with incorrect CRC.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>InputFrame</td>
<td>Inbound framing errors.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>InputOverrun</td>
<td>Input overruns.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>InputQueueDrops</td>
<td>Input queue drops.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>InputTotalDrops</td>
<td>Inbound correct packets discarded.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>InputTotalErrors</td>
<td>Inbound incorrect packets discarded.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>InUcastPkts</td>
<td>Unicast packets received.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>InputUnknownProto</td>
<td>Inbound packets discarded with unknown protocol.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>OutBroadcastPkts</td>
<td>Broadcast packets sent.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>OutMulticastPkts</td>
<td>Multicast packets sent.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>OutOctets</td>
<td>Bytes sent.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>OutPackets</td>
<td>Packets sent.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>OutputTotalDrops</td>
<td>Outbound correct packets discarded.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>OutputTotalErrors</td>
<td>Outbound incorrect packets discarded.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>OutUcastPkts</td>
<td>Unicast packets sent.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>OutputUnderrun</td>
<td>Output underruns.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td>Entity</td>
<td>Attributes</td>
<td>Description</td>
<td>Values</td>
</tr>
<tr>
<td>-------------</td>
<td>------------------</td>
<td>--------------------------------------------------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>mpls ldp</td>
<td>AddressMsgsRcvd</td>
<td>Address messages received.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>AddressMsgsSent</td>
<td>Address messages sent.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>AddressWithdrawMsgsRcvd</td>
<td>Address withdraw messages received.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>AddressWithdrawMsgsSent</td>
<td>Address withdraw messages sent.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>InitMsgsSent</td>
<td>Initial messages sent.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>InitMsgsRcvd</td>
<td>Initial messages received.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>KeepaliveMsgsRcvd</td>
<td>Keepalive messages received.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>KeepaliveMsgsSent</td>
<td>Keepalive messages sent.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>LabelMappingMsgsRcvd</td>
<td>Label mapping messages received.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>LabelMappingMsgsSent</td>
<td>Label mapping messages sent.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>LabelReleaseMsgsRcvd</td>
<td>Label release messages received.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>LabelReleaseMsgsSent</td>
<td>Label release messages sent.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>LabelWithdrawMsgsRcvd</td>
<td>Label withdraw messages received.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>LabelWithdrawMsgsSent</td>
<td>Label withdraw messages sent.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>NotificationMsgsRcvd</td>
<td>Notification messages received.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>NotificationMsgsSent</td>
<td>Notification messages sent.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>TotalMsgsRcvd</td>
<td>Total messages received.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>TotalMsgsSent</td>
<td>Total messages sent.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td>Entity</td>
<td>Attributes</td>
<td>Description</td>
<td>Values</td>
</tr>
<tr>
<td>---------------</td>
<td>----------------</td>
<td>--------------------------------------------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>node cpu</td>
<td>AverageCPUUsed</td>
<td>Average percent CPU utilization.</td>
<td>Range is a percentage from 0 to 100.</td>
</tr>
<tr>
<td></td>
<td>NoProcesses</td>
<td>Number of processes.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td>node memory</td>
<td>CurrMemory</td>
<td>Current application memory (in bytes) in use.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>PeakMemory</td>
<td>Maximum system memory (in MB) used since bootup.</td>
<td>Range is from 0 to 4194304.</td>
</tr>
<tr>
<td>node process</td>
<td>AverageCPUUsed</td>
<td>Average percent CPU utilization.</td>
<td>Range is a percentage from 0 to 100.</td>
</tr>
<tr>
<td></td>
<td>NoThreads</td>
<td>Number of threads.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>PeakMemory</td>
<td>Maximum dynamic memory (in KB) used since startup time.</td>
<td>Range is from 0 to 4194304.</td>
</tr>
<tr>
<td>Entity</td>
<td>Attributes</td>
<td>Description</td>
<td>Values</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------</td>
<td>--------------------------------------------------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>ospf v2protocol</td>
<td>InputPackets</td>
<td>Total number of packets received.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>OutputPackets</td>
<td>Total number of packets sent.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>InputHelloPackets</td>
<td>Number of Hello packets received.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>OutputHelloPackets</td>
<td>Number of Hello packets sent.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>InputDBDs</td>
<td>Number of DBD packets received.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>InputDBDsLSA</td>
<td>Number of LSA received in DBD packets.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>OutputDBDs</td>
<td>Number of DBD packets sent.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>OutputDBDsLSA</td>
<td>Number of LSA sent in DBD packets.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>InputLSRequests</td>
<td>Number of LS requests received.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>InputLSRequestsLSA</td>
<td>Number of LSA received in LS requests.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>OutputLSRequests</td>
<td>Number of LS requests sent.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>OutputLSRequestsLSA</td>
<td>Number of LSA sent in LS requests.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>InputLSAUpdates</td>
<td>Number of LSA updates received.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>InputLSAUpdatesLSA</td>
<td>Number of LSA received in LSA updates.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>OutputLSAUpdates</td>
<td>Number of LSA updates sent.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>OutputLSAUpdatesLSA</td>
<td>Number of LSA sent in LSA updates.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>InputLSAAcks</td>
<td>Number of LSA acknowledgements received.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td>Entity</td>
<td>Attributes</td>
<td>Description</td>
<td>Values</td>
</tr>
<tr>
<td>----------------</td>
<td>----------------</td>
<td>-------------------------------------------------------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>InputLSAAcksLSA</td>
<td>Number of LSA</td>
<td>Number of LSA received in LSA acknowledgements.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OutputLSAAcks</td>
<td>Number of LSA</td>
<td>Number of LSA acknowledgements sent</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OutputLSAAcksLSA</td>
<td>Number of LSA</td>
<td>Number of LSA sent in LSA acknowledgements.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ChecksumErrors</td>
<td>Number of packets received with checksum errors.</td>
<td>Range is from 0 to 4294967295.</td>
<td></td>
</tr>
<tr>
<td>Entity</td>
<td>Attributes</td>
<td>Description</td>
<td>Values</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------</td>
<td>--------------------------------------------------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>ospf v3protocol</td>
<td>InputPackets</td>
<td>Total number of packets received.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>OutputPackets</td>
<td>Total number of packets sent.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>InputHelloPackets</td>
<td>Number of Hello packets received.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>OutputHelloPackets</td>
<td>Number of Hello packets sent.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>InputDBDs</td>
<td>Number of DBD packets received.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>InputDBDsLSA</td>
<td>Number of LSA received in DBD packets.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>OutputDBDs</td>
<td>Number of DBD packets sent.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>OutputDBDsLSA</td>
<td>Number of LSA sent in DBD packets.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>InputLSRequests</td>
<td>Number of LS requests received.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>InputLSRequestsLSA</td>
<td>Number of LSA received in LS requests.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>OutputLSRequests</td>
<td>Number of LS requests sent.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>OutputLSRequestsLSA</td>
<td>Number of LSA sent in LS requests.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>InputLSAUpdates</td>
<td>Number of LSA updates received.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>InputLSRequestsLSA</td>
<td>Number of LSA received in LS requests.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>OutputLSAUpdates</td>
<td>Number of LSA updates sent.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>OutputLSAUpdatesLSA</td>
<td>Number of LSA sent in LSA updates.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td></td>
<td>InputLSAAcks</td>
<td>Number of LSA acknowledgements received.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td>Entity</td>
<td>Attributes</td>
<td>Description</td>
<td>Values</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------</td>
<td>--------------------------------------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>InputLSAAcks</td>
<td>LSA</td>
<td>Number of LSA received in LSA acknowledgements.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td>OutputLSAAcks</td>
<td>LSA</td>
<td>Number of LSA acknowledgements sent</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
<tr>
<td>OutputLSAAcks</td>
<td>LSA</td>
<td>Number of LSA sent in LSA acknowledgements.</td>
<td>Range is from 0 to 4294967295.</td>
</tr>
</tbody>
</table>

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

*Table 13: Entity Instances and Monitoring Commands*

<table>
<thead>
<tr>
<th>Entity</th>
<th>Command Description</th>
</tr>
</thead>
</table>
| BGP                 | Use the `performance-mgmt apply monitor bgp` command to enable entity instance monitoring for a BGP entity instance.  
**Syntax:**  
```plaintext
performance-mgmt
    apply monitor
    bgp
    ip-address
    template-name | 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:**  
```plaintext
performance-mgmt
    apply
    monitor
    interface
data-rates
type
    interface-path-id {template-name | default}
```
<p>|                     | RP/0/RP0/CPU0:Router(config)# performance-mgmt apply monitor bgp 10.12.0.4 default |
|                     | RP/0/RP0/CPU0:Router(config)# performance-mgmt apply monitor interface data-rates    |
|                     | HundredGigE 0/3/0/24 default                                                        |</p>
<table>
<thead>
<tr>
<th>Entity</th>
<th>Command Description</th>
</tr>
</thead>
</table>
| 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:** <br>```
performance-mgmt
  apply
  monitor
  interface
  basic-counters
  type
  interface-path-id {template-name | default}
```  <br>RP/0/RP0/CPU0:Router(config)# performance-mgmt apply monitor interface basic-counters HundredGigE 0/3/0/24 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:** <br>```
performance-mgmt
  apply
  monitor
  interface
  generic-counters
  type
  interface-path-id {template-name | default}
```  <br>RP/0/RP0/CPU0:Router(config)# performance-mgmt apply monitor interface generic-counters HundredGigE 0/3/0/24 default |
| MPLS LDP                  | Use the `performance-mgmt apply monitor mpls ldp` command to enable entity instance monitoring for an MPLS LDP entity instance. **Syntax:** <br>```
performance-mgmt
  apply
  monitor
  mpls
  ldp
  ip-address {template-name | default}
```  <br>RP/0/RP0/CPU0:Router(config)# performance-mgmt apply monitor mpls ldp 10.34.64.154 default |
<table>
<thead>
<tr>
<th>Entity</th>
<th>Command Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node CPU</td>
<td>Use the <strong>performance-mgmt apply monitor node cpu</strong> command to enable entity instance monitoring for a node CPU entity instance.</td>
</tr>
<tr>
<td></td>
<td><strong>Syntax:</strong></td>
</tr>
<tr>
<td></td>
<td>```</td>
</tr>
<tr>
<td></td>
<td>performance-mgmt apply monitor node cpu</td>
</tr>
<tr>
<td></td>
<td>location node-id {template-name</td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:Router(config)# performance-mgmt apply monitor node cpu location 0/RP0/CPU0 default</td>
</tr>
<tr>
<td>Node Memory</td>
<td>Use the <strong>performance-mgmt apply monitor node memory</strong> command to enable entity instance monitoring for a node memory entity instance.</td>
</tr>
<tr>
<td></td>
<td><strong>Syntax:</strong></td>
</tr>
<tr>
<td></td>
<td>```</td>
</tr>
<tr>
<td></td>
<td>performance-mgmt apply monitor node memory</td>
</tr>
<tr>
<td></td>
<td>location node-id {template-name</td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:Router(config)# performance-mgmt apply monitor node memory location 0/RP0/CPU0 default</td>
</tr>
<tr>
<td>Node Process</td>
<td>Use the <strong>performance-mgmt apply monitor node process</strong> command to enable entity instance monitoring collection for a node process entity instance.</td>
</tr>
<tr>
<td></td>
<td><strong>Syntax:</strong></td>
</tr>
<tr>
<td></td>
<td>```</td>
</tr>
<tr>
<td></td>
<td>performance-mgmt apply monitor node process</td>
</tr>
<tr>
<td></td>
<td>location node-id pid {template-name</td>
</tr>
<tr>
<td></td>
<td>RP/0/RP0/CPU0:Router(config)# performance-mgmt apply monitor node process location p 0/RP0/CPU0 275 default</td>
</tr>
</tbody>
</table>
Performance Management: Details
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 52
- Information About Configuring and Managing Embedded Event Manager Policies, on page 52
- How to Configure and Manage Embedded Event Manager Policies, on page 61
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 53

Writing an EEM script includes the following steps:

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

Embedded Event Manager Policy Tcl Command Extension Categories

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

<table>
<thead>
<tr>
<th>Category</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>EEM event Tcl command extensions(three types: event information, event registration, and event publish)</td>
<td>These Tcl command extensions are represented by the <code>event_register_xxx</code> family of event-specific commands. There is a separate event information Tcl command extension in this category as well: <code>event_reqinfo</code>. 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 <code>event_publish</code> that publishes an application-specific event.</td>
</tr>
<tr>
<td>EEM action Tcl command extensions</td>
<td>These Tcl command extensions (for example, <code>action_syslog</code>) 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.</td>
</tr>
<tr>
<td>EEM utility Tcl command extensions</td>
<td>These Tcl command extensions are used to retrieve, save, set, or modify application information, counters, or timers.</td>
</tr>
<tr>
<td>EEM system information Tcl command extensions</td>
<td>These Tcl command extensions are represented by the <code>sys_reqinfo_xxx</code> family of system-specific information commands. These commands are used by a policy to gather system information.</td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>Two-Character Abbreviation</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>ap</td>
<td>event_register_appl</td>
</tr>
<tr>
<td>ct</td>
<td>event_register_counter</td>
</tr>
<tr>
<td>st</td>
<td>event_register_stat</td>
</tr>
<tr>
<td>no</td>
<td>event_register_none</td>
</tr>
<tr>
<td>oi</td>
<td>event_register_oir</td>
</tr>
<tr>
<td>pr</td>
<td>event_register_process</td>
</tr>
<tr>
<td>sl</td>
<td>event_register_syslog</td>
</tr>
<tr>
<td>tm</td>
<td>event_register_timer</td>
</tr>
<tr>
<td>ts</td>
<td>event_register_timer_subscriber</td>
</tr>
<tr>
<td>wd</td>
<td>event_register_wdsysmon</td>
</tr>
</tbody>
</table>

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 16: Embedded Event Manager Built-In Actions

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

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 17: System Manager Event Detector Event Monitoring Requests

<table>
<thead>
<tr>
<th>Embedded Event Manager Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal process termination EEM event—built in</td>
<td>Occurs when a process matching the screening criteria terminates.</td>
</tr>
<tr>
<td>Abnormal process termination EEM event—built in</td>
<td>Occurs when a process matching the screening criteria terminates abnormally.</td>
</tr>
<tr>
<td>Process startup EEM event—built in</td>
<td>Occurs when a process matching the screening criteria starts.</td>
</tr>
</tbody>
</table>

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 18: Syslog Event Detector Event Monitoring Requests

<table>
<thead>
<tr>
<th>Embedded Event Manager Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syslog message EEM event</td>
<td>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.</td>
</tr>
</tbody>
</table>
**None Event Detector**

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

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

**Watchdog System Monitor Event Detector**

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

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

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

---

**Note**

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

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

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

**Table 19: Watchdog System Monitor Event Detector Requests**

<table>
<thead>
<tr>
<th>Embedded Event Manager Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process percent CPU EEM event—built in</td>
<td>Occurs when the CPU time for a specified process is either greater than or equal to a specified maximum percentage of available CPU time or is less than or equal to a specified minimum percentage of available CPU time.</td>
</tr>
<tr>
<td>Total percent CPU EEM event—built in</td>
<td>Occurs when the CPU time for a specified processor complex is either greater than or equal to a specified maximum percentage of available CPU time or is less than or equal to a specified minimum percentage of available CPU time.</td>
</tr>
</tbody>
</table>
### Embedded Event Manager Event

<table>
<thead>
<tr>
<th>Embedded Event Manager Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process percent memory EEM event—built in</td>
<td>Occurs when the memory used for a specified process has either increased or decreased by a specified value.</td>
</tr>
<tr>
<td>Total percent available Memory EEM event—built in</td>
<td>Occurs when the available memory for a specified processor complex has either increased or decreased by a specified value.</td>
</tr>
<tr>
<td>Total percent used memory EEM event—built in</td>
<td>Occurs when the used memory for a specified processor complex has either increased or decreased by a specified value.</td>
</tr>
</tbody>
</table>

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

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

### Distributed Event Detectors

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

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

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

### Embedded Event Manager Event Scheduling and Notification

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

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

### Reliability Statistics

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

#### Hardware Card Reliability Metric Data

Hardware card reliability metric data is recorded in a table indexed by disk ID.
Data maintained by the hardware card is as follows:

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

**Process Reliability Metric Data**

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

Process terminations include the following cases:

- Normal termination—Process exits with an exit value equal to 0.
- Abnormal termination by process—Process exits with an exit value not equal to 0.
- Abnormal termination by Linux—Linux operating system aborts 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
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
```

```
<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>_email_to</td>
<td><a href="mailto:beta@cisco.com">beta@cisco.com</a></td>
</tr>
<tr>
<td>2</td>
<td>_cron_entry</td>
<td>0-59/2 0-23/1 * * 0-7</td>
</tr>
<tr>
<td>3</td>
<td>_email_from</td>
<td><a href="mailto:beta@cisco.com">beta@cisco.com</a></td>
</tr>
</tbody>
</table>
```

Registering Embedded Event Manager Policies

You should register an EEM policy to run a policy when an event is triggered. Registering an EEM policy is performed with the `event manager policy` command. An EEM script is available to be scheduled by the EEM until the `no` form of this command is entered. Prior to registering a policy, display EEM policies that are available to be registered with the `show event manager policy available` command.
The EEM schedules and runs policies on the basis of an event specification that is contained within the policy itself. When the `event manager policy` command is invoked, the EEM examines the policy and registers it to be run when the specified event occurs.

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

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

---

**Note**

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

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

---

**Configuration Example**

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

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

---

**How to Write Embedded Event Manager Policies Using Tcl**

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

This section contains these tasks:

**Registering and Defining an EEM Tcl Script**

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

---

**Note**

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

---

**Configuration Example**

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

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

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

Suspending EEM Policy Execution

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

Configuration Example

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

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

Specifying a Directory for Storing EEM Policies

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

Configuration Example

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

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

Programming EEM Policies with Tcl

Perform this task to help you program a policy using Tcl command extensions. We recommend that you copy an existing policy and modify it. There are two required parts that must exist in an EEM Tcl policy: the event_register Tcl command extension and the body. For detailed information about the Tcl policy structure and requirements, see EEM Policies Using TCL: Details, on page 73
SUMMARY STEPS

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

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> show event manager policy available [system</td>
<td>user]</td>
</tr>
<tr>
<td>Example: RP/0/RP0/CPU0:Router# show event manager policy available</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> Cut and paste the contents of the sample policy displayed on the screen to a text editor.</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> Define the required event_register Tcl command extension.</td>
<td>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:</td>
</tr>
<tr>
<td></td>
<td>• event_register_appl</td>
</tr>
<tr>
<td></td>
<td>• event_register_counter</td>
</tr>
<tr>
<td></td>
<td>• event_register_stat</td>
</tr>
<tr>
<td></td>
<td>• event_register_wdsysmon</td>
</tr>
<tr>
<td></td>
<td>• event_register_oir</td>
</tr>
<tr>
<td></td>
<td>• event_register_process</td>
</tr>
<tr>
<td></td>
<td>• event_register_syslog</td>
</tr>
<tr>
<td></td>
<td>• event_register_timer</td>
</tr>
</tbody>
</table>
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.

This is an optional step. Must defines is a section of the policy that tests whether any EEM environment variables that are required by the policy are defined before the recovery actions are taken. The must defines section is not required if the policy does not use any EEM environment variables. EEM environment variables for EEM scripts are Tcl global variables that are defined external to the policy before the policy is run. To define an EEM environment variable, use the EEM configuration command `event manager environment`. By convention, all Cisco EEM environment variables begin with "_" (an underscore). To avoid future conflict, customers are urged not to define new variables that start with "_".
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note</strong></td>
<td>You can display the Embedded Event Manager environment variables set on your system by using the <code>show event manager environment</code> 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.</td>
</tr>
</tbody>
</table>

- **_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.
- 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).
<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 8</strong></td>
<td>Check the exit status to determine whether or not to apply the default action for this event, if a default action exists.</td>
<td>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.</td>
</tr>
<tr>
<td><strong>Step 9</strong></td>
<td>Set Cisco Error Number (_cerrno) Tcl global variables.</td>
<td>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).</td>
</tr>
</tbody>
</table>
| **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 15: Two-Character Abbreviation Specification, on page 54) 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 54. |
| **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. |
| **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** | commit | |
Creating an EEM User Tcl Library Index

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

SUMMARY STEPS

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

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>On your workstation (UNIX, Linux, PC, or Mac) create a library directory and copy the Tcl library files into the directory.</td>
</tr>
<tr>
<td><strong>Step 15</strong></td>
<td>Cause the policy to execute, and observe the policy.</td>
</tr>
<tr>
<td><strong>Step 16</strong></td>
<td>Use debugging techniques if the policy does not execute correctly.</td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Step 2</strong> tclsh</td>
<td>Enters the Tcl shell.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>workstation% tclsh</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> auto_mkindex directory_name *.tcl</td>
<td>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.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td>workstation% auto_mkindex eem_library *.tcl</td>
<td>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:</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>tclIndex</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td># Tcl autoload index file, version 2.0</td>
</tr>
<tr>
<td></td>
<td># This file is generated by the &quot;auto_mkindex&quot; command</td>
</tr>
<tr>
<td></td>
<td># and sourced to set up indexing information for one or</td>
</tr>
<tr>
<td></td>
<td># more commands. Typically each line is a command that</td>
</tr>
<tr>
<td></td>
<td># sets an element in the auto_index array, where the</td>
</tr>
<tr>
<td></td>
<td># element name is the name of a command and the value is</td>
</tr>
<tr>
<td></td>
<td># a script that loads the command.</td>
</tr>
<tr>
<td></td>
<td>set auto_index(test1) [list source [file join $dir lib1.tcl]]</td>
</tr>
<tr>
<td></td>
<td>set auto_index(test2) [list source [file join $dir lib1.tcl]]</td>
</tr>
<tr>
<td></td>
<td>set auto_index(test3) [list source [file join $dir lib2.tcl]]</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>Copy a user-defined EEM policy file written in Tcl to the directory used for storing user-defined EEM policies on the target router.</td>
</tr>
<tr>
<td></td>
<td>The directory can be the same directory used in step 4.</td>
</tr>
<tr>
<td></td>
<td>The following example user-defined EEM policy can be used to test the Tcl library support in EEM: libtest.tcl</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>::cisco::eem::event_register_none</td>
</tr>
<tr>
<td></td>
<td>namespace import ::cisco::eem::*</td>
</tr>
</tbody>
</table>
Purpose Command or Action

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

Step 7 event manager directory user library path
Example:
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

Step 9 event manager policy policy-name username [persist-time [seconds | infinite] | type [system | user]]
Example:
RP/0/RP0/CPU0:Router(config)# event manager policy libtest.tcl username user_a

Step 10 event manager run policy [argument]
Example:
RP/0/RP0/CPU0:Router(config)# event manager run libtest.tcl

Step 11 commit

Creating an EEM User Tcl Package Index

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

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

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

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

### SUMMARY STEPS

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

### DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>On your workstation (UNIX, Linux, PC, or Mac) create a library directory and copy the Tcl package files into the directory.</td>
<td>—</td>
</tr>
<tr>
<td>2</td>
<td><code>tclsh</code></td>
<td>Enters the Tcl shell.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>workstation% tclsh</code></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td><code>pkg_mkindex directory_name *.tcl</code></td>
<td>Use the <code>pkg_mkindex</code> 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 <code>pkg_mkindex</code> 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 <code>pkg_mkindex</code> command</td>
</tr>
</tbody>
</table>
## Creating an EEM User Tcl Package Index

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>in a directory determines which Tcl package file or files are indexed using a specific pkgIndex. The following example pkgIndex is created when some Tcl package files are in a library file directory and the pkg_mkindex command is run: <strong>pkgIndex</strong></td>
<td></td>
</tr>
</tbody>
</table>

```tcl
# 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]]
```

<table>
<thead>
<tr>
<th>Step 4</th>
<th>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.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 5</td>
<td>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: <strong>packagetest.tcl</strong></td>
</tr>
</tbody>
</table>

```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?"
```

<table>
<thead>
<tr>
<th>Step 6</th>
<th>configure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 7</td>
<td>event manager directory user library path Example:</td>
</tr>
<tr>
<td></td>
<td>Specifies the EEM user library directory; this is the directory to which the files in step 4 were copied.</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>event manager directory user policy path</td>
<td>Specifies the EEM user policy directory; this is the directory to which the file in step 5 was copied.</td>
</tr>
<tr>
<td>event manager policy policy-name username username [persist-time [seconds</td>
<td>infinite] type [system</td>
</tr>
<tr>
<td>event manager run policy [argument]</td>
<td>Manually runs an EEM policy.</td>
</tr>
<tr>
<td>commit</td>
<td></td>
</tr>
</tbody>
</table>

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:

```tcl
::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:

```tcl
# 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
} elseif {![info exists _email_from]} {
    set result "Policy cannot be run: variable _email_from has not been set"
    error $result $errorInfo
} elseif {![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:

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

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

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

**EEM Entry Status**

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

Entry status designations may use one of three possible values:

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

**EEM Exit Status**

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

**EEM Policies and Cisco Error Number**

Some EEM Tcl command extensions set a Cisco Error Number Tcl global variable known as _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:

\[\text{XYSSSSSSSSSSSSSEEEEEEEEPPPPPPP}\]

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

**Table 20: _cerrno: 32-Bit Error Return Value Variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XY</td>
<td>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 Table 21: Error Class Encodings, on page 76 for the four possible error class encodings specific to this variable.</td>
</tr>
<tr>
<td>SSSSSSSSSSSSS</td>
<td>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.</td>
</tr>
<tr>
<td>EEEEEEEE</td>
<td>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.</td>
</tr>
</tbody>
</table>

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:

\[10000110001000100011100101110\]

The variable, XY, references the possible error class encodings shown in this table.
Table 21: Error Class Encodings

<table>
<thead>
<tr>
<th>Error Return Value</th>
<th>Error Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>CERR_CLASS_SUCCESS</td>
</tr>
<tr>
<td>01</td>
<td>CERR_CLASS_INFO</td>
</tr>
<tr>
<td>10</td>
<td>CERR_CLASS_WARNING</td>
</tr>
<tr>
<td>11</td>
<td>CERR_CLASS_FATAL</td>
</tr>
</tbody>
</table>

An error return value of zero means SUCCESS.
CHAPTER 7

Implementing IP Service Level Agreements

IP Service Level Agreements (IP SLAs) is a portfolio of technologies embedded in most devices that run Cisco IOS XR Software, which allows the user to perform network assessments, verify quality of service (QoS), ease the deployment of new services, and assist administrators with network troubleshooting and so on.

This chapter provides information about this feature and the different steps involved in configuring it.

Table 22: Feature History Table for IP SLA

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Release 6.3.1</td>
<td>Two-Way Active Measurement Protocol (TWAMP) was introduced.</td>
</tr>
</tbody>
</table>

This chapter covers the following topics:

- IP Service Level Agreements Technology Overview, on page 77
- Prerequisites for Implementing IP Service Level Agreements, on page 79
- Restrictions for Implementing IP Service Level Agreements, on page 80
- Measuring Network Performance with IP Service Level Agreements, on page 80
- Two-Way Active Measurement Protocol (TWAMP), on page 82
- TWAMP-Light, on page 85

IP Service Level Agreements Technology Overview

IP SLA uses active traffic monitoring, which generates traffic in a continuous, reliable, and predictable manner to measure network performance. IP SLA sends data across the network to measure performance between multiple network locations or across multiple network paths. It simulates network data and IP services, and collects network performance information in real time. The following information is collected:

- Response times
- One-way latency, jitter (inter-packet delay variance)
- Packet loss
- Network resource availability
IP SLA performs active monitoring by generating and analyzing traffic to measure performance, either between the router or from a router to a remote IP device such as a network application server. Measurement statistics, which are provided by the various IP SLA operations, are used for troubleshooting, problem analysis, and designing network topologies.

This section covers the following topics:

## Service Level Agreements

Internet commerce has grown significantly in the past few years as the technology has advanced to provide faster, more reliable access to the Internet. Many companies need online access and conduct most of their business on line and any loss of service can affect the profitability of the company. Internet service providers (ISPs) and even internal IT departments now offer a defined level of service—a service level agreement—to provide their customers with a degree of predictability.

Network administrators are required to support service level agreements that support application solutions. Figure 4: Scope of Traditional Service Level Agreement Versus IP SLA, on page 78 shows how IP SLA has taken the traditional concept of Layer 2 service level agreements and applied a broader scope to support end-to-end performance measurement, including support of applications.

**Note**

- Provided that the application and the IP-SLA processing rates support it, you can specify the flow rate for IP-SLA flow entries to up to 1500.

- To enable high performance for IP-SLA operations, avoid reuse of same source and destination ports for multiple IP SLA operations on the same device, especially when the scale is huge

![Figure 4: Scope of Traditional Service Level Agreement Versus IP SLA](image)

This table lists the improvements with IP SLA over a traditional service level agreement.

### Table 23: IP SLA Improvements over a Traditional Service Level Agreement

<table>
<thead>
<tr>
<th>Type of Improvement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>End-to-end measurements</td>
<td>The ability to measure performance from one end of the network to the other allows a broader reach and more accurate representation of the end-user experience.</td>
</tr>
<tr>
<td>Sophistication</td>
<td>Statistics, such as delay, jitter, packet sequence, Layer 3 connectivity, and path and download time, that are divided into bidirectional and round-trip numbers provide more data than just the bandwidth of a Layer 2 link.</td>
</tr>
</tbody>
</table>
Benefits of IP Service Level Agreements

This table lists the benefits of implementing IP SLA.

Table 24: List of Benefits for IP SLA

<table>
<thead>
<tr>
<th>Benefit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP SLA monitoring</td>
<td>Provides service level agreement monitoring, measurement, and verification.</td>
</tr>
<tr>
<td>Network performance monitoring</td>
<td>Measure the jitter, latency, or packet loss in the network. In addition, IP SLA provides continuous, reliable, and predictable measurements along with proactive notification.</td>
</tr>
<tr>
<td>IP service network health assessment</td>
<td>Verifies that the existing QoS is sufficient for the new IP services.</td>
</tr>
<tr>
<td>Troubleshooting of network operation</td>
<td>Provides consistent, reliable measurement that immediately identifies problems and saves troubleshooting time.</td>
</tr>
</tbody>
</table>

Prerequisites for Implementing IP Service Level Agreements

Knowledge of general networking protocols and your specific network design is assumed. Familiarity with network management applications is helpful. We do not recommend scheduling all the operations at the same time as this could negatively affect your performance.

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.
Restrictions for Implementing IP Service Level Agreements

- The maximum number of IP SLA operations that is supported by Cisco IOS XR Software is 2048.
- The maximum number of IP SLA configurable operations that is supported by Cisco IOS XR Software is 2000.
- We do not recommend scheduling all the operations at the same start time as this may affect the performance. At the same start time, not more than 10 operations per second should be scheduled. We recommend using the start after configuration.

Note
Setting the frequency to less than 60 seconds will increase the number of packets sent. But this could negatively impact the performance of IP SLA operation when scheduled operations have same start time.

- IP SLA is not HA capable.
- Consider the following guidelines before configuring the frequency, timeout, and threshold commands.
- Control disabled mode gives a better IP-SLA scale when compared to Control Enabled mode.

Measuring Network Performance with IP Service Level Agreements

IP SLA uses generated traffic to measure network performance between two networking devices, such as routers. Figure 5: IP SLA Operations, on page 80 shows how IP SLA starts when the IP SLA device sends a generated packet to the destination device. After the destination device receives the packet and if the operation uses an IP SLA component at the receiving end (for example, IP SLA Responder), the reply packet includes information about the delay at the target device. The source device uses this information to improve the accuracy of the measurements. An IP SLA operation is a network measurement to a destination in the network from the source device using a specific protocol, such as User Datagram Protocol (UDP) for the operation.

Figure 5: IP SLA Operations
To implement IP SLA network performance measurement, perform these tasks:

1. Enable the IP SLA Responder, if appropriate.
2. Configure the required IP SLA operation type.
3. Configure any options available for the specified IP SLA operation type.
4. Configure reaction conditions, if required.
5. Schedule the operation to run. Then, let the operation run for a period of time to gather statistics.
6. Display and interpret the results of the operation using Cisco IOS-XR Software CLI, XML, or an NMS system with SNMP.

The following topics are covered in this section:

**IP SLA Responder and IP SLA Control Protocol**

The IP SLA Responder is a component embedded in the destination Cisco routing device that allows the system to anticipate and respond to IP SLA request packets. The IP SLA Responder provides enhanced accuracy for measurements. The patented IP SLA Control Protocol is used by the IP SLA Responder, providing a mechanism through which the responder is notified on which port it should listen and respond. Only a Cisco IOS-XR software device or other Cisco platforms can be a source for a destination IP SLA Responder.

*Figure 5: IP SLA Operations, on page 80 shows where the IP SLA Responder fits relative to the IP network. The IP SLA Responder listens on a specific port for control protocol messages sent by an IP SLA operation. Upon receipt of the control message, the responder enables the UDP port specified in the control message for the specified duration. During this time, the responder accepts the requests and responds to them. The responder disables the port after it responds to the IP SLA packet or packets, or when the specified time expires. For added security, MD5 authentication for control messages is available.*

---

**Note**

The IP SLA responder needs at least one second to open a socket and program Local Packet Transport Services (LPTS). Therefore, configure the IP SLA timeout to at least 2000 milli seconds.

The IP SLA Responder must be used with the UDP jitter operation. If services that are already provided by the target router are chosen, the IP SLA Responder need not be enabled. For devices that are not Cisco devices, the IP SLA Responder cannot be configured, and the IP SLA can send operational packets only to services native to those devices.

**Response Time Computation for IP SLA**

Because of other high-priority processes, routers can take tens of milliseconds to process incoming packets. The delay affects the response times, because the reply to test packets might be sitting in a queue while waiting to be processed. In this situation, the response times would not accurately represent true network delays. IP SLA minimizes these processing delays on the source router and on the target router (if IP SLA Responder is being used) to determine true round-trip times. Some IP SLA probe packets contain delay information that are used in the final computation to make measurements more accurate.

When enabled, the IP SLA Responder allows the target device to take two time stamps, both when the packet arrives on the interface and again just as it is leaving, and accounts for it when calculating the statistics. This time stamping is made with a granularity of submilliseconds.
Figure 6: IP SLA Responder Time Stamping, on page 82 shows how the responder works. T3 is the time the reply packet is sent at the IP SLA Responder node, and T1 is the time the request is sent at the source node. Four time stamps are taken to make the calculation for round-trip time. At the target router, with the responder functionality enabled, time stamp 2 (TS2) is subtracted from time stamp 3 (TS3) to produce the time spent processing the test packet as represented by delta. This delta value is then subtracted from the overall round-trip time. Notice that the same principle is applied by IP SLA on the source router on which the incoming time stamp 4 (TS4) is taken in a high-priority path to allow for greater accuracy.

**IP SLA Operation Scheduling**

After an IP SLA operation is configured, you must schedule the operation to begin capturing statistics and collecting error information. When scheduling an operation, the operation starts immediately or starts at a certain month and day. In addition, an operation can be scheduled to be in pending state, which is used when the operation is a reaction (threshold) operation waiting to be triggered. Normal scheduling of IP SLA operations lets you schedule one operation at a time.

**Two-Way Active Measurement Protocol (TWAMP)**


**Advantages of TWAMP**

- TWAMP enables complete IP performance measurement.
- TWAMP provides a flexible choice of solutions as it supports all devices deployed in the network.

The following topics are covered in this section:

**The TWAMP Entities**

The TWAMP system consists of 4 logical entities:

- server - manages one or more TWAMP sessions and also configures per-session ports in the end-points.
- session-reflector - reflects a measurement packet as soon as it receives a TWAMP test packet.
- control-client - initiates the start and stop of TWAMP test sessions.
- session-sender - instantiates the TWAMP test packets sent to the session reflector.

The below diagram shows TWAMP implementation where TWAMP runs on two separate hosts. One plays the roles of Control-Client and Session-Sender, and the other plays the roles of Server and Session-Reflector.
The router supports Session-Server and Session Reflector functionality only. Using TWAMP, the IP performance of underlying transport can be measured through cooperation between network elements that include TWAMP support.

Figure 7: The TWAMP Entities

**TWAMP Protocols**

The TWAMP protocol includes three distinct message exchange categories, they are:

- **Connection set-up exchange**: Messages establish a session connection between the Control-Client and the Server. First the identities of the communicating peers are established via a challenge response mechanism. The Server sends a randomly generated challenge, to which the Control-Client then sends a response by encrypting the challenge using a key derived from the shared secret. Once the identities are established, the next step negotiates a security mode that is binding for the subsequent TWAMP-Control commands as well as the TWAMP-Test stream packets.

  **Note** A server can accept connection requests from multiple control clients.

- **TWAMP-control exchange**: The TWAMP-Control protocol runs over TCP and is used to instantiate and control measurement sessions. Unlike the Connection setup exchanges, the TWAMP-Control commands can be sent multiple times. However, the messages cannot occur out of sequence although multiple request-session commands can be sent before a session-start command. The sequence of commands is as follows:
  - request-session
  - start-session
  - stop-session

- **TWAMP-test stream exchange**: The TWAMP-Test runs over UDP and exchanges TWAMP-Test packets between Session-Sender and Session-Reflector. These packets include timestamp fields that contain the instant of packet egress and ingress. In addition, each packet includes an error-estimate that indicates the synchronization skew of the sender (session-sender or session-reflector) with an external time source (e.g. GPS or NTP). The packet also includes a Sequence Number.
TWAMP-Control and TWAMP-test stream, have three security modes: unauthenticated, authenticated, and encrypted.

**Restrictions of TWAMP on the Router**

- This router supports only Session-Server and Session Reflector functionality.
- Hardware Timestamp feature which provides greater accuracy is not supported.

**Configuring TWAMP on the Router**

**Configuration of Session-Server**

```
Router# configure
Router(config)#  ipsla server twamp
Router(config-ipsla-server-twamp)#  port 862
Router(config-ipsla-server-twamp)#  commit
```

**Configuration of Session-Reflector**

```
Router# configure
Router(config)#  ipsla responder twamp
Router(config-twamp-ref)#  commit
```

**Running Configuration**

```
ipsla  
responder  
twamp  
!  
!  
server twamp  
port 862  
!  
```

**Verification of TWAMP**

The status of the TWAMP feature can be verified using the command `show ipsla twamp status`

```
Router# show ipsla twamp status
Thu Aug 17 12:42:38.923 IST
TWAMP Server is enabled
TWAMP Server port : 862
TWAMP Reflector is enabled
```

The TWAMP session can be verified using the command `show ipsla twamp session`

```
Router# show ipsla twamp session
IP SLAs Responder TWAMP is: Enabled
Recvr Addr: 10.5.139.11
Recvr Port: 7222
Sender Addr: 172.27.111.233
Sender Port: 33243
Session Id: 10.5.139.11:70929508:88F7A620
Connection Id: 0
```
The TWAMP test session based on source ip-address can be verified using the command  

```
show ipsla twamp session source-ip <source ip-address> source-port <source port-number>
```

RP/0/0/CPU0:ios# show ipsla twamp session source-ip 172.27.111.233 source-port 33286
IP SLAs Responder TWAMP is: Enabled
Recv Addr: 10.5.139.11
Recv Port: 6198
Sender Addr: 172.27.111.233
Sender Port: 33286
Session Id: 10.5.139.11:71804476:F2721505
Connection Id: D
Mode: Unauthorized
DSCP: 0
Pad Length: 0
Number of Packets Received: 8867

**TWAMP-Light**

TWAMP-light is a light-weight model of TWAMP which eliminates the need for a control session. Unlike the TWAMP feature, you need to configure the parameters of the TWAMP-light test-session at both end devices. So this removes the overhead of establishing and terminating the control session. In addition, the server entity is not required on the reflector device thereby reducing the overhead of maintaining the server.

**Restrictions of TWAMP-Light on the Cisco NCS 5500 Series Router**

- Only 5 TWAMP-light test sessions can run simultaneously and the test-session ID should be between 1 and 5.
- If the TWAMP-light test-session runs on a Virtual Routing and Forwarding (VRF) instance, then the session will work only when the same VRF is also configured on the interface.
- Once you configure a TWAMP-light test-session on a device, it opens a permanent port, which will remain open until you delete the configuration for TWAMP-light. If you do not prefer this behaviour, then you should configure a timeout for the TWAMP-light test-session so that the session will be inactive after the timeout period.
- When there are two clients with two different test-sessions with the same local IP address and local port under the same VRF, there will be only one underlying socket at the responder. In such a scenario, due to UDP restrictions it is not possible to support the maximum number of packets for these two clients. This causes the performance to be impacted. Therefore, any two test-sessions cannot have the same local IP address and local port under the same VRF.

**Configuring TWAMP-Light on the NCS 5500 Series Router**

```
Router# configure
Router (config)# ipsla
Router (config-ipsla)# responder
Router (config-ipsla-resp)# twamp-light test-session 1
Router (config-ipsla-resp)# local-ip 192.0.2.10 local-port 13001 remote-ip 192.0.2.186 remote-port 13002 vrf default
Router (config-ipsla-resp)# timeout 60
Router (config-ipsla-resp)# commit
```
Running Configuration

ipsla
    responder
twamp-light test-session 1
    local-ip 192.0.2.10 local-port 13001 remote-ip 192.0.2.186 remote-port 13002 vrf default
timeout 60
!
!
!

Verification of TWAMP-Light on the NCS 5500 Series Router

The TWAMP-light session can be verified using the command `show ipsla twamp session`. The output of the command shows the state of the session using the **Session status** field as shown below:

```
Router# show ipsla twamp session
***** TWAMP Sessions *****
No records matching query found
***** TWAMP-LIGHT Sessions *****
**Session status: Active**
Recvr Addr: 192.0.2.10
Recvr Port: 13001
Sender Addr: 192.0.2.186
Sender Port: 13002
Sender VRF Name: default
Session ID: 1
Mode: Unauthenticated
Number of Packets Received: 0
Session timeout: 60
```