System Monitoring Configuration Guide for Cisco 8000 Series Routers, IOS XR Release 26.1.x

First Published: 2026-02-28

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

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

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

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

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

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

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

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

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

CONTENTS

PREFACE

CHAPTER 1

CHAPTER 2

CHAPTER 3

Preface

The *System Monitoring Configuration Guide for Cisco 8000 Series Routers* preface contains these sections:

Changes to this Document

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

Date	**Summary**
February 2026	Initial release of this document.

Communications, Services, and Additional Information

- To receive timely, relevant information from Cisco, sign up at Cisco Profile Manager.
- To get the business results 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 DevNet.
- To obtain general networking, training, and certification titles, visit Cisco Press.
- To find warranty information for a specific product or product family, access Cisco Warranty Finder.

Cisco Bug Search Tool

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

CHAPTER 1

New and Changed System Monitoring Features

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

System Monitoring Features Added or Modified in IOS XR Release 26.x.x

Feature	Description	Changed in Release	Where Documented
SSD health metrics	This feature was introduced.	Release 26.1.1	SSD health metrics

CHAPTER 2

YANG Data Models for System Monitoring Features

This chapter provides information about the YANG data models for System Monitoring features.

Using YANG Data Models

Cisco IOS XR supports a programmatic way of configuring and collecting operational data of a network device using YANG data models. Although configurations using CLIs are easier and human-readable, automating the configuration using model-driven programmability results in scalability.

The data models are available in the release image, and are also published in the Github repository. Navigate to the release folder of interest to view the list of supported data models and their definitions. Each data model defines a complete and cohesive model, or augments an existing data model with additional XPaths. To view a comprehensive list of the data models supported in a release, navigate to the **Available-Content.md** file in the repository.

You can also view the data model definitions using the YANG Data Models Navigator tool. This GUI-based and easy-to-use tool helps you explore the nuances of the data model and view the dependencies between various containers in the model. You can view the list of models supported across Cisco IOS XR releases and platforms, locate a specific model, view the containers and their respective lists, leaves, and leaf lists presented visually in a tree structure. This visual tree form helps you get insights into nodes that can help you automate your network.

To get started with using the data models, see the *Programmability Configuration Guide*.

CHAPTER 3

Implementing System Logging

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

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

Feature History for Implementing System Logging

Release	Modification
Release 7.0.11	This feature was introduced.

Implementing System Logging

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

Syslog Message Format

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

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

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

Table 1: Format of Syslog Messages

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

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

Syslog Message Severity Levels

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

Note Utility word count lines are used to calculate the number of logs present in the IOS XR syslog buffer. When there is an increase in the inflow of logs from the routers, if you are executing the **show logging** command, the number of lines calculated by using the word count utility may exceed the value set for login buffer size. You can set the login buffer size by using the **logging buffer entries-count** command.

Table 2: Syslog Message Severity Levels

Severity Keyword	Level	Description
emergencies	0	System unusable
alert	1	Immediate action needed
critical	2	Critical conditions
errors	3	Error conditions
warnings	4	Warning conditions
notifications	5	Normal but significant condition

Severity Keyword	Level	Description
informational	6	Informational messages only
debugging	7	Debugging messages

Prerequisites for Configuring System Logging

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

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

Syslog Messages Sent to Syslog Servers

The Cisco IOS XR Software provides these features to help manage syslog messages sent to syslog servers:

UNIX System Logging Facilities

You can configure the syslog facility in which syslog messages are sent by using the **logging facility** command. Consult the operator manual for your UNIX operating system for more information about these UNIX system facilities. The syslog format is compatible with Berkeley Standard Distribution (BSD) UNIX version 4.3.

This table describes the facility type keywords that can be supplied for the *type* argument.

Table 3: Logging Facility Type Keywords

Facility Type Keyword	Description
auth	Indicates the authorization system.
cron	Indicates the cron facility.
daemon	Indicates the system daemon.
kern	Indicates the Kernel.
local0–7	Reserved for locally defined messages.
lpr	Indicates line printer system.
mail	Indicates mail system.
news	Indicates USENET news.
sys9	Indicates system use.
sys10	Indicates system use.

Facility Type Keyword	Description
sys11	Indicates system use.
sys12	Indicates system use.
sys13	Indicates system use.
sys14	Indicates system use.
syslog	Indicates the system log.
user	Indicates user process.
uucp	Indicates UNIX-to-UNIX copy system.

Hostname Prefix Logging

To help manage system logging messages sent to syslog servers, 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.

To append a hostname prefix to syslog messages sent to syslog servers, use the **logging hostname** command in mode.

Configuration Example

This example shows how to add the hostname prefix host1 to messages sent to the syslog servers from the router.

```
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# logging hostnameprefix host1
RP/0/RP0/CPU0:Router(config)# commit
```

Syslog Source Address Logging

By default, a syslog message contains the IP address (IPv4 and IPv6 are supported) of the interface it uses to leave the router when sent to syslog servers. To set all syslog messages to contain the same IP address, regardless of which interface the syslog message uses to exit the router, use the **logging source-interface** command in mode.

Configuration Example

This example shows how to specify that the IP address for HundredGigE interface 0/1/0/0 be set as the source IP address for all messages.

```
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# logging source-interface HundredGigE interface 0/1/0/0
RP/0/RP0/CPU0:Router(config)# commit
```

Logging History Table

If you have enabled syslog messages traps to be sent to a Simple Network Management Protocol (SNMP) network management station (NMS) with the **snmp-server enable traps syslog** command, you can change the level of messages sent and stored in a history table on the router. You can also change the number of messages that get stored in the history table.

Messages are stored in the history table, because SNMP traps are not guaranteed to reach their destination. By default, one message of the level warning and above (see Table 2: Syslog Message Severity Levels, on page 6) is stored in the history table even if syslog traps are not enabled.

Note Table 2: Syslog Message Severity Levels, on page 6 lists the level keywords and severity level. For SNMP usage, the severity level values use +1. For example, **emergency** equals 1 not 0 and **critical** equals 3 not 2.

Configuration Example

This example shows how to change the severity level and table size defaults of the logging history table.

```
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# logging history severity
RP/0/RP0/CPU0:Router(config)# logging history size number
RP/0/RP0/CPU0:Router(config)# commit
```

Configuring System Logging

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

Configuring Syslog Severity Level for Telemetry

The severity of syslog messages that are generated by the router varies from emergencies to simple notifications. You can specify a severity keyword corresponding to any one of the severity levels—from the highest severity level 0 (emergencies) through the lowest severity level 7 (debugging). Depending upon the severity level you have specified, the router streams data to the telemetry server, starting from the chosen severity level and higher. See Implementing System Logging for more information.

You can specify the severity level by using the **logging yang** *severity-level* command.

Tip You can programmatically monitor syslog messages by using the `openconfig-messages.yang` OpenConfig data model. To get started with using data models, see the *Programmability Configuration Guide for Cisco 8000 Series Routers*.

Configuration Example

This example sets **warnings** as the severity level. This results in the streaming of syslogs only for **warnings**, **errors**, **critical**, **alert**, and **emergencies**. Syslogs of lower severity are not streamed.

```
Router(config)#logging yang warnings
```

Telemetry Output

This example shows sample telemetry operational output when **logging yang warnings** command is configured.

```
--------
{"node_id_str":"ios","subscription_id_str":"app_TEST_200000001","encoding_path":
"openconfig-system:system","collection_id":"40","collection_start_time":"1664513125273",
"msg_timestamp":"1664513125273","data_json":[{"timestamp":"1664513125272","content":
{"messages":
{"state":{"severity":"EMERGENCY","message":
{"msg":"RP/0/0/CPU0:Sep 30 10:15:25.272 IST: logger[67820]: %OS-SYSLOG-0-LOG_EMERG :
TEST_EMERG ","priority":184,"app-name":"logger","procid":"67820","msgid":
"OS-SYSLOG-0-LOG_EMERG"}}}}}],"collection_end_time":"1664513125273"}
--------
 Sub_id 200000001, flag 4, len 534
--------
{"node_id_str":"ios","subscription_id_str":"app_TEST_200000001","encoding_path":
"openconfig-system:system","collection_id":"41","collection_start_time":"1664513137884",
"msg_timestamp":"1664513137884","data_json":[{"timestamp":"1664513137883","content":
{"messages":
{"state":{"severity":"WARNING","message":
{"msg":"RP/0/0/CPU0:Sep 30 10:15:37.882 IST: logger[67997]:
%OS-SYSLOG-4-LOG_WARNING : TEST_WARN
 ","priority":188,"app-name":"logger","procid":"67997","msgid":
"OS-SYSLOG-4-LOG_WARNING"}}}}}],"collection_end_time":"1664513137884"}
--------
 Sub_id 200000001, flag 4, len 529
--------
{"node_id_str":"ios","subscription_id_str":"app_TEST_200000001","encoding_path":
"openconfig-system:system","collection_id":"42","collection_start_time":
"1664513562626","msg_timestamp":"1664513562626","data_json":[{"timestamp":"1664513562624",
"content":{"messages":{"state":{"severity":"CRITICAL","message":
{"msg":"RP/0/0/CPU0:Sep 30 10:22:42.624 IST: logger[68957]: %OS-SYSLOG-2-LOG_CRIT :
 TEST_CRIT ","priority":186,"app-name":"logger","procid":"68957","msgid":
"OS-SYSLOG-2-LOG_CRIT"}}}}}],"collection_end_time":"1664513562626"}
--------
 Sub_id 200000001, flag 4, len 529
--------
{"node_id_str":"ios","subscription_id_str":"app_TEST_200000001","encoding_path":
"openconfig-system:system","collection_id":"43","collection_start_time":
"1664513570004","msg_timestamp":"1664513570004","data_json":[{"timestamp":"1664513570003",
"content":{"messages":{"state":{"severity":"ALERT","message":
{"msg":"RP/0/0/CPU0:Sep 30 10:22:50.002 IST: logger[69113]: %OS-SYSLOG-1-LOG_ALERT :
TEST_ALERT ","priority":185,"app-name":"logger","procid":"69113",
"msgid":"OS-SYSLOG-1-LOG_ALERT"}}}}}],"collection_end_time":"1664513570004"}
--------
 Sub_id 200000001, flag 4, len 525
--------
{"node_id_str":"ios","subscription_id_str":"app_TEST_200000001","encoding_path":
"openconfig-system:system","collection_id":"44","collection_start_time":
"1664513844428","msg_timestamp":"1664513844428","data_json":[{"timestamp":"1664513844427","content":
{"messages":{"state":{"severity":"ERROR","message":
{"msg":"RP/0/0/CPU0:Sep 30 10:27:24.426 IST: logger[69203]: %OS-SYSLOG-3-LOG_ERR :
TEST_ERROR ","priority":187,"app-name":"logger","procid":"69203","msgid":
"OS-SYSLOG-3-LOG_ERR"}}}}}],"collection_end_time":"1664513844428"}
--------
```

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
```

System Logging Message Count

Table 4: Feature History Table

Feature Name	Release Information	Feature Description
System Logging Message Count	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I
System Logging Message Count	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

Feature Name	Release Information	Feature Description
System Logging Message Count	Release 7.11.1	Instead of calculating the bytes consumed by Syslog as you did previously, you can now easily and effectively manage the buffer size of the system log messages by specifying the number of entries the system log displays. The feature introduces these changes: **CLI:** • The **entries-count** keyword is added to the **logging buffered** command. **YANG Data Model:** • New Xpaths for Cisco-IOS-XR-infra-syslog-cfg • New Xpaths for Cisco-IOS-XR-um-logging-cfg

Earlier, you were only able to configure the buffer size of the system log messages in bytes using the **logging buffered** command.

Starting Cisco IOS XR Software Release 7.5.5, you can you can specify the number of entries to be present while displaying the system logs. Based on the number of entries, the system internally calculates the buffer size and reserves the same for system log buffer. The default value is 2545. The range for sytem logging message count entry is from 2545 to 151699. When you disable the command, the logging buffer size points back to the default value of 2545.

If both the **logging buffered** *bytes* and **logging buffered entries-count** commands are present, then the maximum configured value is taken to display the number of system log messages.

Configuration Example for System Logging Message Count

Use the **logging buffered entries-count** command to specify the number of entries to be present while displaying the system logs.

```
Router# configure
Router(config)# logging buffered entries-count 3000
Router(config)# commit
```

Running Configuration

```
Router#show running-config logging
.
.
.
logging console disable
logging buffered entries-count 3000
!
```

Verification

```
Router(config)#show logging last 3
Syslog logging: enabled (0 messages dropped, 0 flushes, 0 overruns)
    Console logging: level warnings, 2 messages logged
    Monitor logging: level debugging, 0 messages logged
    Trap logging: level informational, 0 messages logged
    Buffer logging: level debugging, 63 messages logged

Log Buffer (3000 entries):
```

Configuring Logging to a Remote Server

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

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

Table 5: Features for Managing Syslog Messages

Features	Description
UNIX system log facility	Facility is the identifier used by UNIX to describe the application or process that submitted the log message. You can configure the syslog facility in which syslog messages are sent by using the **logging facility** command.
Hostname prefix logging	Cisco IOS XR Software supports hostname prefix logging. When enabled, hostname prefix logging appends a hostname prefix to syslog messages being sent from the router to syslog servers. You can use hostname prefixes to sort the messages being sent to a given syslog server from different networking devices. Use the **logging hostname** command to append a hostname prefix to syslog messages sent to syslog servers
Syslog source address logging	By default, a syslog message sent to a syslog server contains the IP address of the interface it uses to leave the router. Use the **logging source-interface** command to set all syslog messages to contain the same IP address, regardless of which interface the syslog message uses to exit the router.

Configuration Example for Logging to Syslog Server

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

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

Configuration Example for Logging to SNMP Server

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

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

For more information on SNMP server configurations, see the *Configuring Simple Network Management Protocol* chapter in the *System Management Configuration Guide for Cisco 8000 Series Routers*

Related Topics

- Configuring Logging to the Logging Buffer, on page 10

System Log Facility and Source-address per Remote Server

Table 6: Feature History Table

Feature Name	Release Information	Feature Description
System Log Facility and Source-address per Remote Server	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on: • 8011-12G12X4Y-A • 8011-12G12X4Y-D
System Log Facility and Source-address per Remote Server	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on Cisco 8011-4G24Y4H-I routers.

Feature Name	Release Information	Feature Description
System Log Facility and Source-address per Remote Server	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100, K100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 8712-MOD-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
System Log Facility and Source-address per Remote Server	Release 24.2.11	You can now assign a facility type per remote syslog server, which the router uses to calculate the priority value of the syslog messages sent. You can also configure the source address to choose the interface to send remote syslog packets per remote server. The feature introduces these changes: Modified Command: **CLI** • The keywords **facility** and **source-address** per remote syslog server are introduced in the **logging** command. **YANG Data Models:** • New XPaths for `openconfig-system-logging.yang` (see GitHub, YANG Data Models Navigator)

Configuration Example for Logging to Remote Syslog Server

The configurations for **facility** and **source-address** per remote syslog server takes priority over global configuration.

This example shows how to configure **facility** and **source-address** per remote syslog server:

```
Router#configure
Router(config)#
Router(config)#logging 209.165.201.1 source-address 209.165.201.2
Router(config)#logging 209.165.201.1 facility local2
Router(config)#commit
```

Configuring Logging to Terminal Lines

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

Configuration Example

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

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

Enable Message Logs for Third-Party Software Containers

Table 7: Feature History Table

Feature Name	Release Information	Feature Description
Message Logs for Third-Party Software Containers	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I

Feature Name	Release Information	Feature Description
Message Logs for Third-Party Software Containers	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
Message Logs for Third-Party Software Containers	Release 7.3.15	This feature introduces the logging container all command to monitor messages from a third-party container logs, such as Docker.

Cisco IOS XR operating system can host third-party software containers, such as Docker. To monitor logs from such software containers, use the **logging container all** command.

Configuration Example

This example shows how to enable third-party software container logging and how to view the logs for the third-party software container named Docker:

```
Router# configure
Router(config)# logging container all
Router(config)# commit


Router# show running-config logging

logging container all

Router# show logging | inc DOCKER
Syslog logging: enabled (0 messages dropped, 0 flushes, 0 overruns)
    Console logging: level warnings, 5 messages logged
    Monitor logging: level debugging, 0 messages logged
    Trap logging: level informational, 0 messages logged
    Buffer logging: level debugging, 148 messages logged

Log Buffer (2097152 bytes):

RP/0/RP0/CPU0:Mar  5 06:56:11.913 UTC: exec[66927]: %SECURITY-LOGIN-6-AUTHEN_SUCCESS :
Successfully authenticated user 'lab' from 'console' on 'con0_RP0_CPU0'
RP/0/RP0/CPU0:Mar  5 06:58:13.053 UTC: config[66985]: %MGBL-SYS-5-CONFIG_I : Configured
from console by lab
RP/0/RP0/CPU0:Mar  5 06:59:04.775 UTC: ubuntu-1[67232]: %OS-SYSLOG-6-DOCKER_APP :
^[]0;root@c382b2e7bed6: /^Groot@c382b2e7bed6:/# testlog
RP/0/RP0/CPU0:Mar  5 06:59:04.830 UTC: config[67139]: %MGBL-CONFIG-6-DB_COMMIT : Configuration
```

```
 committed by user 'lab'. Use 'show configuration commit changes 1000000012' to view the
changes.
RP/0/RP0/CPU0:Mar  5 06:59:45.028 UTC: config[67139]: %MGBL-SYS-5-CONFIG_I : Configured
from console by lab
RP/0/RP0/CPU0:Mar  5 06:59:48.552 UTC: run_cmd[67780]: %INFRA-INFRA_MSG-5-RUN_LOGIN : User
 lab logged into shell from con0/RP0/CPU0
RP/0/RP0/CPU0:Mar  5 06:59:56.073 UTC: ubuntu-1[67976]: %OS-SYSLOG-6-DOCKER_APP : testlog-123

RP/0/RP0/CPU0:Mar  5 07:00:12.471 UTC: ubuntu-1[68099]: %OS-SYSLOG-6-DOCKER_APP : testlog-new1

RP/0/RP0/CPU0:Mar  5 07:01:55.747 UTC: ubuntu-1[68245]: %OS-SYSLOG-6-DOCKER_APP : testlog-new1

RP/0/RP0/CPU0:Mar  5 07:02:02.869 UTC: run_cmd[67780]: %INFRA-INFRA_MSG-5-RUN_LOGOUT : User
 lab logged out of shell from con0/RP0/CPU0
```

Third-party applications logging on remote server

Table 8: Feature History Table

Feature Name	Release Information	Feature Description
Third-party applications logging on remote server	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I
Third-party applications logging on remote server	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) This feature enables third-party applications to forward syslog messages to a remote server for handling a high rate of system logging. *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

Third party application logging on remote server is an IOS XR logging feature that efficiently manages large volumes of log messages and high rates of logging by forwarding logs generated by third-party applications to a remote server.

- Before Release 24.4.1, the router couldn't efficiently handle large volumes of log messages and high rates of logging because it stored remote syslog logging configuration information in the router's shared memory.
- From Release 24.4.1 onwards, the router processes only native container logs, which can be viewed using the show-logging buffer command. The router no longer processes third party application logs and sends them directly to the remote syslog server.

The XR syslog module processes only native container logs, which can be viewed using the **show-logging buffer** command. Third-party application logs are sent directly to the remote syslog server and are no longer processed by XR. For example, while you can view NOSi application logs using the **show logging** command, XR will no longer process third-party application logs going forward.

Limitations and restrictions for third-party applications logging on remote server

These are the limitations and restrictions for third-party applications logging on remote server:

- You cannot simultaneously enable both "System Log Facility" and "logging container all" configurations.
- You cannot simultaneously enable both "TLS-based remote syslog" and "logging container all" configurations.

Configure third-party applications logging on remote server

You must perform these steps to configure third-party applications logging on remote server:

Procedure

Step 1 Configure the logging to a remote server feature with the **logging source-interface** command to send syslog messages to an external syslog server. For configuration information, see Configuration Example for Logging to Syslog Server, on page 13

The kernel level refers to operations or functionalities that are executed within the kernel of an operating system. When you enable kernel-level logging in a syslog configuration, you are primarily instructing the system to log messages that are related to kernel operations.

Step 2 Enable third-party software container logging with the **logging container all** command to capture the logs from all third-party container logs. For configuration information, see Configuration Example, on page 17.

Note
If you don’t configure **logging container all** command, the Cisco IOS XR software doesn’t forward the logs to the remote syslog server.

Modifying Logging to Console Terminal

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

Configuration Example

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

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

Modifying Time Stamp Format

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

Configuration Example

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

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

Suppressing Duplicate Syslog Messages

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

Configuration Example

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

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

Displaying System Logging Messages

You can display the syslog messages stored in the logging buffer by using the **show logging** command.

Configuration Example

This example shows how to display the syslog messages stored in the logging buffer.

```
RP/0/RP0/CPU0:Router# show logging
RP/0/RP0/CPU0:Router# show logging location 0/1/CPU0
RP/0/RP0/CPU0:Router# show logging process init
RP/0/RP0/CPU0:Router# show logging string install
RP/0/RP0/CPU0:Router# show logging start december 1 10:30:00
RP/0/RP0/CPU0:Router# show logging end december 2 22:16:00
```

Note The commands can be entered in any order.

```
RP/0/RP0/CPU0:Router# show logging
Syslog logging: enabled (0 messages dropped, 0 flushes, 0 overruns)
```

```
    Console logging: level warnings, 0 messages logged
    Monitor logging: level debugging, 0 messages logged
    Trap logging: level informational, 0 messages logged
    Buffer logging: level debugging, 82 messages logged

Log Buffer (307200 bytes):

RP/0/0/CPU0:Feb  7 15:36:16.655 IST: init[68452]: %OS-INIT-7-MBI_STARTED : total time 0.215
 seconds
RP/0/0/CPU0:Feb  7 15:36:16.759 IST: sysmgr[55]: %OS-SYSMGR-5-NOTICE : Card is COLD started
RP/0/0/CPU0:Feb  7 15:36:16.893 IST: init[68452]: %OS-INIT-7-INSTALL_READY : total time
0.453 seconds
RP/0/0/CPU0:Feb  7 15:36:17.125 IST: sysmgr[278]: %OS-SYSMGR-6-INFO : Backup system manager
 is ready
RP/0/0/CPU0:Feb  7 15:36:17.149 IST: syslogd[405]: %SECURITY-XR_SSL-6-INFO : XR SSL info:
Setting fips register
RP/0/0/CPU0:Feb  7 15:36:17.177 IST: spp[52]: %L2-VTIO-6-NO_PORTS : Plug-in found no ports
 to manage
RP/0/0/CPU0:Feb  7 15:36:18.651 IST: dsc[210]: %PLATFORM-DSC-6-INFO_I_AM_DSC : Setting
myself as DSC
RP/0/0/CPU0:Feb  7 15:36:18.653 IST: sysmgr[55]: %OS-SYSMGR-7-DEBUG : node set to DSC
```

```
RP/0/RP0/CPU0:Router# show logging process smartlicserver
Syslog logging: enabled (0 messages dropped, 0 flushes, 0 overruns)
    Console logging: level warnings, 0 messages logged
    Monitor logging: level debugging, 0 messages logged
    Trap logging: level informational, 0 messages logged
    Buffer logging: level debugging, 82 messages logged

Log Buffer (307200 bytes):

RP/0/0/CPU0:Feb  7 15:37:12.434 IST: smartlicserver[119]: %SMART_LIC-6-AGENT_ENABLED:Smart
 Agent for Licensing is enabled
RP/0/0/CPU0:Feb  7 15:37:23.029 IST: smartlicserver[119]: %SMART_LIC-6-REPORTING_REQUIRED:A
 Usage report acknowledgement will be required in 353 days.
RP/0/0/CPU0:Feb  7 15:37:24.030 IST: smartlicserver[119]: %SMART_LIC-5-IN_COMPLIANCE:All
entitlements and licenses in use on this device are authorized
RP/0/0/CPU0:Feb  7 15:37:29.474 IST: smartlicserver[119]: %SMART_LIC-5-SLR_IN_COMPLIANCE:The
 entitlement regid.2019-03.com.cisco.ENXR-TRK,1.0_2b015ca9-b01d-40eb-80b6-e6647f8fcf76 in
use on this device is authorized
RP/0/0/CPU0:Feb  7 15:37:38.045 IST: smartlicserver[119]:
%SMART_LIC-3-COMM_FAILED:Communications failure with the Cisco Smart License Utility (CSLU)
 : Unable to resolve server hostname/domain name
RP/0/0/CPU0:Feb  7 15:37:39.047 IST: smartlicserver[119]: %SMART_LIC-5-IN_COMPLIANCE:All
entitlements and licenses in use on this device are authorized
RP/0/0/CPU0:ios#
```

```
RP/0/RP0/CPU0:Router# show logging start february 7 15:37:24
Syslog logging: enabled (0 messages dropped, 0 flushes, 0 overruns)
    Console logging: level warnings, 0 messages logged
    Monitor logging: level debugging, 0 messages logged
    Trap logging: level informational, 0 messages logged
    Buffer logging: level debugging, 82 messages logged

Log Buffer (307200 bytes):

RP/0/0/CPU0:Feb  7 15:37:24.030 IST: smartlicserver[119]: %SMART_LIC-5-IN_COMPLIANCE:All
entitlements and licenses in use on this device are authorized
RP/0/0/CPU0:Feb  7 15:37:24.112 IST: iedged[462]: %SUBSCRIBER-SUB_UTIL-5-SESSION_THROTTLE
: Subscriber Infra is ready. Reason: [V6 Subscriber infra process(es) is available].
RP/0/0/CPU0:Feb  7 15:37:24.112 IST: subdb_svr[207]: %SUBSCRIBER-SUB_UTIL-5-SESSION_THROTTLE
 : Subscriber Infra is ready. Reason: [V6 Subscriber infra process(es) is available].
RP/0/0/CPU0:Feb  7 15:37:24.112 IST: pppoe_ma[216]: %SUBSCRIBER-SUB_UTIL-5-SESSION_THROTTLE
 : Subscriber Infra is ready. Reason: [V6 Subscriber infra process(es) is available].
RP/0/0/CPU0:Feb  7 15:37:29.474 IST: smartlicserver[119]: %SMART_LIC-5-SLR_IN_COMPLIANCE:The
```

```
 entitlement regid.2019-03.com.cisco.ENXR-TRK,1.0_2b015ca9-b01d-40eb-80b6-e6647f8fcf76 in
use on this device is authorized
RP/0/0/CPU0:Feb  7 15:37:38.045 IST: smartlicserver[119]:
%SMART_LIC-3-COMM_FAILED:Communications failure with the Cisco Smart License Utility (CSLU)
 : Unable to resolve server hostname/domain name
RP/0/0/CPU0:Feb  7 15:37:39.047 IST: smartlicserver[119]: %SMART_LIC-5-IN_COMPLIANCE:All
entitlements and licenses in use on this device are authorized
RP/0/0/CPU0:Feb  7 15:46:19.976 IST: /pkg/sbin/sysmgr_log[68415]: %OS-SYSMGR-7-CHECK_LOG :
 /pkg/bin/sysmgr_debug_script invoked for : (ltrace_data_export) sysmgr_level_ready_timeout:
 EOI required, but never received from ltrace_data_export,jid=293   Output is in
/disk0://sysmgr_debug/debug.node0_0_CPU0.174906
RP/0/0/CPU0:ios#
```

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 9: Commands Used to Set Syslog Archive Attributes

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

Features	Description
severity severity	Specifies the minimum severity of log messages to archive. All syslog messages greater than or equal to this configured level are archived while those lesser than this are filtered out.
threshold	Specifics the threshold percentage for archive logs.

Configuration Example

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

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

Platform Automated Monitoring

Platform Automated Monitoring (PAM) is a system monitoring tool integrated into Cisco IOS XR software image to monitor the following issues:

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

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

Table 10: Feature History Table

Feature Name	Release	Description
Platform Automated Monitoring for Blocked Processes	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I

Feature Name	Release	Description
Platform Automated Monitoring for Blocked Processes	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
Platform Automated Monitoring for Blocked Processes	Release 7.5.2	You can enable the Platform Automated Monitoring tool integrated into the Cisco IOS XR software image and receive alerts if any process is blocked. Several system failures can cause a blocked process, such as memory leak, network connection loss, and so on. The tool collects the required data to troubleshoot the issue and generates a system log message with the name of the process that is currently blocked. This feature introduces the following commands: • **enable-pam process-monitoring** • **disable-pam process-monitoring** • **show pam process-monitoring-status**

PAM Events

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

The format of the .tgz file is: *PAM-<platform>-<PAM event>-<node-name>-<PAM process>-<YYYYMMDD>-<checksum>.tgz*.For example, *PAM-cisco8000-crash-xr_0_RP0_CPU0-ipv4_rib-2016Aug16-210405.tgz* is the file collected when PAM detects a process crash.

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

For the list of commands used while collecting logs, refer Files Collected by PAM Tool, on page 28.

The Platform Automated Monitoring for blocked processes detects and alerts if any of the processes are blocked, except for the processes which are expected to be blocked by their design. These processes are listed in the table below:

Blocked process	**Blocked on**
lpts_fm	lpts_pa
isis	lspv_server
Ospf	lspv_server
l2vpn_mgr	lspv_server
mpls_ldp	lspv_server
bgp	lspv_server
te_control	lspv_server
xtc_agent	lspv_server

The sections below describe the main PAM events:

Crash Monitoring

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

```
RP/0/RP0/CPU0:Aug 16 21:04:06.442 : logger[69324]: %OS-SYSLOG-4-LOG_WARNING : PAM detected
 crash for ipv4_rib on 0_RP0_CPU0.
All necessary files for debug have been collected and saved at
0/RP0/CPU0 :
harddisk:/cisco_support/PAM-cisco8000-crash-xr_0_RP0_CPU0-ipv4_rib-2016Aug16-210405.tgz
Please copy tgz file out of the router and send to Cisco support. This tgz file will be
removed after 14 days.)
```

Traceback Monitoring

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

```
RP/0/RP0/CPU0:Aug 16 21:42:42.320 : logger[66139]: %OS-SYSLOG-4-LOG_WARNING : PAM detected
```

```
 traceback for ipv4_rib on 0_RP0_CPU0.
All necessary files for debug have been collected and saved at
0/RP0/CPU0 :
harddisk:/cisco_support/PAM-cisco8000-traceback-xr_0_RP0_CPU0-ipv4_rib-2016Aug16-214242.tgz

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

Memory Usage Monitoring

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

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

```
RP/0/RP0/CPU0:Aug 17 05:13:32.684 : logger[67772]: %OS-SYSLOG-4-LOG_WARNING : PAM detected
 significant memory increase
(from 13.00MB at 2016/Aug/16/20:42:41 to 28.00MB at 2016/Aug/17/04:12:55) for
pam_memory_leaker on 0_RP0_CPU0.
All necessary files for debug have been collected and saved at
0/RP0/CPU0 :
harddisk:/cisco_support/PAM-cisco8000-memory_leak-xr_0_RP0_CPU0-pam_memory_leaker-2016Aug17-051332.tgz

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

CPU Monitoring

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

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

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

```
RP/0/RP0/CPU0:Aug 16 00:56:00.819 : logger[68245]: %OS-SYSLOG-4-LOG_WARNING : PAM detected
 CPU hog for cpu_hogger on 0_RP0_CPU0.
All necessary files for debug have been collected and saved at 0/RP0/CPU0 :
harddisk:/cisco_support/PAM-cisco8000-cpu_hog-xr_0_RP0_CPU0-cpu_hogger-2016Aug16-005600.tgz

(Please copy tgz file out of the router and send to Cisco support. This tgz file will be
removed after 14 days.)
RP/0/RP0/CPU0:Jun 21 15:33:54.517 : logger[69042]: %OS-SYSLOG-1-LOG_ALERT : PAM detected
ifmgr is hogging CPU on 0_RP0_CPU0!
```

File System Monitoring

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

```
RP/0/RP0/CPU0:Jun 20 13:59:04.986 : logger[66125]: %OS-SYSLOG-4-LOG_WARNING : PAM detected
 /misc/config is full on 0_1_CPU0
(please clean up to avoid any fault caused by this). All necessary files for debug have
```

```
been collected and saved at
0/RP0/CPU0 : harddisk:/cisco_support/PAM-cisco8000-disk_usage-xr_0_1_CPU0-2016Jun20-135904.tgz

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

Disable and Re-enable PAM

The PAM tool consists of the following monitoring processes:

- monitor_cpu.pl
- monitor_crash.pl
- monitor_show_logging.pl
- monitor_process.pl

Note Monitor process.pl in PAM monitors all nodes and generates a system log message with the process name that is blocked if it detects any process is blocked for more than 30 minutes. It prevents multiple alarms for the same blocked process.

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

- From Cisco IOS XR Command Line Interface:

```
Router# show pam status
Tue Jun 14 17:58:42.791 UTC
PAM is enabled
```

- From router shell prompt:

```
Router# run ps auxw|egrep perl

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

Disable PAM

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

```
Router# disable-pam
```

Re-enable PAM

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

```
Router# enable-pam
```

Data Archiving in PAM

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

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

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

Files Collected by PAM Tool

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

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

Note Starting from Cisco IOS XR Release 25.1.1, the core file format changes from..`gz` to `.lz4` and the default core file location is changed from `/misc/disk1` to `/misc/disk1/coredumps .`

Event Name	Commands and Files Collected by PAM
Process crash	• **show install active** • **show platform** • **show version** • core (gz) file • core.txt file
Process traceback	• **show dll** • **show install active** • **show logging** • **show platform** • **show version**
Memory leak	• **show install active** • **show platform** • **show version** • core (gz) file • dumpcore running • continuous memory usage snapshots

Event Name	Commands and Files Collected by PAM
Show logging event	• **show install active** • **show logging** • **show platform** • **show version** • core (gz) file • core.txt file
CPU hog	• **follow process** • **pstack** • **show dll** • **show install active** • **show platform** • **show version** • **top -H** • core (gz) file • CPU usage snapshots
Disk usage	• **show install active** • **show platform** • **show version** • console log • core (gz) file • Disk usage snapshots
Process Blockage	• **show version** • **show install active** • **show platform** • **show logging** • **show running-config** • **show process blocked location all** • core (gz) file

Insecure features warning syslog messages

Insecure features warning syslog messages are system notifications that display warning syslog message when you run an insecure command.

Table 11: Feature History Table

Feature Name	Release Information	Feature Description
Insecure features warning syslog messages	Release 25.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100, K100]), 8010 [ASIC: A100]); Centralized Systems (8600 [ASIC:Q200]); Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) Cisco IOS XR software warns you with a syslog message each time you use an insecure command and repeats the warning every 30 days until you remove the command. This helps you identify potential security risks and suggests safer alternatives to improve your network security. Cisco will systematically deprecate and eventually remove these insecure features and protocols in future IOS XR releases. For more information on insecure commands and their alternatives, see Feature deprecation phasing out insecure capabilities.

When you configure or execute an insecure command, the Cisco IOS XR software displays the following types of messages:

- **Periodic warning**: When you configure an insecure command, the Cisco IOS XR software displays a syslog message when you first execute an insecure command, and it repeats the warning syslog message every 30 days until you remove the insecure command.
- **One-Time warning**: When you execute an insecure command, the Cisco IOS XR software displays a syslog message after you execute an insecure command.
- **Informational message**: When you unconfigure an insecure command, the Cisco IOS XR software displays a syslog message after you remove an insecure command.

You cannot disable warning syslog messages. However, you can filter or change the syslog severity level to control which messages you can see.

You can see the messages on both active and standby route processors in a dual-RP system.

From Release 25.4.1 the insecure commands are deprecated and will be removed in the future release.

These documents list all features planned for removal, including insecure commands, and provide recommended secure alternatives to help you maintain network security and compliance.

- Feature deprecation phasing out insecure capabilities
- Feature deprecation and removal details
- Feature removal and suggested alternatives

Warning syslog message format

The Cisco IOS XR software display a warning syslog message in the following format:

- **For configuring an insecure configuration or performing an insecure action**:

```
%INFRA-WARN_INSECURE-4-INSECURE_FEATURE_WARN : Feature <insecure_feature_name> utilized
 or configured. This feature is deprecated as it is known to be insecure; it will be
removed in a future release. <additional_details>
```

- **For unconfigure an insecure feature**:

```
%INFRA-WARN_INSECURE-6-INSECURE_CONFIG_REMOVED : Insecure feature <insecure feature
name> configuration removed.
```

Table 12: Insecure feature or protocol

Insecure feature or protocol	Description
Insecure feature name	Displays the insecure feature short name. For more information on insecure commands and their alternatives, see Feature deprecation phasing out insecure capabilities.
Additional details	Displays additional details such as a secure alternative to use instead of the insecure command or configuration. For more information on insecure commands and their alternatives, see Feature deprecation phasing out insecure capabilities.

Some features may not have an alternate suggestion. In these cases, the one-time warning syslog message will not include additional details.

CHAPTER 4

Monitoring Alarms and Implementing Alarm Log Correlation

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

Monitoring Alarms and Implementing Alarm Log Correlation

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

Prerequisites for Implementing Alarm Log Correlation

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

Information About Monitoring Alarms and Implementing Alarm Log Correlation

Displaying Router Alarms

You can view the router alarms in brief and detail.

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

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

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

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

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

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

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

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

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

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


Location:                1/0/CPU0

AID:                     XR_FABRIC/SW_MISC_ERR/18

Tag String:              FAM_FAULT_TAG_HW_FIA_LC_BANDWIDTH

Module Name:             N/A

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

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

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

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

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

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

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

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

Agent Handle:            93827323237168

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

Alarm Logging and Debugging Event Management System

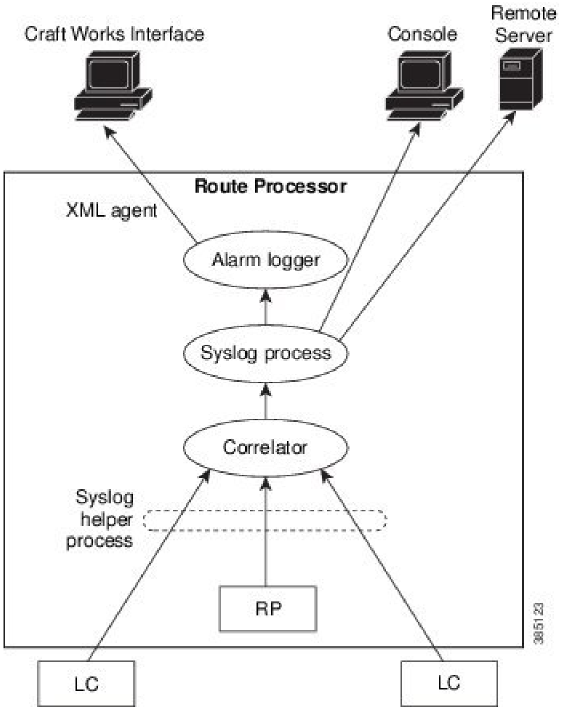
Cisco IOS XR Software Alarm Logging and Debugging Event Management System (ALDEMS) is used to monitor and store alarm messages that are forwarded by system servers and applications. In addition, ALDEMS correlates alarm messages forwarded due to a single root cause.

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

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

Figure 1: ALDEMS Component Communications, on page 36 illustrates the relationship between the components that constitute ALDEMS.

Figure 1: ALDEMS Component Communications



Correlator

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

System Logging Process

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

Alarm Logger

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

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

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

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

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

Configuring Alarm Log Correlation

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

Configuring Logging Correlation Rules

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

Timeout can be configured to specify the time interval for a message search once a match is found. Timeout begins when the correlator captures any alarm message specified for a correlation rule.

Configuration Example

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

```
Router# configure
Router(config)# logging correlator rule rule1 type stateful
Router(config-corr-rule-st)# timeout 60000
Router(config-corr-rule-st)# exit
Router(config)# commit
```

Correlating a Root Cause and Non Root Cause Alarms

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

Configuration Example

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

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

Applying a Logging Correlation Rule

If a correlation rule is applied to a specific set of contexts or locations, then correlation takes place only for those messages that match the configured cause values for the rule and that match at least one of those contexts or locations. When a correlation rule is configured and applied, the correlator starts searching for a message match as specified in the rule.

Configuration Example

This example shows how to apply a logging correlation rule.

```
Router# configure
Router(config)# logging correlator apply rule rule1
Router(config-corr-apply-rule)# all-of-router
or
Router(config-corr-apply-rule)# location 0/1/CPU0
or
Router(config-corr-apply-rule)# context HundredGigE_0_0_0_0
Router(config)# commit
```

Configuring a Logging Correlation Rule Set

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

Configuration Example

This example shows how to configure and apply a logging correlation rule set for multiple correlation rules. To configure a ruleset, you should first configure a rule with the same name. The logging correlation ruleset can be applied to the entire router or to a specific context or location.

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

Configuring Hierarchical Correlation Rule Flags

Hierarchical correlation is when a single alarm is both a root cause for one correlation rule and a non-root cause for another rule, and when alarms are generated resulting in a successful correlation associated with both rules. What happens to a non-root-cause alarm depends on the behavior of its correlated root-cause alarm. There are cases in which you want to control the stateful behavior associated with these hierarchies and to implement flags, such as reparenting and reissuing of non-bistate alarms. For detailed information about hierarchical correlation and correlation flags, see Hierarchical Correlation, on page 43

Configuration Example

This example shows how to configure hierarchical correlation rule flags.

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

Configuring Logging Suppression Rules

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

Configuration Example

This example shows how to configure logging suppression rules.

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

Modifying Logging Events Buffer Settings

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

Configuration Example

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

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

Modifying Logging Correlation Buffer Settings

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

Configuration Example

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

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

Enabling Alarm Source Location Display Field for Bistate Alarms

Bistate alarms are generated by state changes associated with system hardware. The bistate alarm message format is similar to syslog messages. You can optionally configure the output to include the location of the actual alarm source, which may be different from the process that logged the alarm. For more information about bistate alarms see, Bistate Alarms, on page 42

Configuration Example

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

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

Configuring SNMP Correlation Rules

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

Configuration Example

This example shows how to configure and apply correlation rules for SNMP traps. The SNMP correlator buffer size is also configured as 1024 bytes. The default value for buffer size is 64KB.

```
Router# configure terminal
Router(config)# snmp-server correlator buffer-size 1024 (optional)
Router(config)# snmp-server correlator rule rule1
Router(config-corr-rule-nonst)# timeout 100
Router(config-corr-rule-nonst)# rootcause 1.3.6.1.2.1.47.1.1
Router(config-corr-rule-nonst-rootvb)# varbind 1.3.6.1.2.1.47.1.2 index regex .*
Router(config-corr-rule-nonst-rootvb)# varbind 1.3.6.1.2.1.47.1.2 value regex .*
Router(config-corr-rule-nonst-rootvb)# exit
Router(config-corr-rule-nonst)# nonrootcause
Router(config-corr-rule-nonst-nonrc)# trap 1.3.6.1.2.1.47.1.1
Router(config-corr-rule-nonst-nonrcvb)# varbind 1.3.6.1.2.1.47.1.3 index regex .*
Router(config-corr-rule-nonst-nonrcvb)# varbind 1.3.6.1.2.1.47.1.3 value regex .*
Router(config-corr-rule-nonst-nonrcvb)# exit
Router(config-corr-rule-nonst-nonrc)# trap 1.3.6.1.2.1.47.1.2
Router(config-corr-rule-nonst-nonrcvb)# varbind 1.3.6.1.2.1.47.1.4 index regex .*
Router(config-corr-rule-nonst-nonrcvb)# varbind 1.3.6.1.2.1.47.1.4 value regex .*
Router(config-corr-rule-nonst-nonrcvb)# exit
Router(config-corr-rule-nonst-nonrc)# exit
Router(config-corr-rule-nonst)# exit
Router(config)# snmp-server correlator apply rule test host ipv4 address 1.2.3.4
Router(config)# commit
```

Configuring SNMP Correlation Ruleset

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

Configuration Example

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

```
Router# configure terminal
Router(config)# snmp-server correlator ruleset rule1 rulename rule1
Router(config)# snmp-server correlator apply ruleset rule1 host ipv4 address 1.2.3.4
Router(config)# commit
```

Alarm Logging Correlation-Details

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

Correlation Rules

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

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

- Message category
- Message group
- Message code

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

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

Alarm Severity Level and Filtering

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

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

Table 13: Alarm Severity Levels for Event Logging

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

Bistate Alarms

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

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

Bistate alarm notifications provide the following information:

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

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

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

The following is a sample bistate alarm message:

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

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

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

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

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

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

Hierarchical Correlation

Hierarchical correlation takes effect when the following conditions are true:

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

The following example illustrates two hierarchical correlation rules:

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

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

Cause 1 -> Cause 2 -> Cause 3

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

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

Context Correlation Flag

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

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

The following is a scenario of context correlation behavior:

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

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

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

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

Duration Timeout Flags

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

Reparent Flag

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

The following example illustrates context correlation behavior:

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

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

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

Active-standby consistency check

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

Alarm Clearance

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

CHAPTER 5

Onboard Failure Logging

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

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

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

Feature History for Implementing OBFL

Release	Modification
Release 7.0.11	This feature was introduced.

Prerequisites

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

Information About OBFL

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

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

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

Baseline Data Collection

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

Table 14: Data Types for Baseline Data Collection

Data Type	Details
Installation	Chassis serial number and slot number are stored at initial boot.
Run-time	Total run-time is limited to the size of the history buffer used for logging. This is based on the local router clock with logging granularity of 15 minutes.
Temperature	Information from the temperature sensors is recorded after boot. The subsequent recordings are specific to variations based on preset thresholds.
Voltage	Information from the voltage sensors is recorded after boot. The subsequent recordings are specific to variations based on preset thresholds.

Event-Driven Data Collection

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

Table 15: Data Types for Event-Driven Data Collection

Data Type	Details
Alarm	Major and critical alarm state changes.
FPD	FPD upgrade information.

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

Supported Cards and Platform

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

Table 16: OBFL Support on Cisco 8000 Series Router

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

Monitoring and Maintaining OBFL

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

Procedure

	Command or Action	Purpose
Step 1	**Example:** `Router# show logging onboard uptime`	Displays stored OBFL data for all nodes or for a specified node. See the *Onboard Failure Logging Commands* module in the *System Monitoring Command Reference for Cisco 8000 Series Routers*.
Step 2	**show process | include obfl** **Example:** `Router# show process | include obfl`	Confirms that the OBFL environmental monitor process is operating.

	Command or Action	Purpose
Step 3	**show process obflmgr** **Example:** `Router# show process obflmgr`	Displays details about the OBFL manager process.

Clearing OBFL Data

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

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

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

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

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

CHAPTER 6

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 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

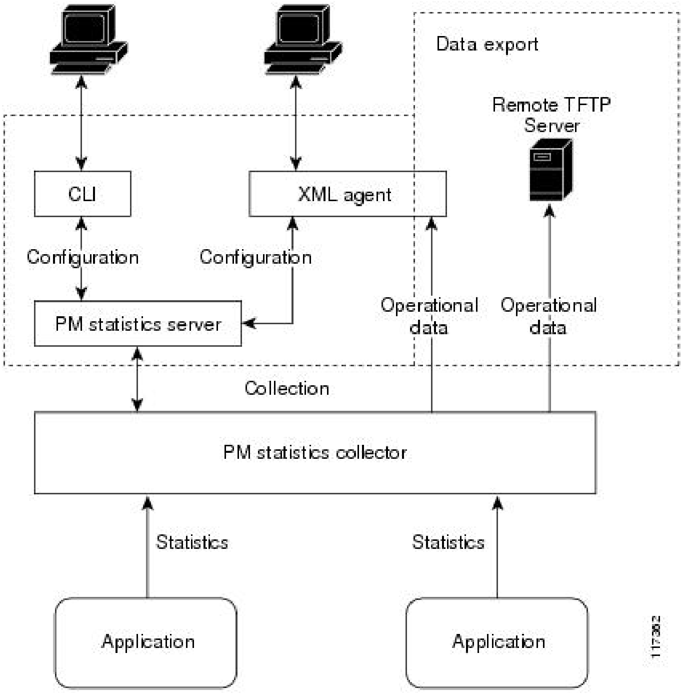
The PM statistics server is the front end for statistic collections, entity instance monitoring collections, and threshold monitoring. All PM statistic collections and threshold conditions configured through the command-line interface (CLI) or through XML schemas are processed by the PM statistics server 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 51 illustrates the relationship between the components that constitute the PM system.

Figure 2: PM Component Communications



PM Benefits

The PM system provides these benefits:

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

PM Statistics Collection Overview

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

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

Table 17: Entity Classes and Associated Instances

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

Note For a list of all attributes associated with the entities that constitute the PM system, see Table 18: Attributes and Values, on page 56.

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

How to Implement Performance Management

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

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

Configuration Examples

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

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

This example configures a local disk for PM statistics collection.

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

Configuring PM Statistics Collection Templates

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

Guidelines for Configuring PM Statistics Collection Templates

When creating PM statistics collection templates, follow these guidelines:

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

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

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

- Node memory
- Node process

Configuration Example

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

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

Configuring PM Threshold Monitoring Templates

The PM system supports the configuration of threshold conditions to monitor an attribute (or attributes) for threshold violations. Threshold conditions are configured through PM threshold monitoring templates. When a PM threshold template is enabled, the PM system monitors all instances of the attribute (or attributes) for the threshold condition configured in the template. If at end of the sample interval a threshold condition is matched, the PM system generates a system logging message for each instance that matches the threshold condition. For the list of attributes and value ranges associated with each attribute for all the entities, see Performance Management: Details, on page 55

Guidelines for Configuring PM Threshold Monitoring Templates

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

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

Configuration Example

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

```
Router# conf t
Router(config)# performance-mgmt thresholds node memory template template20
Router(config-threshold-cpu)# CurrMemory gt 50 percent
Router(config-threshold-cpu)# sample-interval 5
Router(config-threshold-cpu)# exit
Router(config)# performance-mgmt apply thresholds node memory location 0/RP0/CPU0 template20
Router(config)# commit
```

Configuring Instance Filtering by Regular Expression

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

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

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

Configuration Example

This example shows how to define a regular expression group.

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

Performance Management: Details

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

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

Table 18: Attributes and Values

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

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

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

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

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

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

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

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

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

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

Table 19: Entity Instances and Monitoring Commands

Entity	Command Description

BGP	Use the **performance-mgmt apply monitor bgp** command to enable entity instance monitoring for a BGP entity instance.
	Syntax:
	`performance-mgmt apply monitor bgp` *`ip-address template-name`* `|` **`default`**`}`
	`RP/0/RP0/CPU0:Router(config)# performance-mgmt apply monitor bgp 10.12.0.4 default`

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

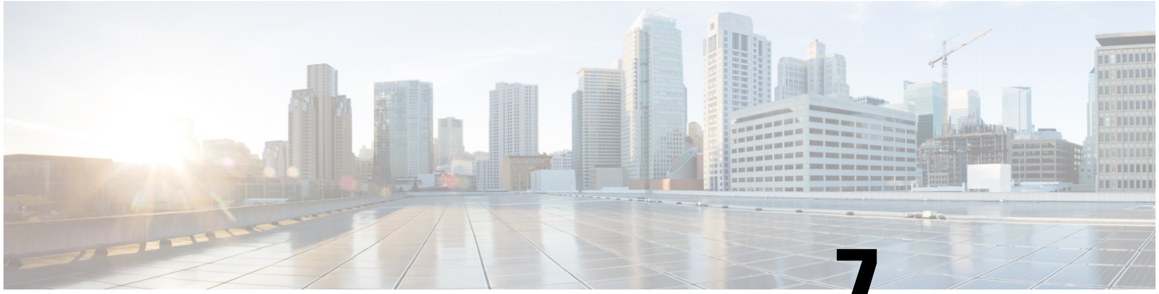
Entity	Command Description

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

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

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

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



CHAPTER 7

System Health Check

System Health Check

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

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

This table describes the significant fields shown in the display.

Table 20: System Health Check Metrics

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

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

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

The system resources exhibit one of these states:

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

The infrastructure services show one of these states:

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

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

You can configure the metrics and their values using CLI. In addition to the CLI, the service supports NETCONF client to apply configuration (`Cisco-IOS-XR-healthcheck-cfg.yang`) and retrieve operational data (`Cisco-IOS-XR-healthcheck-oper.yang`) using YANG data models. It also supports subscribing to metrics and their reports to stream telemetry data. For more information about streaming telemetry data, see *Telemetry Configuration Guide for Cisco 8000 Series Routers*.

Configure Health Check

To enable health check, you must configure the following:

- **netconf-yang agent ssh**
- **healthcheck enable**
- From IOS XR Release 7.3.3 onwards, you must also enable Google Remote Procedure Call (gRPC) using the command **grpc local-connection**.

Configuration Example

```
Router# config
Router(config)# netconf-yang agent ssh
Router(config)# grpc local-connection
Router(config)# healthcheck enable
Router(config)# commit
```

To change the preset cadence, use the **healthcheck cadence** *cadence-value* command:

```
Router(config)#healthcheck cadence 30
```

Note

- Healthcheck use-cases will not work if **grpc no-tls** is configured.

Monitoring Critical System Resources

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

Procedure

Step 1 Check the status of all metrics with its associated threshold and configured parameters in the system.

Example:

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

Collector Cadence: 60 seconds

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

       Tracked CPU utilization: 15 min avg utilization

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

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

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

Infra Services metrics
   fpd

   fabric-health
```

Step 2 View the health state for each enabled metric.

Example:

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

cpu
  State: Normal

free-momry
   State: Normal

shared-memory
```

```
   State: Normal

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

fabric-health
   State: Normal
```

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

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

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

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

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

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

Step 3 Customize the health check threshold value for the following parameters:

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

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

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

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

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

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

Monitoring Infrastructure Services

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

Procedure

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

Example:

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

```
Router# show healthcheck metric platform
Platform Metric State: Normal ==========> Health of the metric
Last Update Time: 25 Jun 05:17:03.508172 =====> Timestamp at which the metric data was collected
Platform Service State: Enabled =====> Service state of Platform
Number of Racks: 1 ======> Total number of racks in the testbed
Rack Name: 0
Number of Slots: 12
Slot Name: RP0
Number of Instances: 2
Instance Name: CPU0
Node Name 0/RP0/CPU0
Card Type 8800-RP
Card Redundancy State Active
Admin State NSHUT
Oper State IOS XR RUN
```

Example:

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

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

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

Example:

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

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

Example:

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

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

Monitoring Counters

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

From IOS XR Release 7.3.5 onwards, all interfaces, including bundles, sub-interfaces, physical interfaces, can be monitored via health check. Previously, only physical interfaces could be monitored.

Procedure

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

Example:

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

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

Example:

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

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

Example:

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

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

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

Example:

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

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

Example:

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

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

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

Example:

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

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

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

Example:

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

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

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

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

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

Example:

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

```
Router# show healthcheck metric fabric-stats summary
Thu Jun 25 06:51:13.154 UTC
Fabric-stats Health State: Normal ==========> Health of the metric
Last Update Time: 25 Jun 06:51:05.669753 =====> Timestamp at which the metric data was collected
Fabric-stats Service State: Enabled =====> Service state of the metric
```

```
Fabric plane id 0 =====> Plane ID
Counter-Names Count Average Consistently-Increasing
------------------------------------------------------------------------------------------------
mcast-lost-cells 0 0 N
Counter-Names =====> Name of the counters
Count =====> Value of the counter collected at "Last Update Time"
Average =====> Average of all values available in buffer
Consistently-Increasing =====> Trend of the counter values, as per data available in buffer
```

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

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

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

Example:

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

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

Example:

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

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

Example:

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

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

SSD health metrics

A SSD health metric is a storage performance indicator that

- reflects the current state and reliability of a Solid State Drive (SSD)
- utilizes Self-Monitoring, Analysis, and Reporting Technology (SMART) to collect real-time data on drive health, and
- helps predict and prevent hardware failures by alerting users to potential issues before they affect operations.

Table 21: Feature History Table

Feature Name	Release Information	Feature Description
Monitor SSD health	Release 26.1.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q100, Q200, P100], 8700 [ASIC: P100, K100], 8010 [ASIC: A100]); Centralized Systems (8600 [ASIC: Q200]); Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) You can now maintain optimal health of Solid State Drives (SSDs) and promptly identify hardware issues to minimize the downtime of your router. The SSD in your router is crucial, as it stores critical data and system files necessary for the router's operation. To ensure proper working of the router, the feature utilizes Self-Monitoring, Analysis, and Reporting Technology (SMART) to monitor SSDs. By default, SMART continuously performs self-tests and logs errors and warnings to specific log files. You can promptly identify hardware issues by reviewing these logs. This feature introduces these changes: **CLI:** • show smart-monitor **YANG Date Model:** • openconfig-platform-storage.yang The YANG model provides several health metrics for gNMI subscribe and also supports NETCONF for GET requests. (See GitHub, YANG Data Models Navigator)

Metrics to monitor SSD health

These metrics help network administrators monitor SSD health proactively to prevent unplanned outages or data loss by identifying drives at risk of failure early and enabling timely replacement or maintenance:

- Soft read error rate: Tracks uncorrected read errors on the drive.
- Reallocated sectors: Counts bad sectors found and remapped by the controller.
- End to end error: Counts parity errors that may occur in the data path.
- Offline uncorrectable sectors count: Shows the total number of errors that cannot be corrected by the device.
- Life left: Indicates the approximate remaining lifespan of the SSD.
- Percentage used: Displays the manufacturer’s estimated total usage for the drive subsystem.

System Health Check Use-Cases

Table 22: Feature History Table

Feature Name	Release Information	Feature Description
System Health Check Use-cases	Release 7.3.3 Release 7.5.4	System Health Check use-cases are a version of the system health check where the user can monitor specific metrics of the system to determine the health and detect potential failures in the system caused by ASIC reset or packet drops. When seen from the device health point of view, it is conceptually determining and analyzing metrics that detect anomalies in the router. When the metric degrades beyond a certain threshold, the router itself raises the alarm. This service supports NETCONF client retrieve operational data using the following YANG data models: • Cisco-IOS-XR-ofa-npu-stats-oper.yang • Cisco-IOS-XR-infra-syslog-oper.yang This feature introduces two new keywords in the system health check metrics use-case: • asic-reset • packet-drop

System Health Check use-cases are an enhanced version of the system health check where you can monitor NPU traffic related counters. This service helps to monitor, track and analyze these metrics to detect failures in the system caused by:

- ASIC resets
- packet drops

The system health can be gauged with the values reported by these metrics when the configured threshold values exceed or are nearing the threshold value. This feature determines the Packet forwarding state inside the router, and the data is collected and plotted with respect to time to determine if there are any failures that can affect the packet forwarding state of the router. When seen from the device health point of view, it is conceptually determining and analyzing metrics that detect anomalies in the router. When the metric degrades beyond a certain threshold, the router itself raises the alarm.

Once enabled, it collects metrics from Syslogs, NPU traps, and NPU packet counters. It then analyses the raw data per metric and transforms them into actionable metrics. It then correlates the metrics based on the use case and if all conditions are met, it reports the event as a gray-failure. The user can use can then take action and troubleshoot as required.

System Health check and use-cases are not part of the base package and you must explicitly install the '*xr-healthcheck*' optional package to use this service.

Note In Health check use-cases, packet drop is calculated as the total Bytes egressing the NPU subtracted by total bytes ingressing the NPU. If the traffic arriving on the npu via physical ports/interfaces is less than the Inter-fabric traffic on the Cisco 8000 Distributed platform then the trends will not be seen. This will be updated in a future.

Feature Behavior and Guidelines

- Feature drops such as ACL and QoS are also treated as NPU drops.
- Packet replicating features like Multicast, SPAN, LI, can lead to missed packet drop trends.
- If the Total traffic on a given NPU is less than 10mbps, the trends will not trigger an alarm

Trends Supported by Health Check Use-cases

The use-cases, ASIC resets and packet drops, demonstrate three trends:

1. Peak
2. Plateau
3. Recovery

Peak Trend

A Peak trend is observed when there is a sudden spike in the packet drop or npu traps count and the percentage of this spike is higher than the configured tolerance limit.To verify the configured tolerance limit use the **show healthcheck status** command and see the *drop tolerance* value in the output.

Example:

```
asic_reset
       Drop tolerance: 10

   packet_drop
       Drop tolerance: 10
```

Plateau Trend

A Plateau trend is observed when the packet drop or npu traps count remains higher than the tolerance limit for ten consecutive cadence intervals. Cadence is the time period used by the health check use-cases to examine the data received, transmitted packets and the npu traps per npu.To verify the configured cadence value, use the **show healthcheck status** command and see the *Collector Cadence* value in the output.

Example:

```
 Collector Cadence: 30 seconds
```

Recovery Trend

After Peak or Plateau trend, if the packet drop and npu traps count stays within the tolerance limits for ten cadence intervals, then the recovery trend is seen.

ASIC Reset Use-case and Monitoring

A soft reset of the NPU takes place when pre-determined set of *error-interrupts* occur which are serviced by the NPU-driver. In this case, the recommended action is to reset specific set of blocks inside the ASIC. After reset of the ASIC , the NPU-driver will check if these interruptions occur again within a certain time window. This use-case intends to detect these scenarios and alarm the user that traffic did not recover after the ASIC reset.

This section explains how to check the health of a system ASIC reset information. In this use-case, if the NPU does not recover fully from a soft reset and traffic gets dropped, it gets detected and an alarm is triggered.

Limitations and Restrictions for ASIC Reset Use-case and Monitoring

The Cisco IOS-XR Release 24.4.1 does not support ASIC reset use-case and monitoring.

Configure ASIC Reset Monitoring

Configure the use-case asic-reset drop tolerance threshold. If the traffic input is below the configured threshold, an alarm is triggered.

```
Router(config)# healthcheck
Router(config-healthcheck)# use-case asic-reset drop-tolerance 10
Router(config-healthcheck)# enable
```

Note You can re-configure the drop-tolerance based on the expected drops in your network.

Retrieve ASIC Reset Trends

This example shows how to obtain the status for the asic-reset use-case:

```
Router# show healthcheck use-case asic-reset detail npu all location all
Mon Nov 29 05:06:51.240 UTS
Node: 0/0/CPU0             NPU Instance: 0
 Timestamp: Mon 2021-11-29 05:06:29 GMT
Alerts     :
0. asic reset for NPU 0 location 0/0/CPU0 triggered at Dec 10 2020 10:52:34
1. peak detected in queue drops for NPU 0 location 0/0/CPU1

Node: 0/0/CPU0             NPU Instance: 1
 Timestamp: Mon 2021-11-29 05:06:29 GMT
Alerts     :
0. asic reset for NPU 0 location 0/0/CPU0 triggered at Dec 10 2020 10:52:34
1. peak detected in queue drops for NPU 0 location 0/0/CPU1
```

Show Command Examples for Asic Reset Use-case

Detailed show outputs of this use-case listed below shows the trend of the packet-drops and the time-stamp when asic reset was triggered. If the asic reset stopped the packet drop then the packet-drop trend moves to Recovery and the status is shown as normal.

Initial show output when no asic-reset is triggered:

```
Router# show healthcheck use-case asic-reset summary
Mon Jun  5 11:13:51.901 IST
Use Case Name: asic_reset
Timestamp    : Mon 2023-06-05 11:13:51 IST
State        : Normal
```

Syslogs when asic-reset is triggered

```
 ----------------------------------------------
Router:Jun  5 11:17:31.664 IST: npu_drvr[299]: %FABRIC-NPU_DRVR-3-ASIC_ERROR_ACTION : [8698]

: npu[0]: HARD_RESET needed for hmc_cgm.cgm_int.total_buffers_underflow
Router:Jun  5 11:19:17.883 IST: NOSi[66650]: %OS-SYSLOG-6-DOCKER_APP : 2023-06-05 11:19:17,882
 [WARNING ] NOSi: 2023-06-05 11:19:01 0/1/CPU0 {NPU:0} :PACKET DROP ALERT.
Waiting for dropped packets to recover post asic reset.
```

From IOS XR Release 25.1.1 onwards, the router displays this syslog for asic-reset.

```
Router:Jun 11 10:50:53.343 UTC: NOSi[66902]: %OS-SYSLOG-6-DOCKER_APP :
 2025-06-11 10:50:53,342 [WARNING ] NOSi: 2025-06-11 10:48:22 {LOCATION:0/0/CPU0}
{NPU/Instance:2}
 {CAUSE:} :PACKET DROP ALERT. Waiting for dropped packets to recover post asic reset.
```

Show output when asic-reset is triggered

```
Router# show healthcheck use-case asic-reset summary
Mon Jun  5 11:20:28.307 IST
Use Case Name: asic_reset
Timestamp    : Mon 2023-06-05 11:20:02 IST
State        : Warning
Alert        : Usecase asic_reset has warnings and alerts


 Router# show healthcheck use-case asic-reset detail npu all location all
Mon Jun  5 11:20:35.330 IST
Node: 0/1/CPU0     NPU Instance: 0
    Timestamp : Mon 2023-06-05 11:20:02 IST
    Alerts    :
        0. packet-counters location: node-name: 0/1/CPU0 npu-id: 0 trend: PLATEAU at
2023-06-05 11:20:02
        1. syslog location: node-name: 0/1/CPU0 npu-id: 0 event at 2023-06-05 11:17:31
```

Syslog when packet drops recovered after asic-reset:

```
Router:Jun  5 11:28:18.551 IST: NOSi[66650]: %OS-SYSLOG-6-DOCKER_APP : 2023-06-05
11:28:18,550 [INFO    ] NOSi: 2023-06-05 11:28:02 0/1/CPU0 {NPU:0} :PACKET DROP ALERT
CLEARED.
Dropped packet counters within tolerance limits post asic reset
```

From IOS XR Release 25.1.1 onwards, the router displays this syslog when packet drops recovered after asic-reset.

```
Router:Jun 11 11:08:53.337 UTC: NOSi[66902]: %OS-SYSLOG-6-DOCKER_APP : 2025-06-11
 11:08:53,335 [INFO    ] NOSi: 2025-06-11 10:48:22 {LOCATION:0/0/CPU0} {NPU/Instance:2}
{CAUSE:} :PACKET DROP ALERT CLEARED.
Dropped packet counters within tolerance limits post asic reset
```

Show commands after packet drops recovered

```
Router# show healthcheck use-case asic-reset summary
Mon Jun  5 11:28:47.961 IST
Use Case Name: asic_reset
Timestamp    : Mon 2023-06-05 11:28:02 IST
State        : Normal
```

```
Router# show healthcheck use-case asic-reset detail npu all location all
Mon Jun  5 11:28:54.111 IST
Node: 0/1/CPU0     NPU Instance: 0
    Timestamp : Mon 2023-06-05 11:28:02 IST
    Alerts    :
        0. syslog location: node-name: 0/1/CPU0 npu-id: 0 event at 2023-06-05 11:17:31
        1. packet-counters location: node-name: 0/1/CPU0 npu-id: 0 trend: RECOVERY at
2023-06-05 11:28:02
```

Packet drop use-case and monitoring

Table 23: Feature History Table

Feature Name	Release Information	Feature Description
Improved packet loss detection and monitoring	Release 25.4.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100]) *This feature is supported on Cisco 8711-48Z-M routers
Improved packet loss detection and monitoring	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on Cisco 8011-4G24Y4H-I routers.

Feature Name	Release Information	Feature Description
Improved packet loss detection and monitoring	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100, K100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) You can now see the historical trend data for the packet-drop use-case, with detailed granularity on a per node and per NPU basis. This feature introduces these changes: **CLI:** The **history** keyword is introduced in the **show healthcheck use-case packet-drop** command. *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • Cisco 8712-MOD-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

Feature Name	Release Information	Feature Description
Improved packet loss detection and monitoring	Release 24.1.1	Introduced in this release on: Fixed Systems (8200); Centralized Systems (8600); Modular Systems (8800 [LC ASIC: Q100, Q200]) You can now set the tolerance to monitor packet drops in the Network Processing Unit (NPU) for three different NPU trap categories at a configurable cadence of your choice. When a NPU trap breaks the configured packet-drop tolerance for that trap category, the router alerts you with a system log message and you can monitor the trend using healthcheck show commands. This enables you to prioritize taking action depending on the trap category for which the router logs the message. In earlier releases, you could only monitor packet-drops globally for all NPU trap cateogories at a fixed cadence. This feature introduces these changes: **CLI:** The **tolerance** and **window-size** keywords are introduced in the **use-case** command. **YANG Data Model:** New xpaths for `Cisco-IOS-XR-healthcheck-cfg.yang` data model. (see GitHub, YANG Data Models Navigator)

NPU traps are signals that the NPU raises in response to certain types of packets received by the router, such as errored packets, packets that will be dropped by the router, or packets that require extra processing by the CPU.

Packet-drop use-case checks the health of the system by monitoring NPU traps to check for packets dropped per NPU for the cadence interval. You can enable the packet-drop use-case using the **use-case packet-drop** command, In this use-case, you can monitor packet-drops in the NPU by configuring a global drop-tolerance value for all NPU traps for a fixed number of cadence intervals.

From Cisco IOS XR Release 24.1.1 onwards, instead of configuring a global drop-tolerance, you can configure separate drop-tolerance values for low, medium and high tolerance NPU traps. You can also configure the number of cadence intervals to alert you of packet drops.

If the packet-drops exceed configured drop-tolerance rates and continues over the set cadence interval, the router software detects it and generates a system log message.

Apart from this, based on the tolerance limit configured the router software also shows the system health trends in the output of the **show healthcheck use-case packet-drop detail npu all location all** command.

Configure packet-drop monitoring

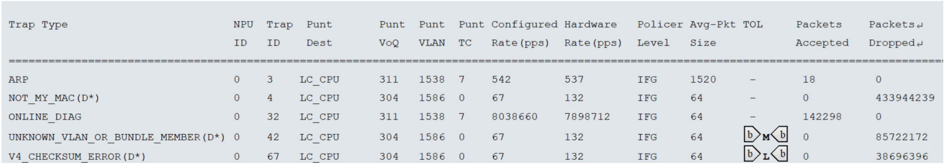
Prior to Cisco IOS XR Release 24.1.1, you could set the **drop-tolerance** threshold as shown in the following code-block.

```
Router(config)# healthcheck
Router(config-healthcheck)# use-case packet-drop drop-tolerance 10
Router(config-healthcheck)# enable
```

From Cisco IOS XR Release 24.1.1 onwards, you must configure separate drop-tolerance values for low, medium or high tolerance NPU traps. You can also configure the cadence interval.

1. **Check NPU traps tolerance:** Execute the **show controllers npu stats traps-all detail instance all location all** command to check whether an NPU trap is low, medium or high tolerance. The **TOL** column in the following show command output shows the tolerance levels for the different NPU traps.

```
Router# show controllers npu stats traps-all detail instance all location all
Trap stats for 0/RP0/CPU0
Traps marked (D*) are punted (post policing) to the local CPU internal VLAN 1586 for
debugging
They can be read using "show captured packets traps" CLI
Traps marked (D) are dropped in the NPU
TOL represent tolerance level for healthcheck monitoring
 - TOL H/M/L represent High/Medium/Low tolerance
Traps punted to internal VLAN 1538 are processed by the process "spp" on the "Punt Dest"
 CPU
They can also be read using "show captured packets traps" CLI
"Configured Rate" is the rate configured by user (or default setting) in pps at the LC
level
"Hardware Rate" is the actual rate in effect after hardware adjustments
Policer Level:
        NPU: Trap meter is setup per NPU in packets per second
        IFG: Trap meter is setup at every IFG in bits per second
        The per IFG meter is converted from the user configured/default rate (pps)
        based on the "Avg-Pkt Size" into bps.
        Due to hardware adjustments, the "Configured Rate" and
        "Hardware Rate" differ in values.

NOTE:The displayed stats are NOT real-time and are updated every 30 SECONDS from the
hardware.
```



```
Trap Type                          NPU  Trap  Punt      Punt  Punt  Punt Configured Hardware   Policer Avg-Pkt TOL   Packets   Packets↵
                                   ID   ID    Dest      VoQ   VLAN  TC   Rate(pps)  Rate(pps)  Level   Size          Accepted  Dropped↵
=======================================================================================================================================
ARP                                0    3    LC_CPU     311   1538  7    542        537        IFG     1520     -     18        0
NOT_MY_MAC(D*)                     0    4    LC_CPU     304   1586  0    67         132        IFG     64       -     0         433944239
ONLINE_DIAG                        0    32   LC_CPU     311   1538  7    8038660    7898712    IFG     64       -     142298    0
UNKNOWN_VLAN_OR_BUNDLE_MEMBER(D*)  0    42   LC_CPU     304   1586  0    67         132        IFG     64       M     0         85722172
V4_CHECKSUM_ERROR(D*)              0    67   LC_CPU     304   1586  0    67         132        IFG     64       L     0         38696396
```

524609

2. **Configure tolerance and window-size:** Specify the NPU-trap tolerance level and configure the drop-tolerance using the **tolerance** keyword. Configure cadence intervals to alert you of packet-drops using the **window-size** keyword, as shown below:

   ```
   Router# conf t
   Router(config)# healthcheck
   Router(config-healthcheck)# use-case packet-drop tolerance high 300000
   Router(config-healthcheck)# use-case packet-drop window-size 5
   Router(config-healthcheck)# enable
   Router(config-healthcheck)# commit
   ```

Retrieve packet-drop trends

The following example shows how to obtain the packet-drop use-case trends prior to Cisco IOS XR Release 24.1.1:

```
Router# show healthcheck use-case packet-drop detail npu all location all

Node: 0_0_CPU0     NPU Instance: 0
    Timestamp : Thu 2021-09-02 21:41:00 UTC
    Alerts    :
        0. npu-traps-sum location: node-name: 0/0/CPU0 npu-id: 0 trend: PEAK at 2021-09-02
 21:41:00
       1. packet-counters location: node-name: 0/0/CPU0 npu-id: 0 trend: PEAK at 2021-09-02
 21:40:51
Node: 0_1_CPU0     NPU Instance: 2
    Timestamp : Thu 2021-09-02 21:42:30 UTC
    Alerts    :
```

The following example shows the packet-drop use-case trends from Cisco IOS XR Release 24.1.1 onwards:

```
Router# show healthcheck use-case packet-drop detail npu all location all
Node: 0/3/CPU0     NPU Instance: 0
    Timestamp : Thu 2024-02-08 17:43:45 UTC
    Alerts    :
        1. npu-traps location: node-name: 0/3/CPU0 npu-id: 0 trap-string:
UNKNOWN_VLAN_OR_BUNDLE_MEMBER(D*) trend: RECOVERY at 2024-02-08 17:43:45
Node: 0/3/CPU0     NPU Instance: 0
    Timestamp : Fri 2024-02-09 04:12:25 UTC
    Alerts    :
        1. npu-traps location: node-name: 0/3/CPU0 npu-id: 0 trap-string:
V4_CHECKSUM_ERROR(D*) trend: PLATEAU at 2024-02-09 04:12:22
```

Show command examples for packet drops use-case

Show outputs of this use-case listed below shows the trend of the packet-drops use-case.

Initial show output when no packet-drop is seen:

```
Router# show healthcheck use-case packet-drop summary
Mon Jun  5 10:38:25.885 IST
Use Case Name: packet_drop
Timestamp    : Mon 2023-06-05 10:38:25 IST
State        : Normal
```

The Syslog format is changed from Release 24.4.1.

In IOS XR Releases before 24.4.1, the router displays this syslog for peak trend.

```
Syslog when peak trend is seen:
Router:Jun 5 10:59:19.164 IST: NOSi[66650]: %OS-SYSLOG-6-DOCKER_APP : 2023-06-05 10:59:19,164
[WARNING ] NOSi: 2023-06-05
```

```
10:59:18 0/1/CPU0 {NPU:0} :PACKET DROP ALERT.
Dropped packets shows PEAK trend due to PEAK trend observed for NPU traps
```

Only for IOS XR Release 24.4.1, the router displays this syslog for peak trend.

```
Router:Nov 25 08:35:30.541 UTC: NOSi[66820]: %OS-SYSLOG-6-DOCKER_APP :
2024-11-25 08:35:30,538 [WARNING ] NOSi: [PACKET_DROP_ALERT] [PEAK] 2024-11-25 08:35:26
{LOCATION: 0/0/CPU0}
{NPU/Instance: 0 } {CAUSE: V4_CHECKSUM_ERROR(D*)  }
```

From IOS XR Release 25.1.1 onwards, the router displays this syslog for peak trend.

```
RP/0/RP0/CPU0:Jun 11 10:49:59.396 UTC: NOSi[66902]: %OS-SYSLOG-6-DOCKER_APP :
 2025-06-11 10:49:59,396 [WARNING ] NOSi: [PACKET_DROP_ALERT] [PEAK] 2025-06-11 10:49:50
{LOCATION:0/0/CPU0}
 {NPU/Instance:2} {CAUSE:V4_CHECKSUM_ERROR(D*)}
```

Show command output when peak trend is seen:

```
Router# show healthcheck use-case packet-drop summary
Mon Jun  5 10:59:30.800 IST
Use Case Name: packet_drop
Timestamp    : Mon 2023-06-05 10:59:18 IST
State        : Warning
Alert        : Usecase packet_drop has warnings and alerts
```

```
Router# show healthcheck use-case packet-drop detail npu all location all
Mon Nov 25 08:35:47.701 UTC
Node: 0/0/CPU0     NPU Instance: 0
    Timestamp : Mon 2024-11-25 08:35:29 UTC
    Alerts    :
        1. npu-traps location: node-name: 0/0/CPU0 npu-id: 0 trap-string:
V4_CHECKSUM_ERROR(D*) tolerance: LOW trend: PEAK at 2024-11-25 08:35:26
```

In IOS XR Releases before 24.4.1, the router displays this syslog for plateau trend.

```
Router:Jun 5 11:03:49.417 IST: NOSi[66650]: %OS-SYSLOG-6-DOCKER_APP : 2023-06-05 11:03:49,416
[WARNING ] NOSi: 2023-06-05
11:03:48 0/1/CPU0 {NPU:0} :PACKET DROP ALERT.
Dropped packets shows PLATEAU trend due to PLATEAU trend observed for NPU traps
```

Only for IOS XR Release 24.4.1, the router displays this syslog for plateau trend.

```
Router:Nov 25 08:40:00.472 UTC: NOSi[66820]: %OS-SYSLOG-6-DOCKER_APP :
2024-11-25 08:40:00,470 [WARNING ] NOSi: [PACKET_DROP_ALERT] [PLATEAU] 2024-11-25 08:39:56
 {LOCATION: 0/0/CPU0}
{NPU/Instance: 0 } {CAUSE: V4_CHECKSUM_ERROR(D*)  }
```

From IOS XR Release 25.1.1 onwards, the router displays this syslog for plateau trend.

```
Router:Jun 11 10:58:59.476 UTC: NOSi[66902]: %OS-SYSLOG-6-DOCKER_APP :
 2025-06-11 10:58:59,476 [WARNING ] NOSi: [PACKET_DROP_ALERT] [PLATEAU] 2025-06-11 10:58:50
 {LOCATION:0/0/CPU0}
 {NPU/Instance:2} {CAUSE:V4_CHECKSUM_ERROR(D*)}
```

Show command output when plateau trend is seen:

```
Router# show healthcheck use-case packet-drop summary
Mon Jun  5 11:05:34.428 IST
Use Case Name: packet_drop
Timestamp    : Mon 2023-06-05 11:03:48 IST
State        : Warning
Alert        : Usecase packet_drop has warnings and alerts
```

```
Router# show healthcheck use-case packet-drop detail npu all location all
Mon Nov 25 08:40:39.843 UTC
Node: 0/0/CPU0     NPU Instance: 0
```

```
    Timestamp : Mon 2024-11-25 08:39:59 UTC
    Alerts    :
        1. npu-traps location: node-name: 0/0/CPU0 npu-id: 0 trap-string:
V4_CHECKSUM_ERROR(D*) tolerance: LOW trend: PLATEAU at 2024-11-25 08:39:56
```

In IOS XR Releases before 24.4.1, the router displays this syslog for recovery trend.

```
Router:Jun 5 11:11:49.866 IST: NOSi[66650]: %OS-SYSLOG-6-DOCKER_APP : 2023-06-05
11:11:49,865 [INFO ] NOSi: 2023-06-05 11:11:49 0/1/CPU0 {NPU:0} :PACKET DROP ALERT
CLEARED.
NPU trap and dropped packet counters within tolerance limits
```

Only for IOS XR Release 24.4.1, the router displays this syslog for recovery trend.

```
Router:Nov 25 08:46:30.431 UTC: NOSi[66820]: %OS-SYSLOG-6-DOCKER_APP :
2024-11-25 08:46:30,429 [WARNING ] NOSi: [PACKET_DROP_ALERT] [RECOVERY] 2024-11-25 08:46:26
 {LOCATION: 0/0/CPU0}
{NPU/Instance: 0 } {CAUSE: V4_CHECKSUM_ERROR(D*)  }
```

From IOS XR Release 25.1.1 onwards, the router displays this syslog for recovery trend.

```
Router:Jun 11 11:08:59.287 UTC: NOSi[66902]: %OS-SYSLOG-6-DOCKER_APP :
 2025-06-11 11:08:59,286 [WARNING ] NOSi: [PACKET_DROP_ALERT] [RECOVERY] 2025-06-11 11:08:50
 {LOCATION:0/0/CPU0}
 {NPU/Instance:2} {CAUSE:V4_CHECKSUM_ERROR(D*)}
```

Show command output when Recovery trend is seen:

```
Router# show healthcheck use-case packet-drop summary
Mon Jun  5 11:12:59.387 IST
Use Case Name: packet_drop
Timestamp    : Mon 2023-06-05 11:11:49 IST
State        : Normal


Router# show healthcheck use-case packet-drop detail npu all location all
Mon Nov 25 08:46:57.484 UTC
Node: 0/0/CPU0     NPU Instance: 0
    Timestamp : Mon 2024-11-25 08:46:29 UTC
    Alerts    :
        1. npu-traps location: node-name: 0/0/CPU0 npu-id: 0 trap-string:
V4_CHECKSUM_ERROR(D*) tolerance: LOW trend: RECOVERY at 2024-11-25 08:46:26
```

The option to view history is available only from IOS XR Release 24.4.1 onwards.

Show command output to see the history of use-case packet-drop:

```
Router# show healthcheck use-case packet-drop history npu 0 location 0/0/CPU0
Node: 0/0/CPU0     NPU Instance: 0
Timestamp:
Alerts    :
        1. npu-traps location: node-name: 0/0/CPU0 npu-id: 0 trap-string:
UNKNOWN_VLAN_OR_BUNDLE_MEMBER(D*) trend: PEAK at 2023-11-28 07:11:37
        2. npu-drops location: node-name: 0/0/CPU0 npu-id: 0 field-name:
counters.drop.slice[0].ifg[0].ifgb.port_4.rx_undersize_pkt_ severity: ERROR trend: PEAK at
 2023-11-28 07:11:37
        3. npu-traps location: node-name: 0/0/CPU0 npu-id: 0 trap-string:
V4_CHECKSUM_ERROR(D*) trend: PLATEAU at 2023-11-28 07:13:37
        4. npu-traps location: node-name: 0/0/CPU0 npu-id: 0 trap-string:
UNKNOWN_VLAN_OR_BUNDLE_MEMBER(D*) trend: PLATEAU at 2023-11-28 07:13:37
        5. npu-drops location: node-name: 0/0/CPU0 npu-id: 0 field-name:
counters.drop.slice[0].ifg[0].ifgb.port_4.rx_undersize_pkt_ severity: ERROR trend: PLATEAU
 at 2023-11-28 07:21:37
        6. npu-traps location: node-name: 0/0/CPU0 npu-id: 0 trap-string:
UNKNOWN_VLAN_OR_BUNDLE_MEMBER(D*) trend: PLATEAU at 2023-11-28 07:31:09
        7. npu-traps location: node-name: 0/0/CPU0 npu-id: 0 trap-string:
V4_CHECKSUM_ERROR(D*) trend: PLATEAU at 2023-11-28 07:52:39
```

```
        8. npu-traps location: node-name: 0/0/CPU0 npu-id: 0 trap-string:
UNKNOWN_VLAN_OR_BUNDLE_MEMBER(D*) trend: PLATEAU at 2023-11-28 07:52:39
        9. npu-traps location: node-name: 0/0/CPU0 npu-id: 0 trap-string:
UNKNOWN_VLAN_OR_BUNDLE_MEMBER(D*) trend: RECOVERY at 2023-11-28 08:04:40
        10. npu-traps location: node-name: 0/0/CPU0 npu-id: 0 trap-string:
V4_CHECKSUM_ERROR(D*) trend: RECOVERY at 2023-11-28 08:04:40
```

Retrieval of Data

For the purpose of analyzing the metrics or troubleshooting once an alarm is raised, you can retrieve the data. The data can be retrieved using CLI. You can use the following show commands to retrieve the data:

- **show healthcheck use-case asic-reset detail npu all location all**
- **show healthcheck use-case packet-drop detail npu all location all**

The service supports NETCONF client to retrieve operational data (*Cisco-IOS-XR-ofa-npu-stats-oper.yang* and *Cisco-IOS-XR-infra-syslog-oper.yang*) using YANG data models.

It also supports subscribing to metrics and their reports to stream telemetry data. For more information about streaming telemetry data, see *Telemetry Configuration Guide for Cisco 8000 Series Routers*. You can also view the data model definitions using the YANG Data Models Navigator tool.

CHAPTER 8

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

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 91.

Writing an EEM script includes the following steps:

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

Regular Embedded Event Manager Scripts

Regular EEM scripts are used to implement policies when an EEM event is published. EEM scripts are identified to the EEM using the **event manager policy** configuration command. An EEM script remains available to be scheduled by the EEM until the **no event manager policy** command is entered.

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

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

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

Embedded Event Manager Policy Tcl Command Extension Categories

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

Table 24: Embedded Event Manager Tcl Command Extension Categories

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

Cisco File Naming Convention for Embedded Event Manager

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

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

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

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

Table 25: Two-Character Abbreviation Specification

Two-Character Abbreviation	Specification
ap	event_register_appl
no	event_register_none

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

Embedded Event Manager Built-in Actions

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

This table describes each EEM handler request or action.

Table 26: Embedded Event Manager Built-In Actions

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

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

Application-specific Embedded Event Management

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

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

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

Event Detection and Recovery

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

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

- System EEM events
- Application-defined events

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

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

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

These event detectors are supported:

System Manager Event Detector

The System Manager Event Detector has four roles:

- Records process reliability metric data.
- Screens for processes that have EEM event monitoring requests outstanding.

- Publishes events for those processes that match the screening criteria.
- Asks the System Manager to perform its default action for those events that do not match the screening criteria.

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

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

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

Table 27: System Manager Event Detector Event Monitoring Requests

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

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

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

Timer Services Event Detector

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

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

Syslog Event Detector

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

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

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

Table 28: Syslog Event Detector Event Monitoring Requests

Embedded Event Manager Event	Description
Syslog message EEM event	Occurs for a just-logged message. It occurs when there is a match for either the syslog message severity code or the syslog message text pattern. Both can be specified when an application requests a syslog message EEM event.
Process event manager EEM event—built in	Occurs when the event-processed count for a specified process is either greater than or equal to a specified maximum or is less than or equal to a specified minimum.

None Event Detector

The None Event Detector publishes an event when the Cisco IOS XR7 software **event manager run** CLI command executes an EEM policy. EEM schedules and runs policies on the basis of an event specification that is contained within the policy itself. An EEM policy must be identified and registered to be permitted to run manually before the **event manager run** command will execute.

Event manager none detector provides user the ability to run a tcl script using the CLI. The script is registered first before running. Cisco IOS XR7 software version provides similar syntax with Cisco IOS EEM (refer to the applicable EEM Documentation for details), so scripts written using Cisco IOS EEM is run on Cisco IOS XR7 software with minimum change.

Distributed Event Detectors

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

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

- System Manager Fault Detector

Embedded Event Manager Event Scheduling and Notification

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

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

Reliability Statistics

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

Hardware Card Reliability Metric Data

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

Data maintained by the hardware card is as follows:

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

Process Reliability Metric Data

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

Process terminations include the following cases:

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

Data to be maintained by process is as follows:

- Most recent process start time
- Most recent normal process end time
- Most recent abnormal process end time
- Most recent abnormal process end type

- Previous ten process end times and types
- Cumulative process available time
- Cumulative process unavailable time
- Cumulative process run time (the time when the process is actually running on the CPU)
- Number of times started
- Number of times ended normally
- Number of times ended abnormally
- Number of abnormal failures within the past 60 minutes
- Number of abnormal failures within the past 24 hours
- Number of abnormal failures within the past 30 days

How to Configure and Manage Embedded Event Manager Policies

Configuring Environmental Variables

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

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

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

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

Configuration Example

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

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

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

```
3    _email_from                  beta@cisco.com
RP/0/RP0/CPU0:Router#
```

Registering Embedded Event Manager Policies

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

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

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

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

Note AAA authorization (such as the **aaa authorization eventmanager** command) must be configured before EEM policies can be registered. See the *Configuring AAA Services* module of *Configuring AAA Services on Cisco IOS XR7 software* for more information about AAA authorization configuration.

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

Configuration Example

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

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

How to Write Embedded Event Manager Policies Using Tcl

This section provides information on how to write and customize Embedded Event Manager (EEM) policies using Tool Command Language (Tcl) scripts to handle Cisco IOS XR7 software faults and events.

This section contains these tasks:

Registering and Defining an EEM Tcl Script

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

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

Configuration Example

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

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

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

Displaying EEM Registered Policies

Perform this optional task to display EEM registered policies.

SUMMARY STEPS

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

DETAILED STEPS

Procedure

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

Unregistering EEM Policies

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

SUMMARY STEPS

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

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**show event manager policy registered** [**event-type** *type*] [**system** | **user**] [**time-ordered** | **name-ordered**] **Example:** `Router# show event manager policy registered system`	Displays information about currently registered policies. • The **event-type** keyword displays the registered policies for a specific event type. • The **time-ordered** keyword displays information about currently registered policies sorted by time. • The **name-ordered** keyword displays the policies in alphabetical order by the policy name.
Step 2	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 3	**no event manager policy** *policy-name* **Example:** `Router(config)# no event manager policy tm_cli_cmd.tcl`	Removes the EEM policy from the configuration, causing the policy to be unregistered.
Step 4	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 5	Repeat step 1 to ensure that the policy has been removed.	—

Suspending EEM Policy Execution

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

Configuration Example

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

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

Specifying a Directory for Storing EEM Policies

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

Configuration Example

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

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

Sample EEM Policies

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

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

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

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

This table describes the sample EEM policies.

Table 29: Sample EEM Policy Descriptions

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

SUMMARY STEPS

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

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**show event manager policy available** [**system** | **user**] **Example:** `Router# show event manager policy available`	Displays EEM policies that are available to be registered.
Step 2	**configure** **Example:**	Enters global configuration mode.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router# configure`	
Step 3	**event manager directory user** {**library** *path* | **policy** *path*} **Example:** `Router(config)# event manager directory user library harddisk:/mirror/EEM/user_library`	Specifies a directory to use for storing user library files or user-defined EEM policies.
Step 4	**event manager policy** *policy-name* **username** *username* [**persist-time** [*seconds* | **infinite**] | **type** [**system** | **user**]] **Example:** `Router(config)# event manager policy test.tcl username user_a type user`	Registers the EEM policy to be run when the specified event defined within the policy occurs.
Step 5	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Programming EEM Policies with Tcl

Perform this task to help you program a policy using Tcl command extensions. We recommend that you copy an existing policy and modify it. There are two required parts that must exist in an EEM Tcl policy: the event_register Tcl command extension and the body. For detailed information about the Tcl policy structure and requirements, see EEM Policies Using TCL: Details, on page 114

Procedure

	Command or Action	Purpose
Step 1	**show event manager policy available** [**system** | **user**] **Example:** `RP/0/RP0/CPU0:Router# show event manager policy available`	Displays EEM policies that are available to be registered.
Step 2	Cut and paste the contents of the sample policy displayed on the screen to a text editor.	—

	Command or Action	Purpose
Step 3	Define the required event_register Tcl command extension.	Choose the appropriate event_register Tcl command extension for the event that you want to detect, and add it to the policy. The following are valid Event Registration Tcl Command Extensions: • event_register_appl • event_register_oir • event_register_process • event_register_syslog • event_register_timer • event_register_timer_subscriber • event_register_none
Step 4	Add the appropriate namespace under the ::cisco hierarchy.	Policy developers can use the new namespace ::cisco in Tcl policies to group all the extensions used by Cisco IOS XR EEM. There are two namespaces under the ::cisco hierarchy. The following are the namespaces and the EEM Tcl command extension categories that belongs under each namespace: • ::cisco::eem • EEM event registration • EEM event information • EEM event publish • EEM action • EEM utility • EEM context library • EEM system information • CLI library • ::cisco::lib • SMTP library **Note** Ensure that the appropriate namespaces are imported, or use the qualified command names when using the preceding commands.
Step 5	Program the must defines section to check for each environment variable that is used in this policy.	This is an optional step. Must defines is a section of the policy that tests whether any EEM environment variables

	Command or Action	Purpose
		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 "_". **Note** You can display the Embedded Event Manager environment variables set on your system by using the **show event manager environment** command. For example, EEM environment variables defined by the sample policies include e-mail variables. The sample policies that send e-mail must have the following variables set in order to function properly. The following are the e-mail-specific environment variables used in the sample EEM policies. • **_email_server**—A Simple Mail Transfer Protocol (SMTP) mail server used to send e-mail (for example, mailserver.example.com) • **_email_to**—The address to which e-mail is sent (for example, engineering@example.com) • **_email_from**—The address from which e-mail is sent (for example, devtest@example.com) • **_email_cc**—The address to which the e-mail must be copied (for example, manager@example.com)
Step 6	Program the body of the script.	In this section of the script, you can define any of the following: • The **event_reqinfo** event information Tcl command extension that is used to query the EEM for information about the detected event. • The action Tcl command extensions, such as **action_syslog**, that are used to specify actions specific to EEM. • The system information Tcl command extensions, such as **sys_reqinfo_routername**, that are used to obtain general system information.

	Command or Action	Purpose
		• 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).
Step 8	Check the exit status to determine whether or not to apply the default action for this event, if a default action exists.	A value of zero means that the default action should not be performed. A value of nonzero means that the default action should be performed. The exit status is passed to subsequent policies that are run for the same event.
Step 9	Set Cisco Error Number (_cerrno) Tcl global variables.	Some EEM Tcl command extensions set a Cisco Error Number Tcl global variable _cerrno. Whenever _cerrno is set, four other Tcl global variables are derived from _cerrno and are set along with it (_cerr_sub_num, _cerr_sub_err, , and _cerr_str).
Step 10	Save the Tcl script with a new filename, and copy the Tcl script to the router.	Embedded Event Manager policy filenames adhere to the following specification: • An optional prefix—Mandatory.—indicating, if present, that this is a system policy that should be registered automatically at boot time if it is not already registered. For example: Mandatory.sl_text.tcl. • A filename body part containing a two-character abbreviation (see Table 25: Two-Character Abbreviation Specification, on page 92) 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 92. Copy the file to the file system on the router—typically harddisk:/mirror/.
Step 11	**configure**	Enters global configuration mode.

	Command or Action	Purpose
Step 12	**event manager directory user** {**library** *path* | **policy** *path*} **Example:** `RP/0/RP0/CPU0:Router(config)# event manager directory user library harddisk:/mirror/EEM/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	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 15	Cause the policy to execute, and observe the policy.	—
Step 16	Use debugging techniques if the policy does not execute correctly.	—

Creating an EEM User Tcl Library Index

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

Procedure

	Command or Action	Purpose
Step 1	On your workstation (UNIX, Linux, PC, or Mac) create a library directory and copy the Tcl library files into the directory.	The following example files can be used to create a tclIndex on a workstation running the Tcl shell: **lib1.tcl** `proc test1 {} {`

	Command or Action	Purpose

		`  puts "In procedure test1"`
		`}`
		`proc test2 {} {`
		`  puts "In procedure test2"`
		`}`
		lib2.tcl
		`proc test3 {} {`
		`  puts "In procedure test3"`
		`}`

Step 2	**tclsh** **Example:** `workstation% tclsh`	Enters the Tcl shell.

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

Step 4	Copy the Tcl library files from step 1 and the tclIndex file from step 3 to the directory used for storing user library files on the target router.	—

	Command or Action	Purpose

Step 5	Copy a user-defined EEM policy file written in Tcl to the directory used for storing user-defined EEM policies on the target router.	The directory can be the same directory used in step 4.
		The following example user-defined EEM policy can be used to test the Tcl library support in EEM:
		libtest.tcl
		`::cisco::eem::event_register_none`
		`namespace import ::cisco::eem::*`
		`namespace import ::cisco::lib::*`
		`global auto_index auto_path`
		`puts [array names auto_index]`
		`if { [catch {test1} result]} {`
		`  puts "calling test1 failed result = $result`
		`$auto_path"`
		`}`
		`if { [catch {test2} result]} {`
		`  puts "calling test2 failed result = $result`
		`$auto_path"`
		`}`
		`if { [catch {test3} result]} {`
		`  puts "calling test3 failed result = $result`
		`$auto_path"`
		`}`

Step 6	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.

Step 7	**event manager directory user library** *path*
	Example:
	`RP/0/RP0/CPU0:Router(config)# event manager`
	`directory user library`
	`harddisk:/mirror/EEM/eem_library`
		Specifies the EEM user library directory; this is the directory to which the files in step 4 were copied.

Step 8	**event manager directory user policy** *path*
	Example:
	`RP/0/RP0/CPU0:Router(config)# event manager`
	`directory user policy`
	`harddisk:/mirror/EEM/eem_policies`
		Specifies the EEM user policy directory; this is the directory to which the file in step 5 was copied.

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

Step 10	**event manager run** *policy* [*argument*] **Example:**	Manually runs an EEM policy.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:Router(config)# event manager run libtest.tcl`	
Step 11	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Creating an EEM User Tcl Package Index

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

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

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

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

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

Procedure

	Command or Action	Purpose
Step 1	On your workstation (UNIX, Linux, PC, or Mac) create a library directory and copy the Tcl package files into the directory.	–
Step 2	**tclsh** **Example:** `workstation% tclsh`	Enters the Tcl shell.

	Command or Action	Purpose

Step 3	**pkg_mkindex** *directory_name* ***.tcl**
	Example:
	`workstation% pkg_mkindex eem_library *.tcl`
		Use the **pkg_mkindex** command to create the pkgIndex file. The pkgIndex file contains a directory of all the packages contained in the Tcl library files. We recommend that you run the **pkg_mkindex** command inside a directory, because there can be only a single pkgIndex file in any directory and you may have other Tcl files to be grouped together. Running the **pkg_mkindex** command in a directory determines which Tcl package file or files are indexed using a specific pkgIndex.
		The following example pkgIndex is created when some Tcl package files are in a library file directory and the pkg_mkindex command is run:
		pkgIndex
		`# Tcl package index file, version 1.1`
		`# This file is generated by the "pkg_mkIndex"`
		`command`
		`# and sourced either when an application starts`
		`up or`
		`# by a "package unknown" script. It invokes the`
		`# "package ifneeded" command to set up`
		`package-related`
		`# information so that packages will be loaded`
		`automatically`
		`# in response to "package require" commands. When`
		` this`
		`# script is sourced, the variable $dir must`
		`contain the`
		`# full path name of this file's directory.`
		`package ifneeded xmlrpc 0.3 [list source [file`
		`join $dir xmlrpc.tcl]]`

Step 4	Copy the Tcl package files from step 1 and the pkgIndex file from step 3 to the directory used for storing user library files on the target router.	—

Step 5	Copy a user-defined EEM policy file written in Tcl to the directory used for storing user-defined EEM policies on the target router.	The directory can be the same directory used in step 4.
		The following example user-defined EEM policy can be used to test the Tcl library support in EEM:
		packagetest.tcl
		`::cisco::eem::event_register_none maxrun`
		`1000000.000`
		`#`
		`# test if xmlrpc available`
		`#`
		`#`
		`# Namespace imports`
		`#`
		`namespace import ::cisco::eem::*`
		`namespace import ::cisco::lib::*`
		`#`

	Command or Action	Purpose

		`package require xmlrpc`
		`puts "Did you get an error?"`

Step 6	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 7	**event manager directory user library** *path* **Example:** `RP/0/RP0/CPU0:Router(config)# event manager directory user library harddisk:/mirror/EEM/eem_library`	Specifies the EEM user library directory; this is the directory to which the files in step 4 were copied.
Step 8	**event manager directory user policy** *path* **Example:** `RP/0/RP0/CPU0:Router(config)# event manager directory user policy harddisk:/mirror/EEM/eem_policies`	Specifies the EEM user policy directory; this is the directory to which the file in step 5 was copied.
Step 9	**event manager policy** *policy-name* **username** *username* [**persist-time** [*seconds* | **infinite**] | **type** [**system** | **user**]] **Example:** `RP/0/RP0/CPU0:Router(config)# event manager policy packagetest.tcl username user_a`	Registers a user-defined EEM policy.
Step 10	**event manager run** *policy* [*argument*] **Example:** `RP/0/RP0/CPU0:Router(config)# event manager run packagetest.tcl`	Manually runs an EEM policy.
Step 11	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

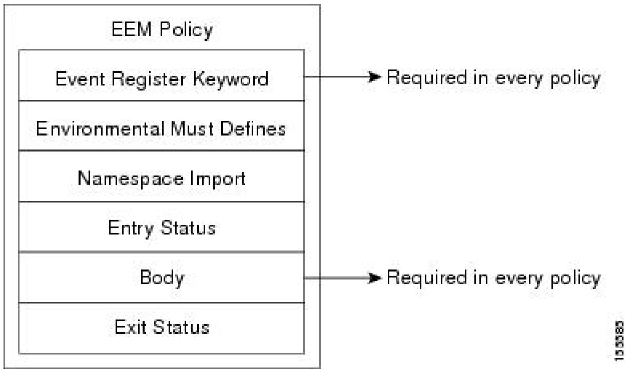
EEM Policies Using TCL: Details

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

Tcl Policy Structure and Requirements

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

Figure 3: Tcl Policy Structure and Requirements



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

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

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

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

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

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

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

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

EEM Entry Status

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

Entry status designations may use one of three possible values:

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

EEM Exit Status

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

EEM Policies and Cisco Error Number

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

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

```
XYSSSSSSSSSSSSSEEEEEEEEPPPPPPPP
```

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

Table 30: _cerrno: 32-Bit Error Return Value Variables

Variable	Description
XY	The error class (indicates the severity of the error). This variable corresponds to the first two bits in the 32-bit error return value; 10 in the preceding case, which indicates CERR_CLASS_WARNING: See #unique_129 unique_129_Connect_42_tab_1130225 for the four possible error class encodings specific to this variable.
SSSSSSSSSSSSS	The subsystem number that generated the most recent error(13 bits = 8192 values). This is the next 13 bits of the 32-bit sequence, and its integer value is contained in $_cerr_sub_num.
EEEEEEEE	The subsystem specific error number (8 bits = 256 values). This segment is the next 8 bits of the 32-bit sequence, and the string corresponding to this error number is contained in $_cerr_sub_err.

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

```
862439AE
```

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

```
10000110001001000011100110101110
```

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

Table 31: Error Class Encodings

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

An error return value of zero means SUCCESS.

Configuration Examples for Writing Embedded Event Manager Policies Using Tcl

EEM Sample Policy Descriptions

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

Registration of Some EEM Policies

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

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

Basic Configuration Details for All Sample Policies

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

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

Embedded Event Manager Policy Tcl Command Extension Reference

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

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

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

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

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

```
[type ?]
```

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

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

Embedded Event Manager Event Registration Tcl Command Extensions

The following EEM event registration Tcl command extensions are supported:

event_register_appl

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

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

Syntax

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

Arguments

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

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

Result String

None

Set _cerrno

No

event_register_cli

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

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

Syntax

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

Arguments

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

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

Result String

None

Set _cerrno

No

event_register_config

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

Syntax

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

Arguments

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

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

Result String

None

Set _cerrno

No

event_register_none

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

Syntax

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

Arguments

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

Result String

None

Set _cerrno

No

event_register_oir

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

Syntax

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

Arguments

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

Result String

None

Set _cerrno

No

event_register_process

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

Syntax

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

Arguments

abort	(Mandatory) Abnormal process termination. Process may terminate because of exiting with a nonzero exit status, receiving a kernel-generated signal, or receiving a SIGTERM or SIGKILL signal that is not sent because of user request.
term	(Mandatory) Normal process termination.
start	(Mandatory) Process start.
job_id	(Optional) Number assigned to the EEM policy that published the process event. Number is set to 798, because all other numbers are reserved for Cisco use.
instance	(Optional) Process instance ID. If specified, this argument must be an integer between 1 and 4294967295, inclusive.
path	(Optional) Process pathname (regular expression string).
node	(Optional) The node name is a string that consists of the word "node" followed by two fields separated by a slash (/), using the following format: node<slot-number>/<cpu-number> The slot-number is the hardware slot number. The cpu-number is the hardware CPU number. For example, the SP CPU in a Supervisor card on a Cisco Catalyst 6500 series switch located in slot 0 would be specified as node0/0. The RP CPU in a Supervisor card on a Cisco Catalyst 6500 series switch located in slot 0 would be addressed as node0/1. If the *node* argument is not specified, the default node specification is always the regular expression pattern match of * representing all applicable nodes.
queue_priority	(Optional) Priority level at which the script will be queued; normal priority is greater than low priority but less than high priority. The priority here is not execution priority, but queuing priority. If this argument is not specified, the default priority is normal.
maxrun	(Optional) Maximum run time of the script (specified in SSSSSSSSSS[.MMM] format, where SSSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the default 20-second run-time limit is used.

nice	(Optional) Policy run-time priority setting. When the *nice* argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.
tag	Tag is acceptable but ignored. Cisco IOS EEM scripts with the tag option can run in an Cisco IOS XR7 software environment without any error. Since Cisco IOS XR7 software does not support multiple events, the tag has no effect.

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

Result String

None

Set _cerrno

No

event_register_snmp_notification

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

Syntax

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

Argument

tag	(Optional) String identifying a tag that can be used with the trigger Tcl command extension to support multiple event statements within a Tcl script.
oid	(Mandatory) OID number of the data element in SNMP dot notation (for example, 1.3.6.1.2.1.2.1.0). If the specified OID ends with a dot (.), then all OIDs that start with the OID number before the dot are matched. It supports all OID supported by SNMP in XR.
oid_val	(Mandatory) OID value with which the current OID data value should be compared to decide if the SNMP event should be raised.
op	(Mandatory) Comparison operator used to compare the current OID data value with the SNMP Protocol Data Unit (PDU) OID data value; if this is true, an event is raised.

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

Result String

None

Set _cerrno

No

event_register_syslog

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

Syntax

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

Arguments

occurs	(Optional) Number of occurrences before the event is raised; if not specified, the event is raised on the first occurrence. If specified, the value must be greater than 0.

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

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

Table 32: Severity Level Mapping For Syslog Events

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

Result String

None

Set _cerrno

No

event_register_timer

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

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

Syntax

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

Arguments

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

cron_entry	(Optional) Entry must be specified if the CRON timer type is specified. Must not be specified if any other timer type is specified. A cron_entry is a partial UNIX crontab entry (the first five fields) as used with the UNIX CRON daemon. A cron_entry specification consists of a text string with five fields. The fields are separated by spaces. The fields represent the time and date when CRON timer events will be triggered. The fields are described in Table 33: Time and Date When CRON Events Will Be Triggered , on page 129. Ranges of numbers are allowed. Ranges are two numbers separated with a hyphen. The specified range is inclusive. For example, 8-11 for an hour entry specifies execution at hours 8, 9, 10, and 11. A field may be an asterisk (*), which always stands for "first-last." Lists are allowed. A list is a set of numbers (or ranges) separated by commas. Examples: "1,2,5,9" and "0-4,8-12". Step values can be used in conjunction with ranges. Following a range with "/<number>" specifies skips of the number's value through the range. For example, "0-23/2" is used in the hour field to specify an event that is triggered every other hour. Steps are also permitted after an asterisk, so if you want to say "every two hours", use "*/2". Names can also be used for the month and the day of week fields. Use the first three letters of the particular day or month (case does not matter). Ranges or lists of names are not allowed. The day on which a timer event is triggered can be specified by two fields: day of month and day of week. If both fields are restricted (that is, are not *), an event will be triggered when either field matches the current time. For example, "30 4 1,15 * 5" would cause an event to be triggered at 4:30 a.m. on the 1st and 15th of each month, plus every Friday. Instead of the first five fields, one of seven special strings may appear. These seven special strings are described in Table 34: Special Strings for cron_entry, on page 129 Example 1: "0 0 1,15 * 1" would trigger an event at midnight on the 1st and 15th of each month, as well as on every Monday. To specify days by only one field, the other field should be set to *; "0 0 * * 1" would trigger an event at midnight only on Mondays. Example 2: "15 16 1 * *" would trigger an event at 4:15 p.m. on the first day of each month. Example 3: "0 12 * * 1-5" would trigger an event at noon on Monday through Friday of each week. Example 4: "@weekly" would trigger an event at midnight once a week on Sunday.
time	(Optional) Time must be specified if a timer type other than CRON is specified. Must not be specified if the CRON timer type is specified. For watchdog and countdown timers, the number of seconds and milliseconds until the timer expires; for the absolute timer, the calendar time of the expiration time. Time is specified in SSSSSSSSSS[.MMM] format, where SSSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999. An absolute expiration date is the number of seconds and milliseconds since January 1, 1970. If the date specified has already passed, the timer expires immediately.
queue_priority	(Optional) Priority level at which the script will be queued; normal priority is greater than low priority but less than high priority. The priority here is not execution priority, but queuing priority. If this argument is not specified, the default priority is normal.

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

Table 33: Time and Date When CRON Events Will Be Triggered

Field	Allowed Values
minute	0-59
hour	0-23
day of month	1-31
month	1-12 (or names, see Table 34: Special Strings for cron_entry, on page 129)
day of week	0-7 (0 or 7 is Sun, or names; see Table 34: Special Strings for cron_entry, on page 129)

Table 34: Special Strings for cron_entry

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

Result String

None

Set _cerrno

No

See Also

#unique_145

event_register_timer_subscriber

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

Syntax

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

Arguments

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

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

Result String

None

Set _cerrno

No

See Also

event_register_timer, on page 127

event_register_track

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

Syntax

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

Arguments

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

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

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

Result String

None

Set _cerrno

No

Embedded Event Manager Event Information Tcl Command Extension

The following EEM Event Information Tcl Command Extensions are supported:

event_reqinfo

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

Syntax

```
event_reqinfo
```

Arguments

None

Result String

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

For EEM_EVENT_APPLICATION

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

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

Event Type	Description
sub_system	Number assigned to the EEM policy that published the application event. Number is set to 798 because all other numbers are reserved for Cisco use.
type	Event subtype within the specified component.
data1data2data3data4	Argument data that is passed to the application-specific event when the event is published. The data is character text, an environment variable, or a combination of the two.

For EEM_EVENT_COUNTER

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

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

For EEM_EVENT_NONE

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

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

For EEM_EVENT_OIR

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

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

For EEM_EVENT_PROCESS (Software Modularity Only)

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

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

Event Type	Description
exit_status	Process last exit status.
respawn_count	Number of times that the process was restarted.
last_respawn_seclast_respawn_msec	Calendar time when the last restart occurred.
fail_count	Number of restart attempts of the process that failed. This count will be reset to 0 when the process is successfully restarted.
Event Type	**Description**
dump_count	Number of core dumps taken of the process.
node_name	Name of the node that the process is on. The node name is a string that consists of the word "node" followed by two fields separated by a slash character using the following format: node<slot-number>/<cpu-number> The slot-number is the hardware slot number. The cpu-number is the hardware CPU number.

For EEM_EVENT_RF

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

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

For EEM_EVENT_SYSLOG_MSG

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

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

For EEM_EVENT_TIMER_ABSOLUTE

EEM_EVENT_TIMER_COUNTDOWN

EEM_EVENT_TIMER_WATCHDOG

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

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

For EEM_EVENT_TIMER_CRON

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

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

For EEM_EVENT_TRACK

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

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

For EEM_EVENT_WDSYSMON

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

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

Where the subevent info string is for a deadlock subevent:

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

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

Where each entry is:

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

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

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

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

For dispatch_mgr Subevent

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

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

For cpu_proc Subevent

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

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

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

For cpu_tot Subevent

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

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

For mem_proc Subevent

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

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

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

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

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

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

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

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

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

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

- *value* is 0.

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

For mem_tot_avail Subevent

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

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

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

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

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

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

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

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

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

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

For mem_tot_used Subevent

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

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

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

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

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

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

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

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

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

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

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

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

Set _cerrno

Yes

Embedded Event Manager Action Tcl Command Extensions

action_process

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

Syntax

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

Arguments

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

Result String

None

Set _cerrno

Yes

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

This error means that the action command requested was unknown.

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

This error means that the arguments passed in were invalid.

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

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

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

This error means that the process name was not known.

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

This error means that an object size exceeded its maximum.

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

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

action_program

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

Syntax

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

Arguments

path	(Mandatory) Pathname of a program to run.

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

Result String

None

Set _cerrno

Yes

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

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

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

This error means that the action command requested was unknown.

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

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

action_script

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

Syntax

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

Arguments

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

Result String

None

Set _cerrno

Yes

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

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

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

This error means that the action command requested was unknown.

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

This error means that a configuration error has occurred.

action_setnode

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

- action_process
- sys_reqinfo_proc
- sys_reqinfo_proc_all
- sys_reqinfo_crash_history
- sys_reqinfo_proc_version

Syntax

```
action_setnode [node ?]
```

Arguments

node	(Mandatory) Name of the node.

Result String

None

Set _cerrno

Yes

action_syslog

Logs a message.

Syntax

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

Arguments

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

Result String

None

Set _cerrno

Yes

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

This error means that the action command requested was unknown.

Embedded Event Manager Utility Tcl Command Extensions

appl_read

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

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

Syntax

```
appl_read name ? length ?
```

Arguments

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

Result String

```
data %s
```

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

Set _cerrno

Yes

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

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

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

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

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

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

appl_reqinfo

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

Syntax

```
appl_reqinfo key ?
```

Arguments

key	(Mandatory) String key of the data.

Result String

```
data %s
```

Where data is the application string data to be retrieved.

Set _cerrno

Yes

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

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

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

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

appl_setinfo

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

Syntax

```
appl_setinfo key ? data ?
```

Arguments

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

Result String

None

Set _cerrno

Yes

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

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

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

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

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

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

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

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

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

This error means that the API message length was invalid.

counter_modify

Modifies a counter value.

Syntax

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

Arguments

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

Result String

```
val_remain %d
```

Where val_remain is the current value of the counter.

Set _cerrno

Yes

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

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

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

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

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

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

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

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

timer_arm

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

Syntax

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

Arguments

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

Result String

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

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

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

Set _cerrno

Yes

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

This error means that the event was not registered.

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

This error means that the event detector was unavailable.

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

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

timer_cancel

Cancels a timer.

Syntax

```
timer_cancel event_id ?
```

Arguments

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

Result String

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

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

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

Set _cerrno

Yes

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

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

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

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

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

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

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

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

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

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

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

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

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

This error means that the event detector was unavailable.

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

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

Embedded Event Manager System Information Tcl Command Extensions

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

sys_reqinfo_cpu_all

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

Syntax

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

Arguments

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

Result String

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

Where each process CPU info string is:

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

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

Set _cerrno

Yes

sys_reqinfo_crash_history

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

Syntax

```
sys_reqinfo_crash_history
```

Arguments

None

Result String

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

Where each crash info string is:

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

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

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

Set _cerrno

Yes

sys_reqinfo_mem_all

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

Syntax

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

Arguments

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

Result String

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

Where each process mem info string is:

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

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

Set _cerrno

Yes

sys_reqinfo_proc

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

Syntax

```
sys_reqinfo_proc job_id ?
```

Arguments

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

Result String

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

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

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

Set _cerrno

Yes

sys_reqinfo_proc_all

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

Syntax

```
sys_reqinfo_proc_all
```

Arguments

None

Result String

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

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

Set _cerrno

Yes

sys_reqinfo_proc_version

Queries the version of the given process.

Syntax

```
sys_reqinfo_proc_version [job_id ?]
```

Arguments

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

Result String

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

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

Set _cerrno

Yes

sys_reqinfo_routername

Queries the router name.

Syntax

```
sys_reqinfo_routername
```

Arguments

None

Result String

```
routername %s
```

Where routername is the name of the router.

Set _cerrno

Yes

sys_reqinfo_syslog_freq

Queries the frequency information of all syslog events.

Syntax

```
sys_reqinfo_syslog_freq
```

Arguments

None

Result String

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

Where each event frequency string is:

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

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

pattern	Regular expression used to perform syslog message pattern matching.

Set _cerrno

Yes

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

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

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

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

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

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

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

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

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

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

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

This error means that the event detector was unavailable.

sys_reqinfo_syslog_history

Queries the history of the specified syslog message.

Syntax

```
sys_reqinfo_syslog_history
```

Arguments

None

Result String

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

Where each log hist string is:

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

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

Set _cerrno

Yes

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

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

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

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

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

This error means that the history list was empty.

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

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

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

This error means that the event detector was unavailable.

sys_reqinfo_stat

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

Syntax

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

Arguments

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

mod_2	(Optional) Statistics data element modifier 2.

Result String

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

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

Set _cerrno

Yes

sys_reqinfo_snmp

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

Syntax

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

Arguments

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

Result String

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

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

SMTP Library Command Extensions

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

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

E-Mail Template

The e-mail template file has the following format:

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

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

The following is a sample e-mail template file:

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

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

Exported Tcl Command Extensions

smtp_send_email

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

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

Syntax

```
smtp_send_email text
```

Arguments

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

Result String

None

Set _cerrno

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

Sample Scripts

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

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

smtp_subst

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

Syntax

```
smtp_subst e-mail_template
```

Arguments

e-mail_template	(Mandatory) Name of an e-mail template file in which global variables need to be substituted by a user-defined value. An example filename could be harddisk:/mirror/example.template which represents a file named example.template in mirror directory on the harddisk.

Result String

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

Set _cerrno

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

CLI Library Command Extensions

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

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

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

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

For noninteractive commands, once the command is entered, the command will run to completion. To run different types of commands using an EEM script, different CLI library command sequences should be used, which are documented in the Using the CLI Library to Run a Noninteractive Command, on page 174 and in the Using the CLI Library to Run an Interactive Command, on page 175.

Exported Tcl Command Extensions

cli_close

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

Syntax

```
cli_close fd tty_id
```

Arguments

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

Result String

None

Set _cerrno

Cannot close the channel.

cli_exec

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

Syntax

```
cli_exec fd cmd
```

Arguments

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

Result String

The output of the CLI command executed.

Set _cerrno

Error reading the channel.

cli_get_ttyname

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

Syntax

```
cli_get_ttyname tty_id
```

Arguments

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

Result String

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

Set _cerrno

None

cli_open

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

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

Syntax

```
cli_open
```

Arguments

None

Result String

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

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

Set _cerrno

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

cli_read

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

Syntax

```
cli_read fd
```

Arguments

fd	(Mandatory) CLI channel handler.

Result String

All the contents read.

Set _cerrno

Cannot get router name.

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

cli_read_drain

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

Syntax

```
cli_read_drain fd
```

Arguments

fd	(Mandatory) The CLI channel handler.

Result String

All the contents read.

Set _cerrno

None

cli_read_line

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

Syntax

```
cli_read_line fd
```

Arguments

fd	(Mandatory) CLI channel handler.

Result String

The line read.

Set _cerrno

None

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

cli_read_pattern

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

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

Syntax

```
cli_read_pattern fd ptn
```

Arguments

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

Result String

All the contents read.

Set _cerrno

None

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

cli_write

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

Syntax

```
cli_write fd cmd
```

Arguments

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

Result String

None

Set _cerrno

None

Sample Usage

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

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

Using the CLI Library to Run a Noninteractive Command

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

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

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

Using the CLI Library to Run an Interactive Command

To run interactive commands, three phases are needed:

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

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

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

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

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

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

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

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

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

Tcl Context Library Command Extensions

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

Exported Commands

context_retrieve

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

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

Syntax

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

Arguments

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

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

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

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

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

Result String

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

Set _cerrno

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

Sample Usage

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

Example 1: Save

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

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

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

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

Example 1: Retrieve

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

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

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

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

Example 2: Save

If var is specified, saves the value of var.

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

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

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

Example 2: Retrieve

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

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

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

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

Example 3: Save

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

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

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

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

Example 3: Retrieve

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

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

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

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

Example 4: Save

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

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

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

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

Example 4: Retrieve

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

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

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

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

context_save

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

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

Syntax

```
context_save ctxt [pattern]
```

Arguments

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

Result String

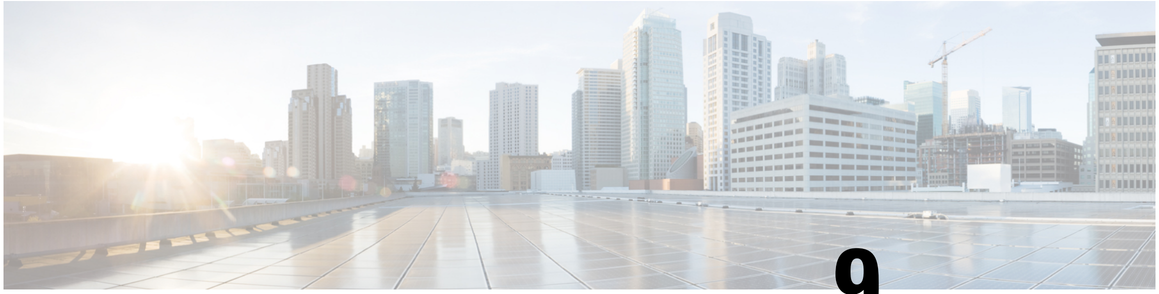
None

Set _cerrno

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

Sample Usage

For examples showing how to use the **context_save** and **context_retrieve** command extension functionality to save and retrieve data, see the Sample Usage, on page 178.



CHAPTER 9

Graceful Handling of Out of Resource Situations

Table 35: Feature History Table

Feature Name	Release Information	Description
Extending Graceful Handling of Out of Resource (OOR) Situations for GRE and MPLS traffic	Release 25.4.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100])(select variants only*) *This feature is supported on Cisco 8711-48Z-M routers.
Graceful Handling of Out of Resource (OOR) Situations	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on Cisco 8011-4G24Y4H-I routers.
Graceful Handling of Out of Resource (OOR) Situations	Release 24.4.1	Introduced in this release on: Fixed Systems (8200, 8700 [LC ASIC: P100, K100]); Modular Systems (8800 [LC ASIC: P100]) (select variants only*) *This feature is now supported on: • 8212-48FH-M • 8711-32FH-M • Cisco 8712-MOD-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

Feature Name	Release Information	Description
Graceful Handling of Out of Resource (OOR) Situations for SR-TE traffic	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on Cisco 8011-4G24Y4H-I routers.
Graceful Handling of Out of Resource (OOR) Situations for SR-TE traffic	Release 24.4.1	Introduced in this release on: Fixed Systems (8200, 8700 [LC ASIC: P100, K100]); Modular Systems (8800 [LC ASIC: P100]) (select variants only*) * This release extends the graceful handling of dropped traffic in an out-of-resource situation to SR-TE traffic to: • 8212-48FH-M • 8711-32FH-M • Cisco 8712-MOD-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
Extending Graceful Handling of Out of Resource (OOR) Situations for SR-TE traffic	Release 7.3.3	This release extends the graceful handling of dropped traffic in an out-of-resource situation to SR-TE traffic. Modified command: • show controllers npu resources
Extending Graceful Handling of Out of Resource (OOR) Situations for GRE and MPLS traffic	Release 7.3.2	This release extends the graceful handling of dropped traffic in an out-of-resource situation to GRE and MPLS-enabled traffic. Modified command: • show controllers npu resources

Feature Name	Release Information	Description
Graceful Handling of Out of Resource (OOR) Situations	Release 7.3.1	This feature enables you to resend any traffic that was dropped during an OOR situation. This enables better monitoring and management of failed traffic. Commands introduced and modified are: • oor hw • show controllers npu resources

Graceful Handling of Out of Resource Situations Overview

OOR situations occur when the network is unable to handle the overload of traffic. It can lead to traffic loss. Graceful handling of OOR situations denotes the router recovers without any traffic loss of unaffected traffic. The recovery of unaffected traffic occurs when the OOR situation is cleared.

When a router reaches the OOR state, you release traffic with few prefixes to reduce the utilization of hardware and SDK resources. You can release traffic with the help of a traffic generator. With the reduced utilization of hardware and SDK resources, the router comes out of the OOR state. After the router is out of the OOR state, you can reinject the traffic that you had released. You can reinject the traffic with the help of traffic generator in a favorable way. You can control the monitoring and resending of failed traffic and gracefully handle OOR situations.

The **OOR State** in the output of the **show controllers npu resources** command changes when the router reaches an OOR situation due to heavy traffic or extreme utilization of hardware and SDK resources. The **OOR State** changes from **Green** to **Yellow**, and finally to **Red**. When the **OOR State** reaches **Red**, the Syslog in the router generates a notification and sends it to the end user.

The different **OOR State** signifies the following:

- **Green**: Favorable utilization of hardware and SDK resources
- **Yellow**: Router is advancing toward the OOR state
- **Red**: Router has reached the OOR state

You can configure the threshold value at which a router reaches the **OOR Red** or **Yellow** states by using the **oor hw** command.

The default values for OOR states are as follows:

- The **Yellow** state occurs when 80% of the router's hardware and SDK resources are utilized.
- The **Red** state occurs when 95% of the router's hardware and SDK resources are utilized.

For more information, see **oor hw** command in the chapter *Graceful Handling of OOR Situations Commands* of *System Monitoring Command Reference for Cisco 8000 Series Routers*.

For FIB hardware programming failure recovery enhancements in non-OOR scenarios, see Enhancing FIB hardware programming failure recovery in non-OOR scenarios, on page 313.

You can use the **show controllers npu resources** command to view the status of utilization of hardware and Software Development Kit (SDK) resources:

Table 36: NPU Resources per Traffic Type

Traffic Type	NPU Resource
IPv4/IPv6	• lpmtcam • centralem • stage1lbgroup • stage1lbmember • stage2lbgroup • stage2lbmember
MPLS	• egresslargeencap • centralem
GRE	• egresslargeencap • sipidxtbl • myipv4tbl • tunneltermination
SR-TE	• counterbank • egresslargeencap • egresssmallencap • stage1lbgroup • stage1lbmember • stage2lbgroup • stage2lbmember

Restrictions

Graceful handling of OOR situations is only supported for IPv4, IPv6, MPLS, SR-TE, and GRE traffic.

Configuration

To configure OOR limits, use the **oor hw** command.

```
Router(config)#oor hw threshold red 96
Router(config)#oor hw threshold yellow 85
Router(config)#commit
```

Verification

To verify the OOR state of a router, use the **show logging | inc OOR** command.

```
Router# show logging | inc OOR
Wed Jan 6 23:36:34.138 EST
LC/0/0/CPU0:Jan 6 23:01:09.609 EST: npu_drvr[278]: %PLATFORM-OFA-4-OOR_YELLOW : NPU 1, Table
 nhgroup, Resource stage2_lb_group
LC/0/0/CPU0:Jan 6 23:01:29.655 EST: npu_drvr[278]: %PLATFORM-OFA-4-OOR_YELLOW : NPU 1, Table
 nhgroup, Resource stage2_lb_member
LC/0/0/CPU0:Jan 6 23:01:38.938 EST: npu_drvr[278]: %PLATFORM-OFA-1-OOR_RED : NPU 3, Table
nhgroup, Resource stage2_lb_group
```

To verify the NPU resource utilization for GRE traffic of a router, use the **show controllers npu resources** command.

```
Router# show controllers npu resources lpmtcam location 0/0/CPU0
Thu Dec 17 11:43:06.931 EST
HW Resource Information
    Name                            : lpm_tcam
    Asic Type                       : Pacific

NPU-0
OOR Summary
        Estimated Max Entries    : 100
        Red Threshold            : 95 % >>> shows the threshold for OOR Status Red
        Yellow Threshold         : 80 % >>> shows the threshold for OOR Status Yellow
        OOR State                : Red >>> shows that the OOR status is Red
        OOR State Change Time    : 2020.Dec.17 09:53:02 EST >>> shows the time at which OOR
 status changed to Red
```

Note The IP BGP ECMP over BVI uses *stage2lbgroup* and *stage2lbmember* NPU resources. You can use the following commands to monitor the total in-use values for resource utilization. The *nhgroup* value in the command outputs does not mean the hardware resource usage value. Please refer to the *Total In-Use* value to get the current hardware resource usage.

- **show controllers npu resources stage2lbgroup location** <*location*>

  For example,

  ```
  Router# show controllers npu resources stage2lbgroup location location all
  ```

- **show controllers npu resources stage2lbmember location** <*location*>

  For example,

  ```
  Router# show controllers npu resources stage2lbmember location location all
  ```

Graceful Handling of Out of Resource for Protection Group

Table 37: Feature History Table

Feature Name	Release Information	Description
Graceful Handling of Out of Resource (OOR) for Protection Group	Release 25.4.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100])(select variants only*) *This feature is supported on Cisco 8711-48Z-M routers.
Graceful Handling of Out of Resource (OOR) for Protection Group	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on Cisco 8011-4G24Y4H-I routers.
Graceful Handling of Out of Resource (OOR) for Protection Group	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100, K100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • Cisco 8712-MOD-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

Feature Name	Release Information	Description
Graceful Handling of Out of Resource (OOR) for Protection Group	Release 7.5.4	This feature adds a new feedback path from the Hardware Abstraction Layer to Forwarding Information Base, to enable improved Graceful Recovery. This feature helps the system recover by re-programming routes when hardware resources become available. The following commands are modified: • show cef object queue location • show ofa transport async stats client fib • show controllers npu resources

Data structure memories in the NPU are limited, and can reach OOR situations either due to network state changes or user configuration change.

With Cisco IOS XR Release 7.5.4, you can improve the network state change-related OOR handling, which is distributed by control protocols. A new feedback path between Open Forwarding Abstraction (OFA) and Forwarding Information Base (FIB) provides for Graceful Recovery.

OFA or SDK async errors and OOR failures are now reported back to FIB. The FIB is a subsystem responsible for updating NPU on the L3 network state, such as routes and adjecencies. This process is goverened by the **fib_mgr** utility in XR, on the router.

The following states summarize the stages within OOR for Protection Groups:

- **Accounting** - Provides a snapshot of the hardware resource utilization. Accounting happens in the SDK, and the Open Forwarding Abstraction provides hooks to read the usage level for CLI or telemetry.
- **Monitoring** - Checks and tracks the utlization based on the following threshold levels:
    - **Green**: Resource utilization is normal and within limit. No action from the user is required.
    - **Yellow**: Resource utilization has reached a warning point where users are made aware.
    - **Red**: Resource utilization has reached a critical point where traffic loss could occur.
- **Reporting** - Generates syslogs when OOR crosses the threshold.

  During changes in the threshold level, OFA will generate a syslog and these can be accessed using event-driven telemetry. Syslogs are generated to notify or alert users about the OOR state change for each resource table. The syslog messages are generated when resource utilization level crosses either above or below a threshold. For example, a Yellow notification is issued when the resource utlization has crossed the threshold warning point. A Green notification is generated when the threshold level reverts to below the resource utilization limit.

```
LC/0/0/CPU0:Dec 16 15:53:20.692 EST: npu_drvr[212]: %PLATFORM-OFA-4-OOR_YELLOW : NPU
```

```
2, Table ip6rte, Resource central_em
LC/0/0/CPU0:Dec 16 15:53:36.586 EST: npu_drvr[212]: %PLATFORM-OFA-1-OOR_RED : NPU 1,
Table ip6rte, Resource central_em
LC/0/0/CPU0:Dec 16 16:37:15.688 EST: npu_drvr[212]: %PLATFORM-OFA-4-OOR_YELLOW : NPU
2, Table iprte, Resource lpm_tcam
LC/0/0/CPU0:Dec 16 16:37:16.186 EST: npu_drvr[212]: %PLATFORM-OFA-5-OOR_GREEN : NPU 3,
 Table iprte, Resource lpm_tcam
LC/0/0/CPU0:Dec 16 16:37:17.273 EST: npu_drvr[212]: %PLATFORM-OFA-4-OOR_YELLOW : NPU
2, Table iprte, Resource lpm_tcam
LC/0/0/CPU0:Dec 16 16:37:19.760 EST: npu_drvr[212]: %PLATFORM-OFA-5-OOR_GREEN : NPU 0,
 Table iprte, Resource lpm_tcam
```

The following is an example for an OOR Protection Group error syslog :

```
RP/0/RP0/CPU0:Aug 12 23:26:50.286 UTC: npu_drvr[361]: %ROUTING-FIB-4-PROTGRP_OOR :
Protected Path programming failure:  Protection Group OOR trans_id 280775408 NPU 0
nhgroup 2491381947832520
```

- The following is an example for an OOR Protection Group alarm syslog:

```
npu_drvr[389]: %PLATFORM-OFA-4-OOR_YELLOW : NPU 0, Table nhgroup, Resource
protection_group
npu_drvr[389]: %PLATFORM-OFA-1-OOR_RED : NPU 0, Table nhgroup, Resource protection_group
%PLATFORM-OFA-5-OOR_GREEN : NPU 0, Table nhgroup, Resource protection_group
```

- **Graceful-Recovery** - Tracks the routes, labels, and encapsulations that are not programmed due to OOR. The **fib_mgr** (producer) of routes tracks and lists the routes that are in OOR Yellow or Red state, and retries when resource availability changes to Green state.

Configuring Graceful Handling of OOR for Protection Group

For information on configuring the OOR for Protection Groups, see the configuration steps in the Graceful Handling of Out of Resource Situations Overview section. The default values are Yellow (80%) and Red (95%). The initial state is Green.

For information on the entries that are queued in the FIB OOR retry queue based on the object queue ID, see the show cef object-queue location command.

For information on the async response error stats that are sent through the out-of-band async channel from OFA npu_drvr to FIM Mgr, you can use the show ofa transport async stats client fib command.

System Log Alerts for Packet Loss

Table 38: Feature History Table

Feature Name	Release Information	Description
System Log Alerts for Packet Loss	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I
System Log Alerts for Packet Loss	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

Feature Name	Release Information	Description
System Log Alerts for Packet Loss	Release 24.1.1	You can quickly get notified about any traffic impacting errors within the router's Network Processing Unit (NPU). These notifications are error log messages on the router console for NPU interrupts that affect traffic. To diagnose traffic loss, follow the recommended actions in the log. Previously, the only way to identify NPU errors that impacted traffic was to run the **show asic-error** command. This feature introduces the following changes: **CLI:** This feature introduces the **hw-module profile packet-loss-alert** command. **YANG Model:** New xpaths for `Cisco-IOS-XR-npu-hw-profile-cfg.yang` data model (see GitHub, YANG Data Models Navigator)

Network packet loss can significantly impact the overall experience of end-users, particularly when using real-time applications such as voice or video. Therefore, network administrators need to address such issues promptly to ensure the integrity of these services.

Now you can configure your router to provide immediate alerts in the event of packet loss, along with configuring the duration of packet loss to raise the alert. This feature enables network administrators to quickly identify the specific router within the network that is experiencing the problem. It facilitates a swift and precise response to rectify the issue and maintain optimal network performance.

Note Only line cards and routers with the Q100, Q200, P100, or G100 based Silicon One ASIC support this feature.

Configuration Example

Execute the **hw-module profile packet-loss-alert** command to enable this feature, as shown below:

```
Router# configure
Router(config)# hw-module profile packet-loss-alert 3Min
Router(config)# commit
```

You can configure the duration of packet loss to raise the alert to either 3 minutes or 5 minutes.

Running Configuration

```
Router# show running-config hw-module profile
hw-module profile packet-loss-alert 3Min
```

System Log Alert Generated for Packet Loss

When you enable this feature, the router generates a system log message whenever there’s a packet loss for the configured duration. To diagnose the reason for the packet loss, follow the recommended action in the log.

```
LC/0/3/CPU0:Nov 4 21:12:47.062 UTC: npu_drvr[213]:
%FABRIC-NPU_DRVR-3-ASIC_ERROR_TRAFFIC_IMPACT : [10118] : npu[2]: Potential PACKET_LOSS due
 to error, please check configuration to see if drop is expected; if not, collect showtech
 fabric link-include and follow the TAC guideline for this message
```

CHAPTER 10

Implementing IP Service Level Agreements

Table 39: Feature History Table

Feature Name	Release Information	Description
IP Service Level Agreement	Release 25.4.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100]) (select variants only*) *This feature is supported on: • 8711-48Z-M • 8011-12G12X4Y-A • 8011-12G12X4Y-D
IP Service Level Agreement	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on Cisco 8011-4G24Y4H-I routers.

Feature Name	Release Information	Description
IP Service Level Agreement	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100, K100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • Cisco 8712-MOD-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
IP Service Level Agreement	Release 7.3.2	This feature allows you to actively monitor, measure, and report traffic information continuously across the network. You can configure the router to measure and report jitter, response time, packet loss, QoS thresholds, connectivity, response or downtime, and delays.

This chapter covers the following topics:

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

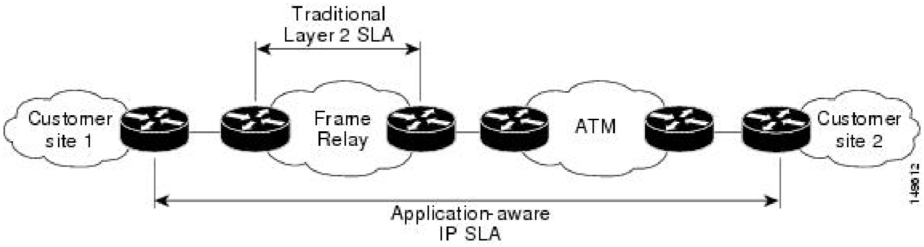
Internet commerce has grown significantly in the past few years as the technology has advanced to provide faster, more reliable access to the Internet. Many companies need online access and conduct most of their business on line and any loss of service can affect the profitability of the company. Internet 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. Scope of Traditional Service Level Agreement Versus IP SLA 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 apllication 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



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

Table 40: IP SLA Improvements over a Traditional Service Level Agreement

Type of Improvement	Description
End-to-end measurements	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.
Sophistication	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.
Accuracy	Applications that are sensitive to slight changes in network performance require the precision of the submillisecond measurement of IP SLA.
Ease of deployment	Leveraging the existing Cisco devices in a large network makes IP SLA easier to implement than the physical operations that are often required with traditional service level agreements.
Application-aware monitoring	IP SLA can simulate and measure performance statistics generated by applications running over Layer 3 through Layer 7. Traditional service level agreements can measure only Layer 2 performance.
Pervasiveness	IP SLA support exists in Cisco networking devices ranging from low-end to high-end routers and switches. This wide range of deployment gives IP SLA more flexibility over traditional service level agreements.

Benefits of IP Service Level Agreements

This table lists the benefits of implementing IP SLA.

Table 41: List of Benefits for IP SLA

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

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 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.
- The current validated scale numbers for scheduling UDP jitter operations is 100 operations with default frequency.
- 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.

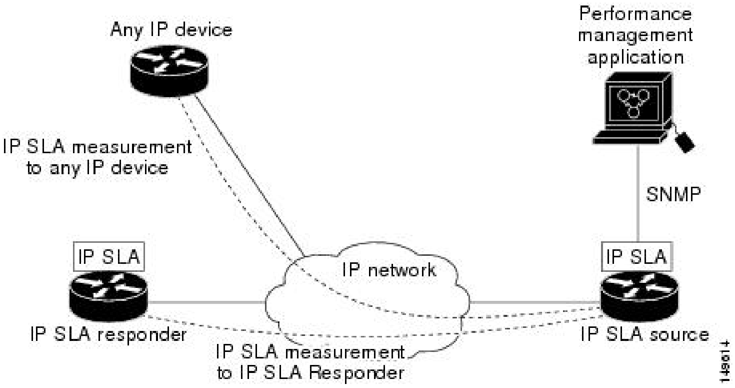
> **Note** Setting the frequency to less than 60 seconds will increase the number of packets sent. But this could negatively impact the performance of IP SLA operation when scheduled operations have same start time.

- IP SLA does not support High Availability (HA). If a HA event occurs on a controller using IP SLA with a TCP connection, you must restart the connection to the responder or server because the connection closes automatically.
- Consider the following guidelines before configuring the frequency, timeout, and threshold commands.
  - For the UDP jitter operation, the following guidelines are recommended:
    - frequency > timeout + 2 seconds + num_packets * packet_interval
    - timeout > rtt_threshold
    - num_packet > loss_threshold
- Control disabled mode gives a better IP-SLA scale when compared to Control Enabled mode.
- UDP ports for IP SLA responder can range from 1024 to 15000.
- You can configure only one ICMP path-echo operation if you use maximum number of parameters such as bucket hours, hop count, path, distribution count, and so on.

Measuring Network Performance with IP Service Level Agreements

IP SLA uses generated traffic to measure network performance between two networking devices, such as routers. IP SLA Operations 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


Any IP device
Performance management application
IP SLA measurement to any IP device
SNMP
IP SLA
IP network
IP SLA
IP SLA responder
IP SLA source
IP SLA measurement to IP SLA Responder
149614

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.

IP SLA Operations 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.

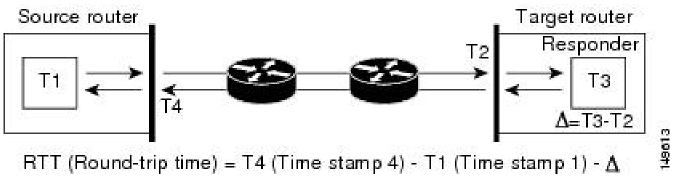
Response Time Computation for IP SLA

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.

IP SLA Responder Time Stamping 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.

Figure 6: IP SLA Responder Time Stamping


Source router
Target router
Responder
T1
T2
T3
T4
Δ=T3-T2
RTT (Round-trip time) = T4 (Time stamp 4) - T1 (Time stamp 1) - Δ
148613

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.

Operation Types for IP Service Level Agreements

IP SLA configures various types of operations to measure response times, jitter, throughput, and packet loss. Also, each operation maps to multiple applications.

This table lists the various types of operations.

Table 42: Types of Operations for IP SLA

Operation	Description
UDP echo	Measures round-trip delay and helps in accurate measurement of response time of UDP traffic.
UDP jitter	Measures round-trip delay, one-way delay, one-way jitter, two-way jitter, and one-way packet loss.
ICMP echo	Measures round-trip delay for the full path.
ICMP path-echo	Calculates the hop-by-hop response time between the router and any IP device on the network. The path is discovered using the traceroute algorithm and then by measuring the response time between the source router and each intermediate hop in the path. If there are multiple equal-cost routes between source and destination devices, the ICMP path-echo operation can select one of the paths by using the Loose Source Routing (LSR) option, which is configurable.
ICMP path-jitter	Measures hop-by-hop jitter, packet loss, and delay measurement statistics in an IP network.
MPLS LSP ping	Tests the connectivity of a label switched paths (LSP) and measures round-trip delay of the LSP in an MPLS network. The following Forwarding Equivalence Classes (FECs) are supported: • IPv4 Label Distribution Protocol (LDP) • Traffic engineering (TE) tunnels • Pseudowire An echo request is sent along the same data path as other packets belonging to the FEC. When the echo request packet reaches the end of the path, it is sent to to the control plane of the egress label switching router (LSR). The LSR verifies that it is indeed an egress for the FEC and sends an echo reply packet that contains information about the FEC whose MPLS path is being verified. Only a default VRF table is supported.
MPLS LSP trace	Traces the hop-by-hop route of an LSP path and measures the hop-by-hop round-trip delay for IPv4 LDP prefixes and TE tunnel FECs in an MPLS network. An echo request packet is sent data to the control plane of each transit LSR, which checks if it is a transit LSR for this path. Each transit LSR also returns information related to the label bound to the FEC that is being tested. Only a default VRF table is supported.

IP SLA VRF Support

Service providers need to monitor and measure network performance from both the perspective of the core network and a customer’s network. To do so, it is necessary to use nondefault VPN routing and forwarding (VRF) tables for IP SLA operations in addition to the default VRF table. IP SLA VRF Support table describes the different IP SLA operations, including information about whether or not an operation supports the use of nondefault VRF tables.

IP SLA—Proactive Threshold Monitoring

This section describes the proactive monitoring capabilities for IP SLA that use thresholds and reaction triggering. IP SLA allows you to monitor, analyze, and verify IP service levels for IP applications and services to increase productivity, lower operational costs, and reduce occurrences of network congestion or outages. IP SLA uses active traffic monitoring to measure network performance.

To perform the tasks that are required to configure proactive threshold monitoring using IP SLA, you must understand these concepts:

IP SLA Reaction Configuration

IP SLA is configured to react to certain measured network conditions. For example, if IP SLA measures too much jitter on a connection, IP SLA can generate a notification to a network management application or trigger another IP SLA operation to gather more data.

IP SLA reaction configuration is performed by using the **ipsla reaction operation** command.

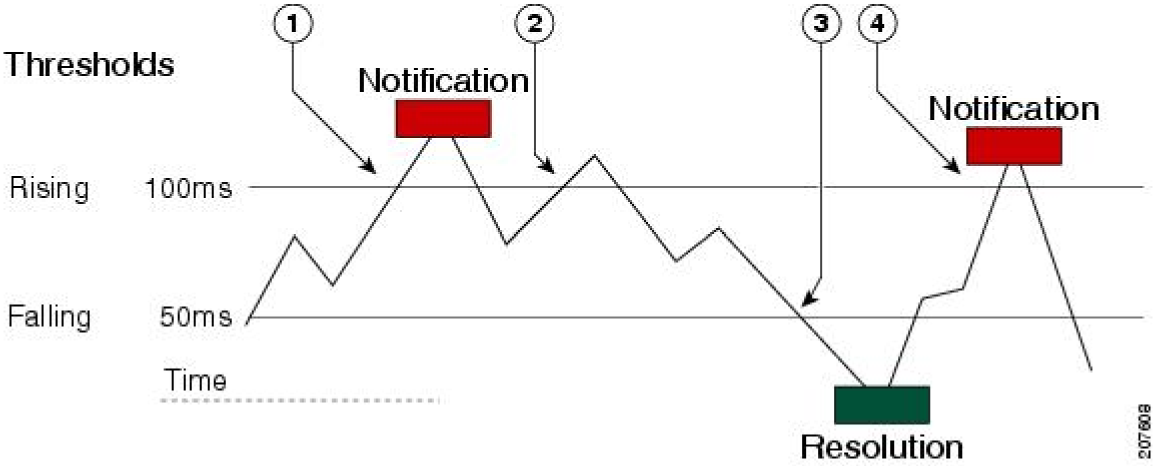
IP SLA Threshold Monitoring and Notifications

IP SLA supports threshold monitoring for performance parameters, such as jitter-average, bidirectional round-trip time, and connectivity. For packet loss and jitter, notifications can be generated for violations in either direction (for example, the source to the destination and the destination to the source) or for round-trip values.

Notifications are not issued for every occurrence of a threshold violation. An event is sent and a notification is issued when the rising threshold is exceeded for the first time. Subsequent threshold-exceeded notifications are issued only after the monitored value falls below the falling threshold before exceeding the rising threshold again.

The following figure illustrates the sequence for a triggered reaction that occurs when the monitored element exceeds the upper threshold.

Figure 7: IP SLAs Triggered Reaction Condition and Notifications for Threshold Exceeded



1	An event is sent and a threshold-exceeded notification is issued when the rising threshold is exceeded for the first time.
2	Consecutive over-rising threshold violations occur without issuing additional notifications.
3	The monitored value goes below the falling threshold.
4	Another threshold-exceeded notification is issued when the rising threshold is exceeded only after the monitored value first fell below the falling threshold.

Similarly, a lower-threshold notification is also issued the first time that the monitored element falls below the falling threshold. Subsequent notifications for lower-threshold violations are issued only after the rising threshold is exceeded before the monitored value falls below the falling threshold again.

MPLS LSP Monitoring

The IP Service Level Agreements (SLAs) label switched path (LSP) monitor feature provides the capability to proactively monitor Layer 3 Multiprotocol Label Switching (MPLS) Virtual Private Networks (VPNs). This feature is useful for determining network availability or testing network connectivity between provider edge (PE) routers in an MPLS VPN. When configured, MPLS LSP monitor automatically creates and deletes IP SLA LSP ping or LSP traceroute operations based on network topology.

The MPLS LSP monitor feature also allows you to perform multi-operation scheduling of IP SLA operations and supports proactive threshold violation monitoring through SNMP trap notifications and syslog messages.

To use the MPLS LSP monitor feature, you must understand these concepts:

How MPLS LSP Monitoring Works

The MPLS LSP monitor feature provides the capability to proactively monitor Layer 3 MPLS VPNs. The general process for how the MPLS LSP monitor works is as follows:

1. The user configures an MPLS LSP monitor instance.

   Configuring an MPLS LSP monitor instance is similar to configuring a standard IP SLA operation. To illustrate, all operation parameters for an MPLS LSP monitor instance are configured after an identification number for the operation is specified. However, unlike standard IP SLA operations, these configured parameters are then used as the base configuration for the individual IP SLA LSP ping and LSP traceroute operations that will be created by the MPLS LSP monitor instance.

   When the first MPLS LSP monitor instance is configured and scheduled to begin, BGP next-hop neighbor discovery is enabled. See the BGP Next-hop Neighbor Discovery , on page 205.

2. The user configures proactive threshold violation monitoring for the MPLS LSP monitor instance.

3. The user configures multioperation scheduling parameters for the MPLS LSP monitor instance.

4. Depending on the configuration options chosen, the MPLS LSP monitor instance automatically creates individual IP SLA LSP ping or LSP traceroute operations for each applicable BGP next-hop neighbor.

   For any given MPLS LSP monitor operation, only one IP SLA LSP ping or LSP traceroute operation is configured per BGP next-hop neighbor. However, more than one MPLS LSP monitor instance can be running on a particular PE router at the same time. (For more details, see the note at the end of this section.)

5. Each IP SLA LSP ping or LSP traceroute operation measures network connectivity between the source PE router and the discovered destination PE router.

Note More than one MPLS LSP monitor instance can be running on a particular PE router at the same time. For example, one MPLS LSP monitor instance can be configured to discover BGP next-hop neighbors belonging to the VRF named VPN1. On the same PE router, another MPLS LSP monitor instance can be configured to discover neighbors belonging to the VRF named VPN2. In this case, if a BGP next-hop neighbor belonged to both VPN1 and VPN2, then the PE router would create two IP SLA operations for this neighbor—one for VPN1 and one for VPN2.

Adding and Deleting IP SLA Operations from the MPLS LSP Monitor Database

The MPLS LSP monitor instance receives periodic notifications about BGP next-hop neighbors that have been added to or removed from a particular VPN. This information is stored in a queue maintained by the MPLS LSP monitor instance. Based on the information in the queue and user-specified time intervals, new IP SLA operations are automatically created for newly discovered PE routers and existing IP SLA operations are automatically deleted for any PE routers that are no longer valid.

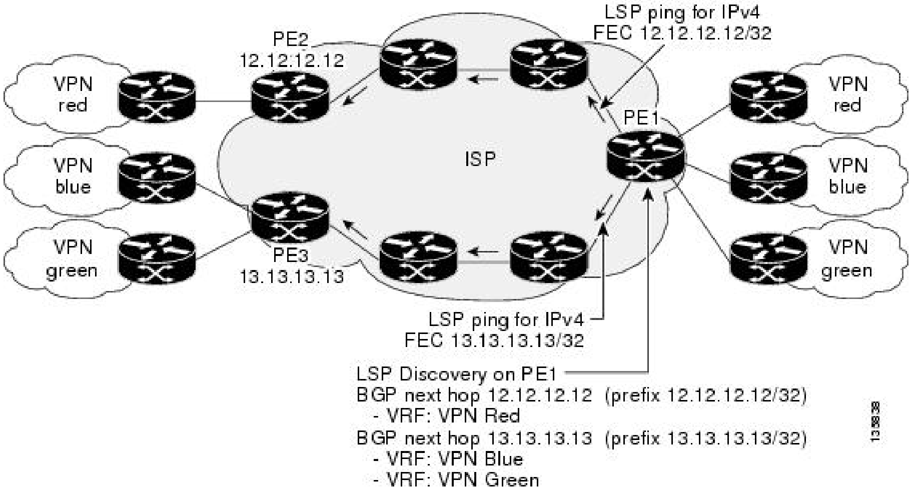
BGP Next-hop Neighbor Discovery

BGP next-hop neighbor discovery is used to find the BGP next-hop neighbors in use by any VRF associated with the source provider edge (PE) router. In most cases, these neighbors are PE routers.

When BGP next-hop neighbor discovery is enabled, a database of BGP next-hop neighbors in use by any VRF associated with the source PE router is generated, based on information from the local VRF and global routing tables. As routing updates are received, new BGP next-hop neighbors are added immediately to the database. However, BGP next-hop neighbors that are no longer valid are removed from the database only periodically, as defined by the user.

Figure 8: BGP Next-hop Neighbor Discovery for a Simple VPN, on page 206 shows how BGP next-hop neighbor discovery works for a simple VPN scenario for an Internet service provider (ISP). In this example, there are three VPNs associated with router PE1: red, blue, and green. From the perspective of router PE1, these VPNs are reachable remotely through BGP next-hop neighbors PE2 (router ID: 12.12.12.12) and PE3 (router ID: 13.13.13.13). When the BGP next-hop neighbor discovery process is enabled on router PE1, a database is generated based on the local VRF and global routing tables. The database in this example contains two BGP next-hop router entries, PE2 12.12.12.12 and PE3 13.13.13.13. The routing entries are maintained per next-hop router to distinguish which next-hop routers belong within which particular VRF. For each next-hop router entry, the IPv4 Forward Equivalence Class (FEC) of the BGP next-hop router in the global routing table is provided so that it can be used by the MPLS LSP ping operation.

Figure 8: BGP Next-hop Neighbor Discovery for a Simple VPN



IP SLA LSP Ping and LSP Traceroute Operations

This feature introduces support for the IP SLA LSP ping and IP SLA LSP traceroute operations. These operations are useful for troubleshooting network connectivity issues and determining network availability in an MPLS VPN. When using MPLS LSP monitoring, IP SLA LSP ping and LSP traceroute operations are automatically created to measure network connectivity between the source PE router and the discovered destination PE routers. Individual IP SLA LSP ping and LSP traceroute operations can also be manually configured. Manual configuration of these operations can be useful for troubleshooting a connectivity issue.

For more information about how to configure IP SLA LSP ping or LSP traceroute operations using MPLS LSP monitoring, see the Configuring an MPLS LSP Monitoring Ping Instance, on page 252 and the Configuring an MPLS LSP Monitoring Trace Instance, on page 256.

The IP SLA LSP ping and IP SLA LSP traceroute operations are based on the same infrastructure used by the MPLS LSP Ping and MPLS LSP Traceroute features, respectively, for sending and receiving echo reply and request packets to test LSPs.

Proactive Threshold Monitoring for MPLS LSP Monitoring

Proactive threshold monitoring support for the MPLS LSP Monitor feature provides the capability for triggering SNMP trap notifications and syslog messages when user-defined reaction conditions (such as a connection loss or timeout) are met. Configuring threshold monitoring for an MPLS LSP monitor instance is similar to configuring threshold monitoring for a standard IP SLAs operation.

Multi-operation Scheduling for the LSP Health Monitor

Multioperation scheduling support for the MPLS LSP Monitor feature provides the capability to easily schedule the automatically created IP SLA operations (for a given MPLS LSP monitor instance) to begin at intervals equally distributed over a specified duration of time (schedule period) and to restart at a specified frequency. Multioperation scheduling is particularly useful in cases where MPLS LSP monitoring is enabled on a source PE router that has a large number of PE neighbors and, therefore, a large number of IP SLAs operations running at the same time.

Note Newly created IP SLA operations (for newly discovered BGP next-hop neighbors) are added to the same schedule period as the operations that are currently running. To prevent too many operations from starting at the same time, the multioperation scheduling feature schedules the operations to begin at random intervals uniformly distributed over the schedule period.

LSP Path Discovery

LSP Path Discovery (LPD) is an enhancement to MPLS LSP monitor (MPLSLM) that allows operations that are part of an MPLSLM instance to initiate the path discovery process and to process the results. This feature relies on the tree trace capabilities provided by the MPLS OAM infrastructure through the LSPV server.

When multiple paths with equal cost exist between two PE routers, also know as equal cost multipath (ECMP), routers between these PE routers perform load balancing on the traffic, based on characteristics of the traffic being forwarded (for example. the destination address in the packet). In network topologies such as this, monitoring only one (or some) of the available paths among PE routers does not provide any guarantee that traffic will be forwarded correctly.

LPD is configured using the **path discover** command.

Note LPD functionality may create considerable CPU demands when large numbers of path discovery requests are received by the LSPV server at one time.

How to Implement IP Service Level Agreements

Configuring IP Service Levels Using the UDP Jitter Operation

The IP SLA UDP jitter monitoring operation is designed to diagnose network suitability for real-time traffic applications such as VoIP, Video over IP, or real-time conferencing.

Jitter means interpacket delay variance. When multiple packets are sent consecutively from source to destination—for example, 10 ms apart—and if the network is behaving ideally, the destination can receive them 10 ms apart. But if there are delays in the network (for example, queuing, arriving through alternate routes, and so on), the arrival delay between packets can be greater than or less than 10 ms. Using this example, a positive jitter value indicates that the packets arrived more than 10 ms apart. If the packets arrive 12 ms apart, positive jitter is 2 ms; if the packets arrive 8 ms apart, negative jitter is 2 ms. For delay-sensitive networks like VoIP, positive jitter values are undesirable, and a jitter value of 0 is ideal.

However, the IP SLA UDP jitter operation does more than just monitor jitter. The packets that IP SLA generates carry sending sequence and receiving sequence information for the packets, and sending and receiving time stamps from the source and the operational target. Based on these, UDP jitter operations are capable of measuring the following functions:

- Per-direction jitter (source to destination and destination to source)
- Per-direction packet-loss
- Per-direction delay (one-way delay)
- Round-trip delay (average round-trip time)

As the paths for the sending and receiving of data may be different (asymmetric), the per-direction data allows you to more readily identify where congestion or other problems are occurring in the network.

The UDP jitter operation functions by generating synthetic (simulated) UDP traffic. By default, ten packet-frames (N), each with a payload size of 32 bytes (S) are generated every 20 ms (T), and the operation is repeated every 60 seconds (F). Each of these parameters is user-configurable, so as to best simulate the IP service you are providing, or want to provide.

This section contains these procedures:

Enabling the IP SLA Responder on the Destination Device

The IP SLA Responder must be enabled on the target device, which is the operational target.

By configuring the **ipsla responder** command, you make the IP SLA Responder open a UDP port 1967 and wait for a control request (not for probes). You can open or close a port dynamically through the IP SLA control protocol (through UDP port 1967). In addition, you can configure permanent ports.

Permanent ports are open until the configuration is removed. Agents can send IP SLA probe packets to the permanent port directly without a control request packet because the port can be opened by the configuration.

If you do not use permanent ports, you have to configure only the **ipsla responder** command.

To use a dynamic port, use the **ipsla responder** command, as shown in this example:

```
configure
ipsla responder
```

The dynamic port is opened through the IP SLA control protocol on the responder side when you start an operation on the agent side.

The example is configured as a permanent port on the responder. UDP echo and UDP jitter can use a dynamic port or a permanent port. If you use a permanent port for UDP jitter, there is no check performed for duplicated or out-of-sequence packets. This is because there is no control packet to indicate the start or end of the probe sequence. Therefore, the verification for sequence numbers are skipped when using permanent ports.

Procedure

Step 1 **configure**

Example:

```
Router# configure
```

Enters XR Config mode.

Step 2 **ipsla responder**

Example:

```
Router(config)# ipsla responder
Router(config-ipsla-resp)#
```

Enables the IP SLA Responder for UDP echo or jitter operations.

Step 3 **type udp ipv4 address** *ip-address* **port** *port*

Example:

```
Router(config-ipsla-resp)# type udp ipv4 address 12.25.26.10 port 10001
```

Enables the permanent address and port on the IP SLA Responder.

Step 4 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

What to do next

After enabling the IP SLA Responder, see the Configuring and Scheduling a UDP Jitter Operation on the Source Device, on page 209 section.

Configuring and Scheduling a UDP Jitter Operation on the Source Device

The IP SLA operations function by generating synthetic (simulated) network traffic. A single IP SLA operation (for example, IP SLA operation 10) repeats at a given frequency for the lifetime of the operation.

A single UDP jitter operation consists of N UDP packets, each of size S, sent T milliseconds apart, from a source router to a target router, at a given frequency of F. By default, ten packets (N), each with a payload size of 32 bytes (S), are generated every 20 ms (T), and the operation is repeated every 60 seconds (F). Each of these parameters is user configurable, as shown in Table 43: UDP Jitter Operation Parameters, on page 210.

Table 43: UDP Jitter Operation Parameters

UDP Jitter Operation Parameter	Default	Configured Using
Number of packets (N)	10 packets	• **ipsla operation** command with the *operation-number* argument • **type udp jitter** command • **packet count** command with the *count* argument
Payload size per packet (S)	32 bytes	• **ipsla operation** command with the *operation-number* argument • **type udp jitter** command • **datasize request** command with the *size* argument
Time between packets, in milliseconds (T)	20 ms	• **ipsla operation** command with the *operation-number* argument • **type udp jitter** command • **packet interval** command with the *interval* argument
Elapsed time before the operation repeats, in seconds (F)	60 seconds	• **ipsla operation** command with the *operation-number* argument • **type udp jitter** command • **frequency** command with the *seconds* argument

Note If the **control disable** command is used to disable control packets while configuring IP SLA, the packets sent out from sender do not have sequence numbers. To calculate jitter, sequence number and time stamp values are required. So, jitter is not calculated when you use the **control disable** command.

Prerequisites for Configuring a UDP Jitter Operation on the Source Device

Use of the UDP jitter operation requires that the IP SLA Responder be enabled on the target Cisco device. To enable the IP SLA Responder, perform the task in the Enabling the IP SLA Responder on the Destination Device, on page 208 section.

Configuring and Scheduling a Basic UDP Jitter Operation on the Source Device

You can configure and schedule a UDP jitter operation.

Procedure

Step 1 **configure**

Example:

```
Router# configure
```

Example:

Enters global configuration mode.

Step 2 **ipsla operation** *operation-number*

Example:

```
Router(config)# ipsla operation 432
```

Specifies the operation number. The range is from 1 to 2048.

Step 3 **type udp jitter**

Example:

```
Router(config-ipsla-op)# type udp jitter
```

Configures the operation as a UDP jitter operation, and configures characteristics for the operation.

Step 4 **destination address** *ipv4address*

Example:

```
Router(config-ipsla-udp-jitter)# destination address 12.25.26.10
```

Specifies the IP address of the destination for the UDP jitter operation.

Step 5 **destination port** *port*

Example:

```
Router(config-ipsla-udp-jitter)# destination port 11111
```

Specifies the destination port number, in the range from 1 to 65535.

Step 6 **packet count** *count*

Example:

```
Router(config-ipsla-udp-jitter)# packet count 30
```

(Optional) Specifies the number of packets to be transmitted during a probe. For UDP jitter operation, the range is 1 to 60000. For ICMP path-jitter operation, the range is 1 to 100.

The default number of packets sent is 10.

Step 7 **packet interval** *interval*

Example:

```
Router(config-ipsla-udp-jitter)# packet interval 30
```

(Optional) Specifies the time between packets. The default interval between packets is 20 milliseconds.

Step 8 **frequency** *seconds*

Example:

```
Router(config-ipsla-udp-jitter)# frequency 300
```

(Optional) Sets the rate at which a specified IP SLA operation is sent into the network.

- (Optional) Use the *seconds* argument to specify the number of seconds between the IP SLA operations. Valid values are in the range from 1 to 12604800 seconds. The default is 60 seconds.

Step 9 **exit**

Example:

```
Router(config-ipsla-udp-jitter)# exit
Router(config-ipsla-op)# exit
Router(config-ipsla)# exit
Router(config)#
```

Exits from IP SLA configuration mode and operational mode, and returns the CLI to global configuration mode.

Configure and schedule a UDP jitter operation with additional characteristics

You can configure and schedule a UDP jitter operation.

Procedure

Step 1 **configure**

Example:

```
Router# configure
```

Example:

Enters global configuration mode.

Step 2 **ipsla operation** *operation-number*

Example:

```
Router(config)# ipsla operation 432
```

Specifies the operation number. The range is from 1 to 2048.

Step 3 **type udp jitter**

Example:

```
Router(config-ipsla-op)# type udp jitter
```

Configures the operation as a UDP jitter operation, and configures characteristics for the operation.

Step 4 **vrf** *vrf-name*

Example:

```
Router(config-ipsla-udp-jitter)# vrf VPN-A
```

(Optional) Enables the monitoring of a VPN (using a nondefault routing table) in a UDP jitter operation. Maximum length is 32 alphanumeric characters.

Step 5 **destination address** *ipv4address*

Example:

```
Router(config-ipsla-udp-jitter)# destination address 12.25.26.10
```

Specifies the IP address of the destination for the proper operation type.

Step 6 **destination port** *port*

Example:

```
Router(config-ipsla-udp-jitter)# destination port 11111
```

Specifies the destination port number, in the range from 1 to 65535.

Step 7 **frequency** *seconds*

Example:

```
Router(config-ipsla-udp-jitter)# frequency 300
```

(Optional) Sets the rate at which a specified IP SLA operation is sent into the network.

- (Optional) Use the *seconds* argument to specify the number of seconds between the IP SLA operations. Valid values are in the range from 1 to 12604800 seconds. The default is 60 seconds.

Step 8 **statistics** [**hourly** | **interval** *seconds*]

Example:

```
Router(config-ipsla-udp-jitter)# statistics hourly
Router(config-ipsla-op-stats)#
```

(Optional) Specifies the statistics collection parameters for UDP jitter operation.

Step 9 **buckets** *hours*

Example:

```
Router(config-ipsla-op-stats)# buckets 10
```

(Optional) Sets the number of hours in which statistics are maintained for the IP SLA operations. This command is valid only with the **statistics** command with **hourly** keyword. The range is 0 to 25 hours. The default value is 2 hours.

Step 10 **distribution count** *slot*

Example:

```
Router(config-ipsla-op-stats)# distribution count 15
```

(Optional) Sets the number of statistic distributions that are kept for each hop during the lifetime of the IP SLA operation. The range is 1 to 20. The default value is 1 distribution.

Step 11 **distribution interval** *interval*

Example:

```
Router(config-ipsla-op-stats)# distribution interval 20
```

(Optional) Sets the time interval for each statistical distribution. The range is 1 to 100 ms. The default value is 20 ms.

Step 12 **exit**

Example:

```
Router(config-ipsla-op-stats)# exit
```

Exits from IP SLA statistics configuration mode.

Step 13 **datasize request** *size*

Example:

```
Router(config-ipsla-udp-jitter)# datasize request 512
```

(Optional) Sets the data size in the payload of the operation's request packets. For UDP jitter, the range is from 16 to 1500 bytes.

Step 14 **timeout** *milliseconds*

Example:

```
Router(config-ipsla-udp-jitter)# timeout 10000
```

Sets the time that the specified IP SLA operation waits for a response from its request packet.

- (Optional) Use the *milliseconds* argument to specify the number of milliseconds that the operation waits to receive a response.

Note
If both the **frequency** and **timeout** values of an IPSLA UDP echo operation are the same, say, 1 s and 1000 ms respectively, the current probe iteration will exceed the configured frequency, say, 1002 ms, when the operation times out. As a result, the upcoming operation is skipped to ensure a correct operation interval cycle.

Step 15 **tos** *number*

Example:

```
Router(config-ipsla-udp-jitter)# tos 255
```

Specifies the type of service number.

Step 16 **exit**

Example:

```
Router(config-ipsla-udp-jitter)# exit
Router(config-ipsla-op)# exit
Router(config-ipsla)# exit
Router(config)#
```

Exits from IP SLA configuration mode and operational mode, and returns the CLI to global configuration mode.

Step 17 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Step 18 **show ipsla statistics** [*operation-number*]

Example:

```
Router # show ipsla statistics 432
```

Displays the current statistics.

Step 19 **show ipsla statistics aggregated** [*operation-number*]

Example:

```
Router # show ipsla statistics aggregated 432
```

Returns the hourly statistics (aggregated data) on the performance of the network.

The UDP jitter operation provides the following hourly statistics:

- Jitter statistics—Interprets telephony and multimedia conferencing requirements.
- Packet loss and packet sequencing statistics—Interprets telephony, multimedia conferencing, streaming media, and other low-latency data requirements.
- One-way latency and delay statistics—Interprets telephony, multimedia conferencing, and streaming media requirements.

Configuring the IP SLA for a UDP Echo Operation

To measure UDP performance on a network, use the IP SLA UDP echo operation. A UDP echo operation measures round-trip delay times and tests connectivity to Cisco devices and devices that are not Cisco devices. The results of a UDP echo operation can be useful in troubleshooting issues with business-critical applications.

Note The UDP echo operation requires a Cisco device that is running the IP SLA Responder or a non-Cisco device that is running the UDP echo service.

Depending on whether you want to configure a basic UDP echo operation or to configure a UDP echo operation with optional parameters, perform one of the following tasks:

Prerequisites for Configuring a UDP Echo Operation on the Source Device

If you are using the IP SLA Responder, ensure that you have completed the Enabling the IP SLA Responder on the Destination Device, on page 208 section.

Configuring and Scheduling a UDP Echo Operation on the Source Device

You can enable a UDP echo operation without any optional parameters.

Procedure

Step 1 **configure**

Example:

```
Router# configure
```

Example:

Enters global configuration mode.

Step 2 **ipsla operation** *operation-number*

Example:

```
Router(config)# ipsla operation 432
```

Specifies the operation number. The range is from 1 to 2048.

Step 3 **type udp echo**

Example:

```
Router(config-ipsla-op)# type udp echo
```

Configures the operation as a UDP echo operation, and configures characteristics for the operation.

Step 4 **destination address** *ipv4address*

Example:

```
Router(config-ipsla-udp-echo)# destination address 12.25.26.10
```

Specifies the IP address of the destination for the proper operation type.You can configure a permanent port on the IP SLA Responder side, or you can use an UDP echo server.

Step 5 **destination port** *port*

Example:

```
Router(config-ipsla-udp-echo)# destination port 11111
```

Specifies the destination port number, in the range from 1 to 65535.

Step 6 **frequency** *seconds*

Example:

```
Router(config-ipsla-udp-echo)# frequency 300
```

(Optional) Sets the rate at which a specified IP SLA operation is sent into the network.

- (Optional) Use the *seconds* argument to specify the number of seconds between the IP SLA operations. Valid values are in the range from 1 to 12604800 seconds. The default is 60 seconds.

Step 7 **exit**

Example:

```
Router(config-ipsla-udp-echo)# exit
Router(config-ipsla-op)# exit
Router(config-ipsla)# exit
Router(config)#
```

Exits IP SLA operation configuration mode and IP SLA configuration mode. Returns to global configuration mode.

Step 8 **show ipsla statistics** [*operation-number*]

Example:

```
Router# show ipsla statistics 432
```

Displays the current statistics.

Step 9 **show ipsla statistics aggregated** [*operation-number*]

Example:

```
Router# show ipsla statistics aggregated 1
```

Displays the hourly statistical errors and the hourly statistics for all the IP SLA operations or specified operation.

Configure and schedule a UDP echo operation with optional parameters on the source device

You can enable a UDP echo operation on the source device and configure some optional IP SLA parameters. The source device is the location at which the measurement statistics are stored.

Procedure

Step 1 **configure**

Example:

```
Router# configure
```

Example:

Enters global configuration mode.

Step 2 **ipsla operation** *operation-number*

Example:

```
Router(config)# ipsla operation 432
```

Specifies the operation number. The range is from 1 to 2048.

Step 3 **type udp echo**

Example:

```
Router(config-ipsla-op)# type udp echo
```

Configures the operation as a UDP echo operation, and configures characteristics for the operation.

Step 4 **vrf** *vrf-name*

Example:

```
Router(config-ipsla-udp-echo)# vrf VPN-A
```

(Optional) Enables the monitoring of a VPN (using a nondefault routing table) in a UDP echo operation. Maximum length is 32 alphanumeric characters.

Step 5 **destination address** *ipv4address*

Example:

```
Router(config-ipsla-udp-echo)# destination address 12.25.26.10
```

Specifies the IP address of the destination for the proper operation type.

Step 6 **destination port** *port*

Example:

```
Router(config-ipsla-udp-echo)# destination port 11111
```

Specifies the destination port number, in the range from 1 to 65535.

Step 7 **frequency** *seconds*

Example:

```
Router(config-ipsla-udp-echo)# frequency 300
```

(Optional) Sets the rate at which a specified IP SLA operation is sent into the network.

- (Optional) Use the *seconds* argument to specify the number of seconds between the IP SLA operations. Valid values are in the range from 1 to 12604800 seconds. The default is 60 seconds.

Step 8 **datasize request** *size*

Example:

```
Router(config-ipsla-udp-echo)# datasize request 512
```

(Optional) Sets the protocol data size in the payload of the IP SLA operation's request packet.

- Use the *size* argument to specify the protocol data size in bytes. The range is from 0 to the maximum of the protocol. The default is 1 byte.

Step 9 **tos** *number*

Example:

```
Router(config-ipsla-udp-echo)# tos 255
```

Defines a type of service (ToS) byte in the IP header of IP SLA operations.

Note
The ToS byte is converted to a Differentiated Services Code Point (DSCP) value, but you cannot enter the DSCP value directly. To use a DSCP value, multiply it by 4 and enter the result as the value of the *number* argument.

Step 10 **timeout** *milliseconds*

Example:

```
Router(config-ipsla-udp-echo)# timeout 10000
```

Sets the time that the specified IP SLA operation waits for a response from its request packet.

- Use the *milliseconds* argument to specify the number of milliseconds that the operation waits to receive a response.

Note
If both the **frequency** and **timeout** values of an IPSLA UDP echo operation are the same, say, 1 s and 1000 ms respectively, the current probe iteration will exceed the configured frequency, say, 1002 ms, when the operation times out. As a result, the upcoming operation is skipped to ensure a correct operation interval cycle.

Step 11 **exit**

Example:

```
Router(config-ipsla-udp-echo)# exit
Router(config-ipsla-op)# exit
Router(config-ipsla)# exit
Router(config)#
```

Exits IP SLA operation configuration mode and IPSLA configuration mode. Returns to global configuration mode.

Step 12 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Step 13 **show ipsla statistics enhanced aggregated** [*operation-number*] **interval** *seconds*

Example:

```
Router# show ipsla statistics enhanced aggregated 432
```

Displays the enhanced history statistics. You must configure the enhanced history statistics to display the sample output.

Step 14 **show ipsla statistics** [*operation-number*]

Example:

```
Router# show ipsla statistics 432
```

Displays the current statistics.

Configuring an ICMP Echo Operation

To monitor IP connections on a device, use the IP SLA ICMP echo operation. An ICMP echo operation measures end-to-end response times between a Cisco router and devices using IP. ICMP echo is used to troubleshoot network connectivity issues.

Note The ICMP echo operation does not require the IP SLA Responder to be enabled.

Depending on whether you want to configure and schedule a basic ICMP echo operation or configure and schedule an ICMP echo operation with optional parameters, perform one of the following procedures:

Configuring and Scheduling a Basic ICMP Echo Operation on the Source Device

You can enable and schedule an ICMP echo operation without any optional parameters.

Procedure

Step 1 **configure**

Example:

```
Router#configure
```

Enters global configurations XR Config mode.

Step 2 **ipsla operation** *operation-number*

Example:

```
Router(config)# ipsla operation 432
```

Specifies the operation number. The range is from 1 to 2048.

Step 3 **type icmp echo**

Example:

```
Router(config-ipsla-op)# type icmp echo
```

Defines an ICMP echo operation type.

Step 4 **destination address** *ipv4address*

Example:

```
Router(config-ipsla-icmp-echo)# destination address 12.25.26.10
```

Specifies the IP address of the destination for the proper operation type.

Step 5 **frequency** *seconds*

Example:

```
Router(config-ipsla-icmp-echo) frequency 300
```

(Optional) Sets the rate at which a specified IP SLA operation is sent into the network.

- (Optional) Use the *seconds* argument to specify the number of seconds between the IP SLA operations. Valid values are in the range from 1 to 12604800 seconds. The default is 60 seconds.

Step 6 **exit**

Example:

```
Router(config-ipsla-icmp-echo)# exit
Router(config-ipsla-op)# exit
Router(config-ipsla)# exit
Router(config)#
```

Exits IP SLA operation configuration mode and IP SLA configuration mode. Returns to global configuration mode.

Step 7 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Step 8 **show ipsla statistics** [*operation-number*]

Example:

```
Router# show ipsla statistics 432
```

Displays the current statistics.

Configure and scheduling an ICMP echo operation with optional parameters on the source device

You can enable an ICMP echo operation on the source device and configure some optional IP SLA parameters.

Procedure

Step 1 **configure**

Example:

```
Router# configure
```

Example:

Enters global configuration mode.

Step 2 **ipsla operation** *operation-number*

Example:

```
Router(config)# ipsla operation 432
```

Specifies the operation number. The range is from 1 to 2048.

Step 3 **type icmp echo**

Example:

```
Router(config-ipsla-op)# type icmp echo
```

Defines an ICMP echo operation type.

Step 4 **vrf** *vrf-name*

Example:

```
Router(config-ipsla-icmp-echo)# vrf VPN-A
```

(Optional) Enables the monitoring of a VPN (using a nondefault routing table) in an ICMP echo operation. Maximum length is 32 alphanumeric characters.

Step 5 **destination address** *ipv4address*

Example:

```
Router(config-ipsla-icmp-echo)# destination address 12.25.26.10
```

Specifies the IP address of the destination for the proper operation type.

Step 6 **frequency** *seconds*

Example:

```
Router(config-ipsla-icmp-echo)# frequency 300
```

(Optional) Sets the rate at which a specified IP SLA operation is sent into the network.

- (Optional) Use the *seconds* argument to specify the number of seconds between the IP SLA operations. Valid values are in the range from 1 to 12604800 seconds. The default is 60 seconds.

Step 7 **datasize request** *size*

Example:

```
Router(config-ipsla-icmp-echo)# datasize request 512
```

(Optional) Sets the protocol data size in the payload of the request packet for the specified IP SLA operation.

- Use the *bytes* argument to specify the protocol data size in bytes. The range is from 0 to 16384. The default is 36 bytes for ICMP echo operation.

Step 8 **tos** *number*

Example:

```
Router(config-ipsla-icmp-echo)# tos 1
```

Defines a type of service (ToS) byte in the IP header of IP SLA operations.

Note
The ToS byte can be converted to a Differentiated Services Code Point (DSCP) value, but you cannot enter the DSCP value directly. To use a DSCP value, multiply it by 4 and enter the result as the value of the *number* argument.

Step 9 **timeout** *milliseconds*

Example:

```
Router(config-ipsla-icmp-echo)# timeout 10000
```

Sets the time that the IP SLA operation waits for a response from its request packet.

- Use the *milliseconds* argument to specify the number of milliseconds that the operation waits to receive a response.

Note
If both the **frequency** and **timeout** values of an IPSLA UDP echo operation are the same, say, 1 s and 1000 ms respectively, the current probe iteration will exceed the configured frequency, say, 1002 ms, when the operation times out. As a result, the upcoming operation is skipped to ensure a correct operation interval cycle.

Step 10 **tag** *text*

Example:

```
Router(config-ipsla-icmp-echo)# tag ipsla
```

(Optional) Creates a user-specified identifier for an IP SLA operation.

Step 11 exit

Example:

```
Router(config-ipsla-icmp-echo)# exit
Router(config-ipsla-op)# exit
Router(config-ipsla)# exit
Router(config)#
```

Exits IP SLA operation configuration mode and IP SLA configuration mode. Returns to global configuration mode.

Step 12 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Step 13 **show ipsla statistics** [*operation-number*]

Example:

```
Router # show ipsla statistics 432
```

Displays the current statistics.

Configuring the ICMP Path-echo Operation

The IP SLA ICMP path-echo operation records statistics for each hop along the path that the IP SLA operation takes to reach its destination. The ICMP path-echo operation determines the hop-by-hop response time between a Cisco router and any IP device on the network by discovering the path using the traceroute facility.

The source IP SLA device uses traceroute to discover the path to the destination IP device. A ping is then used to measure the response time between the source IP SLA device and each subsequent hop in the path to the destination IP device.

Note The ICMP path-echo operation does not require the IP SLA Responder to be enabled.

Depending on whether you want to configure and schedule a basic ICMP path-echo operation or configure and schedule an ICMP path-echo operation with optional parameters, perform one of the following procedures:

Configuring and Scheduling a Basic ICMP Path-echo Operation on the Source Device

You can enable and schedule an ICMP path-echo operation without any optional parameters.

Procedure

Step 1 **configure**

Example:

```
Router#configure
```

Enters global configurations XR Config mode.

Step 2 **ipsla operation** *operation-number*

Example:

```
Router(config)# ipsla operation 432
```

Specifies the operation number. The range is from 1 to 2048.

Step 3 **type icmp path-echo**

Example:

```
Router(config-ipsla-op)# type icmp path-echo
Router(config-ipsla-icmp-path-echo)#
```

Defines an ICMP path-echo operation type.

Step 4 **destination address** *ipv4address*

Example:

```
Router(config-ipsla-icmp-path-echo)# destination address 12.25.26.10
```

Specifies the IP address of the destination for the proper operation type.

Step 5 **frequency** *seconds*

Example:

```
Router(config-ipsla-icmp-path-echo)# frequency 300
```

(Optional) Sets the rate at which a specified IP SLA operation is sent into the network.

- (Optional) Use the *seconds* argument to specify the number of seconds between the IP SLA operations. Valid values are in the range from 1 to 12604800 seconds. The default is 60 seconds.

Step 6 **exit**

Example:

```
Router(config-ipsla-icmp-path-echo)# exit
Router(config-ipsla-op)# exit
Router(config-ipsla)# exit
Router(config)#
```

Exits IP SLA operation configuration mode and IP SLA configuration mode. Returns to global configuration mode.

Step 7 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Step 8 **show ipsla statistics** [*operation-number*]

Example:

```
Router# show ipsla statistics 432
```

Displays the current statistics.

Configure and schedule an ICMP path-echo operation with optional parameters on the source device

You can enable an ICMP path-echo operation on the source device and configure some optional IP SLA parameters.

Procedure

Step 1 **configure**

Example:

```
Router# configure
```

Example:

Enters global configuration mode.

Step 2 **ipsla operation** *operation-number*

Example:

```
Router(config)# ipsla operation 432
```

Specifies the operation number. The range is from 1 to 2048.

Step 3 **type icmp path-echo**

Example:

```
Router(config-ipsla-op)# type icmp path-echo
Router(config-ipsla-icmp-path-echo)#
```

Defines an ICMP path-echo operation type.

Step 4 **vrf** *vrf-name*

Example:

```
Router(config-ipsla-imcp-path-echo)# vrf VPN-A
```

(Optional) Enables the monitoring of a VPN (using a nondefault routing table) in an ICMP path-echo operation. Maximum length is 32 alphanumeric characters.

Note
IP SLA with ICMP path-echo under a VRF does not support VRF in L3VPN scenarios where the next-hop router resides in the global VRF.

Step 5 **destination address** *ipv4address*

Example:

```
Router(config-ipsla-icmp-path-echo)# destination address 12.25.26.10
```

Specifies the IP address of the destination for the proper operation type.

Step 6 **frequency** *seconds*

Example:

```
Router(config-ipsla-icmp-path-echo)# frequency 300
```

(Optional) Sets the rate at which a specified IP SLA operation is sent into the network.

- (Optional) Use the *seconds* argument to specify the number of seconds between the IP SLA operations. Valid values are in the range from 1 to 12604800 seconds. The default is 60 seconds.

Step 7 **datasize request** *size*

Example:

```
Router(config-ipsla-icmp-path-echo)# datasize request 512
```

(Optional) Sets the protocol data size in the payload of the request packet for the specified IP SLA operation.

- Use the *bytes* argument to specify the protocol data size in bytes. The range is from 0 to 16384. The default is 36 bytes.

Step 8 **tos** *number*

Example:

```
Router(config-ipsla-icmp-path-echo)# tos 5
```

Defines a type of service (ToS) byte in the IP header of IP SLA operations.

Note
The ToS byte can be converted to a Differentiated Services Code Point (DSCP) value, but you cannot enter the DSCP value directly. To use a DSCP value, multiply it by 4 and enter the result as the *number* argument.

Step 9 **timeout** *milliseconds*

Example:

```
Router(config-ipsla-icmp-path-echo)# timeout 10000
```

Sets the time that the IP SLA operation waits for a response from its request packet.

- Use the *milliseconds* argument to specify the number of milliseconds that the operation waits to receive a response.

Note
If both the **frequency** and **timeout** values of an IPSLA UDP echo operation are the same, say, 1 s and 1000 ms respectively, the current probe iteration will exceed the configured frequency, say, 1002 ms, when the operation times out. As a result, the upcoming operation is skipped to ensure a correct operation interval cycle.

Step 10 **tag** *text*

Example:

```
Router(config-ipsla-icmp-path-echo)# tag ipsla
```

(Optional) Creates a user-specified identifier for an IP SLA operation.

Step 11 **lsr-path** *ipaddress1* {*ipaddress2* {... {*ipaddress8*}}}

Example:

```
Router(config-ipsla-icmp-path-echo)# lsr-path 20.25.22.1
```

Specifies the path in which to measure the ICMP echo response time.

- (Optional) Use the *ip address* argument of the intermediate node or nodes in a path to the destination.

Step 12 **exit**

Example:

```
Router(config-ipsla-icmp-path-echo)# exit
Router(config-ipsla-op)# exit
Router(config-ipsla)# exit
Router(config)#
```

Exits IP SLA operation configuration mode and IP SLA configuration mode. Returns to global configuration mode.

Step 13 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Step 14 **show ipsla statistics** [*operation-number*]

Example:

```
Router# show ipsla statistics 432
```

Displays the current statistics.

Configuring the ICMP Path-jitter Operation

The IP SLA ICMP path-jitter operation provides hop-by-hop jitter, packet loss, and delay measurement statistics in an IP network. The path-jitter operation functions differently than the standard UDP jitter operation, which provides total one-way data and total round-trip data.

The ICMP path-jitter operation can be used as a supplement to the standard UDP jitter operation. For example, results from the UDP jitter operation can indicate unexpected delays or high jitter values; the ICMP path-jitter operation can then be used to troubleshoot the network path and determine if traffic is bottlenecking in a particular segment along the transmission path.

The operation first discovers the hop-by-hop IP route from the source to the destination using a traceroute utility, and uses ICMP echoes to determine the response times, packet loss and approximate jitter values for each hop along the path. The jitter values obtained using the ICMP path-jitter operation are approximate because they do not account for delays at the target nodes.

The ICMP path-jitter operation functions by tracing the IP path from a source device to a specified destination device, then sending N number of Echo probes to each hop along the traced path, with a time interval of T milliseconds between each Echo probe. The operation as a whole is repeated at a frequency of once every F seconds. The attributes are user-configurable, as described in this table.

Table 44: ICMP Path-jitter Operation Parameters

ICMP Path-jitter Operation Parameter	Default	Configured Using
Number of echo probes (N)	10 echoes	• **ipsla operation** command with the *operation-number* argument • **packet count** command with the *count* argument
Time between Echo probes, in milliseconds (T)	20 ms	• **ipsla operation** command with the *operation-number* argument • **packet interval** command with the *interval* argument
The frequency of how often the operation is repeated (F)	once every 60 seconds	• **ipsla operation** command with the *operation-number* argument • **frequency** command with the *seconds* argument

Depending on whether you want to configure and schedule a basic ICMP path-jitter operation or configure and schedule an ICMP jitter operation with additional parameters, perform one of the following procedures:

Configuring and Scheduling a Basic ICMP Path-jitter Operation

You can configure and schedule an ICMP path-jitter operation using the general default characteristics for the operation.

Procedure

Step 1 **configure**

Example:

```
Router# configure
```

Example:

Enters global configuration mode.

Step 2 **ipsla operation** *operation-number*

Example:

```
Router(config)# ipsla operation 432
```

Specifies the operation number. The range is from 1 to 2048.

Step 3 **type icmp path-jitter**

Example:

```
Router(config-ipsla-op)# type icmp path-jitter
```

Defines an ICMP path-jitter operation type.

Step 4 **destination address** *ipv4address*

Example:

```
Router(config-ipsla-icmp-path-jitter)# destination address 12.25.26.10
```

Specifies the IP address of the destination for the proper operation type.

Step 5 **packet count** *count*

Example:

```
Router(config-ipsla-icmp-path-jitter)# packet count 30
```

(Optional) Specifies the number of packets to be transmitted during a probe. For UDP jitter operation, the range is 1 to 60000. For ICMP path-jitter operation, the range is 1 to 100.

The default number of packets sent is 10.

Step 6 **packet interval** *interval*

Example:

```
Router(config-ipsla-icmp-path-jitter)# packet interval 30
```

(Optional) Specifies the time between packets. The default interval between packets is 20 milliseconds.

Step 7 **frequency** *seconds*

Example:

```
Router(config-ipsla-icmp-path-jitter)# frequency 300
```

(Optional) Sets the rate at which a specified IP SLA operation is sent into the network.

- (Optional) Use the *seconds* argument to specify the number of seconds between the IP SLA operations. Valid values are in the range from 1 to 12604800 seconds. The default is 60 seconds.

Step 8 exit

Example:

```
Router(config-ipsla-icmp-path-jitter)# exit
Router(config-ipsla-op)# exit
Router(config-ipsla)# exit
Router(config)#
```

Exits IP SLA operation configuration mode and IP SLA configuration mode. Returns to global configuration mode.

Step 9 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.

- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Step 10 **show ipsla statistics** [*operation-number*]

Example:

```
Router# show ipsla statistics 432
```

Displays the current statistics.

Configure and schedule an ICMP path-jitter operation with additional parameters

You can enable an ICMP path-echo operation on the source device and configure some optional IP SLA parameters.

Procedure

Step 1 **configure**

Example:

```
Router# configure
```

Example:

Enters global configuration mode.

Step 2 **ipsla operation** *operation-number*

Example:

```
Router(config)# ipsla operation 432
```

Specifies the operation number. The range is from 1 to 2048.

Step 3 **type icmp path-jitter**

Example:

```
Router(config-ipsla-op)# type icmp path-jitter
```

Defines an ICMP path-jitter operation type.

Step 4 **vrf** *vrf-name*

Example:

```
Router(config-ipsla-imcp-path-jitter)# vrf VPN-A
```

(Optional) Enables the monitoring of a VPN (using a nondefault routing table) in an ICMP path-jitter operation. Maximum length is 32 alphanumeric characters.

Step 5 **lsr-path** *ip-address*

Example:

```
Router(config-ipsla-imcp-path-jitter)# lsr-path 20.25.22.1
```

Specifies that a loose source routing path is to be used.

Step 6 **destination address** *ipv4address*

Example:

```
Router(config-ipsla-icmp-path-jitter)# destination address 12.25.26.10
```

Specifies the IP address of the destination for the proper operation type.

Step 7 **packet count** *count*

Example:

```
Router(config-ipsla-icmp-path-jitter)# packet count 30
```

(Optional) Specifies the number of packets to be transmitted during a probe. For UDP jitter operation, the range is 1 to 60000. For ICMP path-jitter operation, the range is 1 to 100.

The default number of packets sent is 10.

Step 8 **packet interval** *interval*

Example:

```
Router(config-ipsla-icmp-path-jitter)# packet interval 30
```

(Optional) Specifies the time between packets. The default interval between packets is 20 milliseconds

Step 9 **frequency** *seconds*

Example:

```
Router(config-ipsla-icmp-path-jitter)# frequency 300
```

(Optional) Sets the rate at which a specified IP SLA operation is sent into the network.

- (Optional) Use the *seconds* argument to specify the number of seconds between the IP SLA operations. Valid values are in the range from 1 to 12604800 seconds. The default is 60 seconds.

Step 10 **datasize request** *size*

Example:

```
Router(config-ipsla-icmp-path-jitter)# datasize request 512
```

(Optional) Sets the protocol data size in the payload of the request packet for the specified IP SLA operation.

- Use the *size* argument to specify the protocol data size in bytes. The default for jitter is 36 bytes. The range is 0 to 16384 bytes.

Step 11 **tos** *number*

Example:

```
Router(config-ipsla-icmp-path-jitter)# tos 1
```

Defines a type of service (ToS) byte in the IP header of IP SLA operations.

Note
The ToS byte can be converted to a Differentiated Services Code Point (DSCP) value, but you cannot enter the DSCP value directly. To use a DSCP value, multiply it by 4 and enter the result as the *number* argument.

Step 12 **timeout** *milliseconds*

Example:

```
Router(config-ipsla-icmp-path-jitter)# timeout 10000
```

Sets the time that the IP SLA operation waits for a response from its request packet.

- Use the *milliseconds* argument to specify the number of milliseconds that the operation waits to receive a response.

Note
If both the **frequency** and **timeout** values of an IPSLA UDP echo operation are the same, say, 1 s and 1000 ms respectively, the current probe iteration will exceed the configured frequency, say, 1002 ms, when the operation times out. As a result, the upcoming operation is skipped to ensure a correct operation interval cycle.

Step 13 **tag** *text*

Example:

```
Router(config-ipsla-icmp-path-jitter)# tag ipsla
```

(Optional) Creates a user-specified identifier for an IP SLA operation.

Step 14 **exit**

Example:

```
Router(config-ipsla-icmp-path-jitter)# exit
Router(config-ipsla-op)# exit
Router(config-ipsla)# exit
Router(config)#
```

Exits IP SLA operation configuration mode and IP SLA configuration mode. Returns to global configuration mode.

Step 15 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Step 16 **show ipsla statistics** [*operation-number*]

Example:

```
Router# show ipsla statistics 432
```

Displays the current statistics.

Configuring IP SLA MPLS LSP Ping and Trace Operations

The MPLS LSP ping and trace operations allow service providers to monitor label switched paths (LSPs) and quickly isolate MPLS forwarding problems. Use these IP SLA operations to troubleshoot network connectivity between a source router and a target router. To test LSPs, the MPLS LSP ping and trace operations send echo request packets and receive echo reply packets.

To configure and schedule an MPLS LSP ping or trace operation, perform one of the following tasks:

Configuring and Scheduling an MPLS LSP Ping Operation

An MPLS LSP ping operation tests connectivity between routers along an LSP path in an MPLS network by sending an echo request (User Datagram Protocol (UDP) packet) to the end of the LSP, and receiving an echo reply back that contains diagnostic data.

The MPLS echo request packet is sent to a target router through the use of the appropriate label stack associated with the LSP to be validated. Use of the label stack causes the packet to be forwarded over the LSP itself.

The destination IP address of the MPLS echo request packet is different from the address used to select the label stack. The destination IP address is defined as a 127.x.y.z/8 address. The 127.x.y.z/8 address prevents the IP packet from being IP switched to its destination if the LSP is broken.

An MPLS echo reply is sent in response to an MPLS echo request. The reply is sent as an IP packet and it is forwarded using IP, MPLS, or a combination of both types of switching. The source address of the MPLS echo reply packet is an address obtained from the router generating the echo reply. The destination address is the source address of the router that originated the MPLS echo request packet. The MPLS echo reply destination port is set to the echo request source port.

The MPLS LSP ping operation verifies LSP connectivity by using one of the supported Forwarding Equivalence Class (FEC) entities between the ping origin and egress node of each FEC. The following FEC types are supported for an MPLS LSP ping operation:

- LDP IPv4 prefixes (configured with the **target ipv4** command)
- MPLS TE tunnels (configured with the **target traffic-eng tunnel** command)
- Pseudowire (configured with the **target pseudowire** command)

Procedure

Step 1 **configure**

Example:

```
Router# configure
```

Enters XR Config mode.

Step 2 **ipsla operation** *operation-number*

Example:

```
Router(config)# ipsla operation 432
```

Configures an IP SLA operation and specifies the operation number. The range is from 1 to 2048.

Step 3 **type mpls lsp ping**

Example:

```
Router(config-ipsla-op)# type mpls lsp ping
```

Configures an MPLS LSP ping operation and enters IP SLA MPLS LSP Ping configuration mode.

Step 4 **output interface** *type interface-path-id*

Example:

```
Router(config-ipsla-mpls-lsp-ping)# output interface pos 0/1/0/0
```

(Optional) Configures the echo request output interface to be used for LSP ping operations.

Note
You cannot use the **output interface** command if pseudowire is specified as the target to be used in an MPLS LSP ping operation

Step 5 **target** {**ipv4** *destination-address destination-mask* | **traffic-eng tunnel** *tunnel-interface* | **pseudowire** *destination-address circuit-id*}

Example:

```
Router(config-ipsla-mpls-lsp-ping)# target ipv4 10.25.26.10 255.255.255.255
```

or

```
Router(config-ipsla-mpls-lsp-ping)# target ipv4 10.25.26.10/32
```

or

```
Router(config-ipsla-mpls-lsp-ping)# target traffic-eng tunnel 12
```

or

```
Router(config-ipsla-mpls-lsp-ping)# target pseudowire 192.168.1.4 4211
```

Specifies the target destination of the MPLS LSP ping operation as a LDP IPv4 address, MPLS traffic engineering tunnel, or pseudowire.

Step 6 **lsp selector ipv4** *ip-address*

Example:

```
Router(config-ipsla-mpls-lsp-ping)# lsp selector ipv4 127.0.0.2
```

(Optional) Specifies the local host IPv4 address used to select the LSP in an MPLS LSP ping operation.

Step 7 **force explicit-null**

Example:

```
Router(config-ipsla-mpls-lsp-ping)# force explicit-null
```

(Optional) Adds an explicit null label to the label stack of an LSP when an echo request is sent.

Step 8 **reply dscp** *dscp-bits*

Example:

```
Router(config-ipsla-mpls-lsp-ping)# reply dscp 2
```

(Optional) Specifies the differentiated services codepoint (DSCP) value to be used in echo reply packets.Valid values are from 0 to 63.

Reserved keywords such as EF (expedited forwarding) and AF11 (assured forwarding class AF11) can be specified instead of numeric values.

Step 9 **reply mode** {**control-channel** | **router-alert**}

Example:

```
Router(config-ipsla-mpls-lsp-ping)# reply mode router-alert
```

or

```
Router(config-ipsla-mpls-lsp-ping)# reply mode control-channel
```

(Optional) Sets echo requests to send echo reply packets by way of a control channel in an MPLS LSP ping operation, or to reply as an IPv4 UDP packet with IP router alert. The router-alert reply mode forces an echo reply packet to be specially handled by the transit LSR router at each intermediate hop as it moves back to the destination.

Note
The **control-channel** keyword can be used only if the target is set to pseudowire.

Step 10 **exp** *exp-bits*

Example:

```
Router(config-ipsla-mpls-lsp-ping)# exp 5
```

(Optional) Specifies the MPLS experimental field (EXP) value to be used in the header of echo reply packets. Valid values are from 0 to 7.

Step 11 **ttl** *time-to-live*

Example:

```
Router(config-ipsla-mpls-lsp-ping)# ttl 200
```

(Optional) Specifies the time-to-live (TTL) value used in the MPLS label of echo request packets. Valid values are from 1 to 255.

Step 12 **exit**

Example:

```
Router(config-ipsla-mpls-lsp-ping)# exit
Router(config-ipsla-op)# exit
Router(config-ipsla)# exit
Router(config)#
```

Exits IP SLA MPLS LSP Ping configuration mode and IP SLA configuration mode. Returns to XR Config mode.

Step 13 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Step 14 **show ipsla statistics** [*operation-number*]

Example:

```
Router# show ipsla statistics 432
```

Displays IP SLA statistics for the current MPLS LSP ping operation.

Configuring and Scheduling an MPLS LSP Trace Operation

An MPLS LSP trace operation traces the hop-by-hop route of LSP paths to a target router in an MPLS network by sending echo requests (UDP packets) to the control plane of each transit label switching router (LSR). A transit LSR performs various checks to determine if it is a transit LSR for the LSP path. A trace operation allows you to troubleshoot network connectivity and localize faults hop-by-hop.

Echo request and reply packets validate the LSP. The success of an MPLS LSP trace operation depends on the transit router processing the MPLS echo request when it receives a labeled packet.

The transit router returns an MPLS echo reply containing information about the transit hop in response to any time-to-live (TTL)-expired MPLS packet or LSP breakage. The destination port of the MPLS echo reply is set to the echo request source port.

In an MPLS LSP trace operation, each transit LSR returns information related to the type of Forwarding Equivalence Class (FEC) entity that is being traced. This information allows the trace operation to check if the local forwarding information matches what the routing protocols determine as the LSP path.

An MPLS label is bound to a packet according to the type of FEC used for the LSP. The following FEC types are supported for an MPLS LSP trace operation:

- LDP IPv4 prefixes (configured with the **target ipv4** command)
- MPLS TE tunnels (configured with the **target traffic-eng tunnel** command)

Procedure

Step 1 **configure**

Example:

```
Router# configure
```

Enters XR Config mode.

Step 2 **ipsla operation** *operation-number*

Example:

```
Router(config)# ipsla operation 432
```

Configures an IP SLA operation and specifies the operation number. The range is from 1 to 2048.

Step 3 **type mpls lsp trace**

Example:

```
Router(config-ipsla-op)# type mpls lsp trace
```

Configures an MPLS LSP trace operation and enters IP SLA MPLS LSP Trace configuration mode.

Step 4 **output interface** *type interface-path-id*

Example:

```
Router(config-ipsla-mpls-lsp-ping)# output interface pos 0/1/0/0
```

(Optional) Configures the echo request output interface to be used for LSP trace operations.

Step 5 Do one of the following:

- **target ipv4** *destination-address destination-mask*
- **target traffic-eng tunnel** *tunnel-interface*

Example:

```
Router(config-ipsla-mpls-lsp-trace)# target ipv4 10.25.26.10 255.255.255.255
Router(config-ipsla-mpls-lsp-trace)# target ipv4 10.25.26.10/32
```

or

```
Router(config-ipsla-mpls-lsp-trace)# target traffic-eng tunnel 12
```

Specifies the target destination of the MPLS LSP trace operation as an LDP IPv4 address or MPLS traffic engineering tunnel.

Step 6 **lsp selector ipv4** *ip-address*

Example:

```
Router(config-ipsla-mpls-lsp-trace)# lsp selector ipv4 127.0.0.2
```

(Optional) Specifies the local host IPv4 address used to select the LSP in the MPLS LSP ping operation.

Step 7 **force explicit-null**

Example:

```
Router(config-ipsla-mpls-lsp-trace)# force explicit-null
```

(Optional) Adds an explicit null label to the label stack of an LSP when an echo request is sent.

Step 8 **reply dscp** *dscp-bits*

Example:

```
Router(config-ipsla-mpls-lsp-trace)# reply dscp 2
```

(Optional) Specifies the differentiated services codepoint (DSCP) value to be used in echo reply packets.Valid values are from 0 to 63.

Reserved keywords such as EF (expedited forwarding) and AF11 (assured forwarding class AF11) can be specified instead of numeric values.

Step 9 **reply mode router-alert**

Example:

```
Router(config-ipsla-mpls-lsp-trace)# reply mode router-alert
```

(Optional) Sets echo requests to reply as an IPv4 UDP packet with IP router alert. The router-alert reply mode forces an echo reply packet to be specially handled by the transit LSR router at each intermediate hop as it moves back to the destination.

Step 10 **exp** *exp-bits*

Example:

```
Router(config-ipsla-mpls-lsp-trace)# exp 5
```

(Optional) Specifies the MPLS experimental field (EXP) value to be used in the header of echo reply packets. Valid values are from 0 to 7.

Step 11 **ttl** *time-to-live*

Example:

```
Router(config-ipsla-mpls-lsp-trace)# ttl 20
```

(Optional) Specifies the time-to-live (TTL) value used in the MPLS label of echo request packets. Valid values are from 1 to 255.

Step 12 **exit**

Example:

```
Router(config-ipsla-mpls-lsp-trace)# exit
Router(config-ipsla-op)# exit
Router(config-ipsla)# exit
Router(config)#
```

Exits IP SLA MPLS LSP Trace configuration mode and IP SLA configuration mode. Returns to XR Config mode.

Step 13 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Step 14 **show ipsla statistics** [*operation-number*]

Example:

```
Router# show ipsla statistics 432
```

Displays the current IP SLA statistics for the trace operation.

Configuring IP SLA Reactions and Threshold Monitoring

If you want IP SLA to set some threshold and inform you of a threshold violation, the **ipsla reaction operation** command and the **ipsla reaction trigger** command are required. Perform the following procedures to configure IP SLA reactions and threshold monitoring:

Configuring Monitored Elements for IP SLA Reactions

IP SLA reactions are configured to be triggered when a monitored value exceeds or falls below a specified level or a monitored event (for example, timeout or connection-loss) occurs. These monitored values and events are called monitored elements. You can configure the conditions for a reaction to occur in a particular operation.

The types of monitored elements that are available are presented in the following sections:

Configuring Triggers for Connection-Loss Violations

You can configure a reaction if there is a connection-loss for the monitored operation.

Procedure

Step 1 **configure**

Example:

```
Router# configure
```

Enters XR Config mode.

Step 2 **ipsla reaction operation** *operation-number*

Example:

```
Router(config)# ipsla reaction operation 432
```

Configures certain actions that are based on events under the control of the IP SLA agent. The *operation-number* argument is the number of the IP SLA operations for the reactions that are configured. The range is from 1 to 2048.

Step 3 **react** [**connection-loss**]

Example:

```
Router(config-ipsla-react)# react connection-loss
Router(config-ipsla-react-cond)#
```

Specifies an element to be monitored for a reaction.

Use the **connection-loss** keyword to specify a reaction that occurs if there is a connection-loss for the monitored operation.

Step 4 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring Triggers for Jitter Violations

Jitter values are computed as source-to-destination and destination-to-source values. Events, for example, traps, can be triggered when the jitter value in either direction or both directions rises above a specified threshold or falls below a specified threshold. You can configure jitter-average as a monitored element.

Procedure

Step 1 **configure**

Example:

```
Router# configure
```

Enters XR Config mode.

Step 2 **ipsla reaction operation** *operation-number*

Example:

```
Router(config)# ipsla reaction operation 432
```

Configures certain actions that are based on events under the control of the IP SLA agent. The *operation-number* argument is the number of the IP SLA operations for the reactions that are configured. The range is from 1 to 2048.

Step 3 **react** [**jitter-average** {**dest-to-source** | **source-to-dest**}]

Example:

```
Router(config-ipsla-react)# react jitter-average
Router(config-ipsla-react-cond)#
```

Specifies an element to be monitored for a reaction.

A reaction occurs if the average round-trip jitter value violates the upper threshold or lower threshold. The following options are listed for the **jitter-average** keyword:

- **dest-to-source**—Specifies the jitter average destination to source (DS).
- **source-to-dest**—Specifies the jitter average source to destination (SD).

Step 4 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring Triggers for Packet Loss Violations

Packet-loss values are computed as source-to-destination and destination-to-source values. Events, for example, traps, can be triggered when the packet-loss values in either direction rise above a specified threshold or fall below a specified threshold. Perform this task to configure packet-loss as a monitored element.

Procedure

Step 1 **configure**

Example:

```
Router# configure
```

Enters XR Config mode.

Step 2 **ipsla reaction operation** *operation-number*

Example:

```
Router(config)# ipsla reaction operation 432
```

Configures certain actions that are based on events under the control of the IP SLA agent. The *operation-number* argument is the number of the IP SLA operations for the reactions that are configured. The range is from 1 to 2048.

Step 3 **react** [**packet-loss** [**dest-to-source** | **source-to-dest**]]

Example:

```
Router(config-ipsla-react)# react packet-loss dest-to-source
Router(config-ipsla-react-cond)#
```

Specifies an element to be monitored for a reaction.

The reaction on packet loss value violation is specified. The following options are listed for the **packet-loss** keyword:

- **dest-to-source**—Specifies the packet loss destination to source (DS) violation.
- **source-to-dest**—Specifies the packet loss source to destination (SD) violation.

Step 4 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring Triggers for Round-Trip Violations

Round-trip time (RTT) is a monitored value of all IP SLA operations. Events, for example, traps, can be triggered when the rtt value rises above a specified threshold or falls below a specified threshold. You can configure rtt as a monitored element.

Procedure

Step 1 **configure**

Example:

```
Router# configure
```

Enters XR Config mode.

Step 2 **ipsla reaction operation** *operation-number*

Example:

```
Router(config)# ipsla reaction operation 432
```

Configures certain actions that are based on events under the control of the IP SLA agent. The *operation-number* argument is the number of the IP SLA operations for the reactions that are configured. The range is from 1 to 2048.

Step 3 **react** [**rtt**]

Example:

```
Router(config-ipsla-react)# react rtt
Router(config-ipsla-react-cond)#
```

Specifies an element to be monitored for a reaction.

Use the **rtt** keyword to specify a reaction that occurs if the round-trip value violates the upper threshold or lower threshold.

Step 4 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.

- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring Triggers for Timeout Violations

You can configure triggers for timeout violations.

Procedure

Step 1 **configure**

Example:

```
Router# configure
```

Enters XR Config mode.

Step 2 **ipsla reaction operation** *operation-number*

Example:

```
Router(config)# ipsla reaction operation 432
```

Configures certain actions that are based on events under the control of the IP SLA agent. The *operation-number* argument is the number of the IP SLA operations for the reactions that are configured. The range is from 1 to 2048.

Step 3 **react** [**timeout**]

Example:

```
Router(config-ipsla-react)# react timeout
Router(config-ipsla-react-cond)#
```

Specifies an element to be monitored for a reaction.

Use the **timeout** keyword to specify a reaction that occurs if there is a timeout for the monitored operation.

Step 4 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring Triggers for Verify Error Violations

You can specify a reaction if there is an error verification violation.

Procedure

Step 1 **configure**

Example:

```
Router# configure
```

Enters XR Config mode.

Step 2 **ipsla reaction operation** *operation-number*

Example:

```
Router(config)# ipsla reaction operation 432
```

Configures certain actions that are based on events under the control of the IP SLA agent. The *operation-number* argument is the number of the IP SLA operations for the reactions that are configured. The range is from 1 to 2048.

Step 3 **react** [**verify-error**]

Example:

```
Router(config-ipsla-react)# react verify-error
Router(config-ipsla-react-cond)#
```

Specifies an element to be monitored for a reaction.

Use the **verify-error** keyword to specify a reaction that occurs if there is an error verification violation.

Step 4 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring Threshold Violation Types for IP SLA Reactions

For each monitored element, you can specify:

- Condition to check for the threshold value.
- Pattern of occurrences of the condition that can generate the reaction, such as a threshold type.

For example, you can specify that a reaction can occur for a particular element as soon as you observe the condition of interest by using the **threshold type immediate** command or when you observe the condition for three consecutive times by using the **threshold type consecutive** command.

The type of threshold defines the type of threshold violation (or combination of threshold violations) that triggers an event.

This table lists the threshold violation types.

Table 45: Threshold Violation Types for IP SLA Reactions

Type of Threshold Violation	Description
consecutive	Triggers an event only after a violation occurs a number of times consecutively. For example, the consecutive violation type can be used to configure an action to occur after a timeout occurs five times in a row or when the round-trip time exceeds the upper threshold value five times in a row. For more information, see Generating Events for Consecutive Violations, on page 247.
immediate	Triggers an event immediately when the value for a reaction type (such as response time) exceeds the upper threshold value or falls below the lower threshold value or when a timeout, connection-loss, or verify-error event occurs. For more information, see Generating Events for Each Violation, on page 246.
X of Y	Triggers an event after some number (X) of violations within some other number (Y) of probe operations (X of Y). For more information, see Generating Events for X of Y Violations, on page 248.
averaged	Triggers an event when the averaged totals of a value for X number of probe operations exceeds the specified upper-threshold value or falls below the lower-threshold value. For more information, see Generating Events for Averaged Violations, on page 249.

Generating Events for Each Violation

You can generate a trap or trigger another operation each time a specified condition is met.

Procedure

Step 1 **configure**

Example:

```
Router# configure
```

Enters XR Config mode.

Step 2 **ipsla reaction operation** *operation-number*

Example:

```
Router(config)# ipsla reaction operation 432
```

Configures certain actions that are based on events under the control of the IP SLA agent. The *operation-number* argument is the number of the IP SLA operations for the reactions that are configured. The range is from 1 to 2048.

Step 3 **react** [**connection-loss** | **jitter-average** {**dest-to-source** | **source-to-dest**} | **packet-loss** [**dest-to-source** | **source-to-dest**] | **rtt** | **timeout** | **verify-error**]

Example:

```
Router(config-ipsla-react)# react timeout
Router(config-ipsla-react-cond)#
```

Specifies an element to be monitored for a reaction.

A reaction is specified if there is a timeout for the monitored operation.

Step 4 **threshold type immediate**

Example:

```
Router(config-ipsla-react-cond)# threshold type immediate
```

Takes action immediately upon a threshold violation.

Step 5 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Generating Events for Consecutive Violations

You can generate a trap or trigger another operation after a certain number of consecutive violations.

Procedure

Step 1 **configure**

Example:

```
Router# configure
```

Enters XR Config mode.

Step 2 **ipsla reaction operation** *operation-number*

Example:

```
Router(config)# ipsla reaction operation 432
```

Configures certain actions that are based on events under the control of the IP SLA agent. The *operation-number* argument is the number of the IP SLA operations for the reactions that are configured. The range is from 1 to 2048.

Step 3 **react** [**connection-loss** | **jitter-average** {**dest-to-source** | **source-to-dest**} | **packet-loss** [**dest-to-source** | **source-to-dest**] | **rtt** | **timeout** | **verify-error**]

Example:

```
Router(config-ipsla-react)# react connection-loss
Router(config-ipsla-react-cond)#
```

Specifies an element to be monitored for a reaction.

A reaction is specified if there is a connection-loss for the monitored operation.

Step 4 **threshold type consecutive** *occurrences*

Example:

```
Router(config-ipsla-react-cond)# threshold type consecutive 8
```

Takes action after a number of consecutive violations. When the reaction condition is set for a consecutive number of occurrences, there is no default value. The number of occurrences is set when specifying the threshold type. The number of consecutive violations is from 1 to 16.

Step 5 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Generating Events for X of Y Violations

You can generate a trap or trigger another operation after some number (X) of violations within some other number (Y) of probe operations (X of Y). The **react** command with the **rtt** keyword is used as an example.

Procedure

Step 1 **configure**

Example:

```
Router# configure
```

Enters XR Config mode.

Step 2 **ipsla reaction operation** *operation-number*

Example:

```
Router(config)# ipsla reaction operation 432
```

Configures certain actions that are based on events under the control of the IP SLA agent. The *operation-number* argument is the number of the IP SLA operations for the reactions that are configured. The range is from 1 to 2048.

Step 3 **react** [**connection-loss** | **jitter-average** {**dest-to-source** | **source-to-dest**} | **packet-loss** [**dest-to-source** | **source-to-dest**] | **rtt** | **timeout** | **verify-error**]

Example:

```
Router(config-ipsla-react)# react rtt
Router(config-ipsla-react-cond)#
```

Specifies that a reaction occurs if the round-trip value violates the upper threshold or lower threshold.

Step 4 **threshold type xofy** *X value Y value*

Example:

```
Router(config-ipsla-react-cond)# threshold type xofy 7 7
```

When the reaction condition, such as threshold violations, are met for the monitored element after some *x* number of violations within some other *y* number of probe operations (for example, *x* of *y*), the action is performed as defined by the **action** command. The default is 5 for both *x value* and *y value;* for example, **xofy** 5 5. The valid range for each value is from 1 to 16.

Step 5 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Generating Events for Averaged Violations

You can generate a trap or trigger another operation when the averaged totals of X number of probe operations violate a falling threshold or rising threshold.

Procedure

Step 1 **configure**

Example:

```
Router# configure
```

Enters XR Config mode.

Step 2 **ipsla reaction operation** *operation-number*

Example:

```
Router(config)# ipsla reaction operation 432
```

Configures certain actions that are based on events under the control of the IP SLA agent. The *operation-number* argument is the number of the IP SLA operations for the reactions that are configured. The range is from 1 to 2048.

Step 3 **react** [**connection-loss** | **jitter-average** {**dest-to-source** | **source-to-dest**} | **packet-loss** [**dest-to-source** | **source-to-dest**] | **rtt** | **timeout** | **verify-error**]

Example:

```
Router(config-ipsla-react)# react packet-loss dest-to-source
Router(config-ipsla-react-cond)#
```

Specifies an element to be monitored for a reaction.

The reaction on packet loss value violation is specified. The following options are listed for the **packet-loss** keyword:

- **dest-to-source**—Specifies the packet loss destination to source (DS) violation.
- **source-to-dest**—Specifies the packet loss source to destination (SD) violation.

Step 4 **threshold type average** *number-of-probes*

Example:

```
Router(config-ipsla-react-cond)# threshold type average 8
```

Takes action on average values to violate a threshold.

Step 5 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Specifying Reaction Events

When a reaction condition is detected, you can configure the type of action that occurs by using the **action** command. The following types of actions are configured:

- **logging**—When the **logging** keyword is configured, a message is generated to the console to indicate that a reaction has occurred.
- **trigger**—When the **trigger** keyword is configured, one or more other operations can be started. As a result, you can control which operations can be started with the **ipsla reaction trigger** *op1 op2* command. This command indicates when *op1* generates an action type trigger and operation *op2* can be started.

You can specify reaction events. The **react** command with the **connection-loss** keyword is used as an example.

Procedure

Step 1 **configure**

Example:

```
Router# configure
```

Enters XR Config mode.

Step 2 **ipsla reaction operation** *operation-number*

Example:

```
Router(config)# ipsla reaction operation 432
```

Configures certain actions that are based on events under the control of the IP SLA agent. The *operation-number* argument is the number of the IP SLA operations for the reactions that are configured. The range is from 1 to 2048.

Step 3 **react** [**connection-loss** | **jitter-average** {**dest-to-source** | **source-to-dest**} | **packet-loss** [**dest-to-source** | **source-to-dest**] | **rtt** | **timeout** | **verify-error**]

Example:

```
Router(config-ipsla-react)# react connection-loss
Router(config-ipsla-react-cond)#
```

Specifies a reaction if there is a connection-loss for the monitored operation.

Step 4 **action** [**logging** | **trigger**]

Example:

```
Router(config-ipsla-react-cond)# action logging
```

Specifies what action or combination of actions the operation performs when you configure the **react** command or when threshold events occur. The following action types are described:

- **logging**—Sends a logging message when the specified violation type occurs for the monitored element. The IP SLA agent generates a syslog and informs SNMP. Then, it is up to the SNMP agent to generate a trap or not.
- **trigger**—Determines that the operational state of one or more operations makes the transition from pending to active when the violation conditions are met. The target operations to be triggered are specified using the **ipsla reaction trigger** command. A target operation continues until its life expires, as specified by lifetime value of the target operation. A triggered target operation must finish its life before it can be triggered again.

Step 5 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.

- **Cancel** —Remains in the configuration session, without committing the configuration changes.

Configuring the MPLS LSP Monitoring Instance on a Source PE Router

Perform this task to configure the operation parameters for an MPLS LSP monitor (MPLSLM) instance. The IP SLA measurement statistics are stored on the source PE router.

To configure an MPLS LSP monitor ping or trace instance, perform one of the following tasks:

Configuring an MPLS LSP Monitoring Ping Instance

Before you begin

Note MPLS LSP monitoring is configured on a PE router.

Procedure

Step 1 **configure**

Example:

```
Router#configure
```

Enters global configurations XR Config mode.

Step 2 **ipsla**

Example:

```
Router(config)# ipsla
```

Enters IP SLA configuration mode and configures IP service level agreements.

Step 3 **mpls discovery vpn**

Example:

```
Router(config-ipsla)# mpls discovery vpn
```

(Optional) Enters MPLS VPN BGP next-hop neighbor discovery configuration mode.

Step 4 **interval** *minutes*

Example:

```
Router(config-ipsla-mpls-discovery-vpn)# interval 120
```

(Optional) Specifies the time interval at which routing entries that are no longer valid are removed from the BGP next-hop neighbor discovery database of an MPLS VPN. The default time interval is 60 minutes.

Step 5 **exit**

Example:

```
Router(config-ipsla-mpls-discovery-vpn)# exit
```

Exits MPLS discovery VPN configuration mode.

Step 6 **mpls lsp-monitor**

Example:

```
Router(config-ipsla)# mpls lsp-monitor
Router(config-ipsla-mplslm)#
```

Enters MPLS LSP monitor mode. From this mode you can configure an LSP monitor instance, configure a reaction for an LSP monitor instance, or schedule an LSP monitor instance.

Step 7 **monitor** *monitor-id*

Example:

```
Router(config-ipsla-mplslm)# monitor 1
Router(config-ipsla-mplslm-def)#
```

Configures an MPLS LSP monitor instance and enters IP SLA MPLS LSP monitor configuration mode.

Step 8 **type mpls lsp ping**

Example:

```
Router(config-ipsla-mplslm-def)# type mpls lsp ping
```

Automatically creates an MPLS LSP ping operation for each discovered BGP next-hop address and enters the corresponding configuration mode to configure the parameters.

Step 9 **vrf** *vrf-name*

Example:

```
Router(config-ipsla-mplslm-lsp-ping)# vrf SANJOSE
```

(Optional) Enables the monitoring of a specific Virtual Private Network (VPN) routing and forwarding (VRF) instance in the ping operation. If no VRF is specified, the MPLS LSP monitoring instance monitors all VRFs.

Step 10 **scan interval** *scan-interval*

Example:

```
Router(config-ipsla-mplslm-lsp-ping)# scan interval 300
```

(Optional) Specifies the time interval (in minutes) at which the MPLS LSP monitor instance checks the scan queue for BGP next-hop neighbor updates. The default time interval is 240 minutes.

At each interval, a new IP SLA operation is automatically created for each newly discovered BGP next-hop neighbor listed in the MPLS LSP monitor instance scan queue.

Step 11 **scan delete-factor** *factor-value*

Example:

```
Router(config-ipsla-mplslm-lsp-ping)# scan delete-factor 2
```

(Optional) Specifies the number of times the MPLS LSP monitor instance should check the scan queue before automatically deleting IP SLA operations for BGP next-hop neighbors that are no longer valid.

The default scan factor is 1. In other words, each time the MPLS LSP monitor instance checks the scan queue for updates, it deletes IP SLA operations for BGP next-hop neighbors that are no longer valid.

If the scan factor is set to 0, IP SLA operations are never deleted by the MPLS LSP monitor instance. We do not recommend this configuration.

Step 12 **timeout** *milliseconds*

Example:

```
Router(config-ipsla-mplslm-lsp-ping)# timeout 50000
```

(Optional) Specifies the amount of time that each MPLS LSP operation waits for a response from the LSP verification (LSPV) server. The default value is 5000 milliseconds.

Step 13 **datasize request** *size*

Example:

```
Router(config-ipsla-mplslm-lsp-ping)# datasize request 512
```

(Optional) Specifies the payload size of the MPLS LSP echo request packets. The default value is 100 bytes.

Note
This command is available in MPLS LSP ping mode only.

Step 14 **lsp selector ipv4** *ip-address*

Example:

```
Router(config-ipsla-mplslm-lsp-ping)# lsp selector ipv4 127.10.10.1
```

(Optional) Specifies a local host IP address (127.*x.x.x*) that is used to select the label switched path (LSP) from among multiple LSPs. The default value is 127.0.0.1.

Step 15 **force explicit-null**

Example:

```
Router(config-ipsla-mplslm-lsp-ping)# force explicit-null
```

(Optional) Specifies whether an explicit null label is added to the label stack of MPLS LSP echo request packets. This is disabled by default.

Step 16 **reply dscp** *dscp-bits*

Example:

```
Router(config-ipsla-mplslm-lsp-ping)# reply dscp 5
```

(Optional) Specifies the differentiated services codepoint (DSCP) value to be used in the IP header of MPLS LSP echo reply packets.

Step 17 **reply mode router-alert**

Example:

```
Router(config-ipsla-mplslm-lsp-ping)# reply mode router-alert
```

(Optional) Enables the use of the router alert option in MPLS LSP echo reply packets. This is disabled by default.

Step 18 **ttl** *time-to-live*

Example:

```
Router(config-ipsla-mplslm-lsp-ping)# ttl 200
```

(Optional) Specifies the maximum hop count for an echo request packet to be used for MPLS LSP operations. The default value is 255.

Step 19 **tag** *text*

Example:

```
Router(config-ipsla-mplslm-lsp-ping)# tag mplslm-tag
```

(Optional) Creates a user-specified identifier for MPLS LSP operations.

Step 20 **exp** *exp-bits*

Example:

```
Router(config-ipsla-mplslm-lsp-ping)# exp 7
```

(Optional) Specifies the experimental field value to be used in the MPLS header of MPLS LSP echo request packets. The default value is 0.

Step 21 **statistics hourly** [**buckets** *hours*]

Example:

```
Router(config-ipsla-mplslm-lsp-ping)# statistics hourly buckets 2
```

(Optional) Specifies the statistics collection parameters for the operations in the MPLS LSP monitoring instance. The default number of hours is 2.

Step 22 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

What to do next

- Configure the reaction conditions.
- Schedule the MPLS LSP monitoring instance operations.

Configuring an MPLS LSP Monitoring Trace Instance

Before you begin

Note MPLS LSP monitoring is configured on a PE router.

Procedure

Step 1 **configure**

Example:

```
Router#configure
```

Enters global configurations XR Config mode.

Step 2 **ipsla**

Example:

```
Router(config)# ipsla
```

Enters IP SLA configuration mode and configures IP service level agreements.

Step 3 **mpls discovery vpn**

Example:

```
Router(config-ipsla)# mpls discovery vpn
```

(Optional) Enables MPLS VPN BGP next-hop neighbor discovery.

Step 4 **interval** *minutes*

Example:

```
Router(config-ipsla-mpls-discovery-vpn)# interval 120
```

(Optional) Specifies the time interval at which routing entries that are no longer valid are removed from the BGP next-hop neighbor discovery database of an MPLS VPN. The default time interval is 60 minutes.

Step 5 **exit**

Example:

```
Router(config-ipsla-mpls-discovery-vpn)# exit
```

Exits MPLS discovery VPN configuration mode.

Step 6 **mpls lsp-monitor**

Example:

```
Router(config-ipsla)# mpls lsp-monitor
Router(config-ipsla-mplslm)#
```

Enters MPLS LSP monitor mode. From this mode you can configure an LSP monitor instance, configure a reaction for an LSP monitor instance, or schedule an LSP monitor instance.

Step 7 **monitor** *monitor-id*

Example:

```
Router(config-ipsla-mplslm)# monitor 1
Router(config-ipsla-mplslm-def)#
```

Configures an MPLS LSP monitor instance and enters IP SLA MPLS LSP monitor configuration mode.

Step 8 **type mpls lsp trace**

Example:

```
Router(config-ipsla-mplsm-def)# type mpls lsp trace
```

Automatically creates an MPLS LSP trace operation for each discovered BGP next-hop address and enters the corresponding configuration mode to configure the parameters.

Step 9 **vrf** *vrf-name*

Example:

```
Router(config-ipsla-mplslm-lsp-trace)# vrf SANJOSE
```

(Optional) Enables the monitoring of a specific Virtual Private Network (VPN) routing and forwarding (VRF) instance in the traceroute operation. If no VRF is specified, the MPLS LSP monitoring instance monitors all VRFs.

Step 10 **scan interval** *scan-interval*

Example:

```
Router(config-ipsla-mplslm-lsp-trace)# scan interval 300
```

(Optional) Specifies the time interval (in minutes) at which the MPLS LSP monitor instance checks the scan queue for BGP next-hop neighbor updates. By default, the scan interval is set to 240 minutes, with a scan delete factor of 1. This means that any changes to the next hop reachability will only be reflected at the end of the 240-minute window. Consequently, if the next hop becomes unreachable, the probes will not be cleared until the scan interval concludes.

At each interval, a new IP SLA operation is automatically created for each newly discovered BGP next-hop neighbor listed in the MPLS LSP monitor instance scan queue.

Step 11 **scan delete-factor** *factor-value*

Example:

```
Router(config-ipsla-mplslm-lsp-trace)# scan delete-factor 2
```

(Optional) Specifies the number of times the MPLS LSP monitor instance should check the scan queue before automatically deleting IP SLA operations for BGP next-hop neighbors that are no longer valid.

The default scan factor is 1. In other words, each time the MPLS LSP monitor instance checks the scan queue for updates, it deletes IP SLA operations for BGP next-hop neighbors that are no longer valid.

If the scan factor is set to 0, IP SLA operations are never deleted by the MPLS LSP monitor instance. We do not recommend this configuration.

Step 12 **timeout** *milliseconds*

Example:

```
Router(config-ipsla-mplslm-lsp-trace)# timeout 50000
```

(Optional) Specifies the amount of time that each MPLS LSP operation waits for a response from the LSP verification (LSPV) server. The default value is 5000 milliseconds.

Step 13 **lsp selector ipv4** *ip-address*

Example:

```
Router(config-ipsla-mplslm-lsp-trace)# lsp selector ipv4 127.10.10.1
```

(Optional) Specifies a local host IP address (127.*x.x.x*) that is used to select the label switched path (LSP) from among multiple LSPs. The default value is 127.0.0.1.

Step 14 **force explicit-null**

Example:

```
Router(config-ipsla-mplslm-lsp-trace)# force explicit-null
```

(Optional) Specifies whether an explicit null label is added to the label stack of MPLS LSP echo request packets. This is disabled by default.

Step 15 **reply dscp** *dscp-bits*

Example:

```
Router(config-ipsla-mplslm-lsp-trace)# reply dscp 5
```

(Optional) Specifies the differentiated services codepoint (DSCP) value to be used in the IP header of MPLS LSP echo reply packets.

Step 16 **reply mode router-alert**

Example:

```
Router(config-ipsla-mplslm-lsp-trace)# reply mode router-alert
```

(Optional) Enables the use of the router alert option in MPLS LSP echo reply packets. This is disabled by default.

Step 17 **ttl** *time-to-live*

Example:

```
Router(config-ipsla-mplslm-lsp-trace)# ttl 40
```

(Optional) Specifies the maximum hop count for an echo request packet to be used for MPLS LSP operations. The default value is 30.

Step 18 **tag** *text*

Example:

```
Router(config-ipsla-mplslm-lsp-trace)# tag mplslm-tag
```

(Optional) Creates a user-specified identifier for MPLS LSP operations.

Step 19 **exp** *exp-bits*

Example:

```
Router(config-ipsla-mplslm-lsp-trace)# exp 7
```

(Optional) Specifies the experimental field value to be used in the MPLS header of MPLS LSP echo request packets. The default value is 0.

Step 20 **statistics hourly** [**buckets** *hours*]

Example:

```
Router(config-ipsla-mplslm-lsp-trace)# statistics hourly buckets 2
```

(Optional) Specifies the statistics collection parameters for the operations in the MPLS LSP monitoring instance. The default number of hours is 2.

Step 21 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

What to do next

- Configure the reaction conditions.
- Schedule the MPLS LSP monitoring instance operations.

Configuring the Reaction Conditions for an MPLS LSP Monitoring Instance on a Source PE Router

Perform this task to configure the reaction conditions for an MPLS LSP monitoring instance.

Before you begin

The MPLS LSP monitoring instance should be defined before you configure the reaction conditions.

Procedure

Step 1 **configure**

Example:

```
Router# configure
```

Enters XR Config mode.

Step 2 **ipsla**

Example:

```
Router(config)# ipsla
```

Enters IP SLA configuration mode and configures IP service level agreements.

Step 3 **mpls lsp-monitor**

Example:

```
Router(config-ipsla)# mpls lsp-monitor
Router(config-ipsla-mplslm)#
```

Enters MPLS LSP monitor mode. From this mode you can configure an LSP monitor instance, configure a reaction for an LSP monitor instance, or schedule an LSP monitor instance.

Step 4 **reaction monitor** *monitor-id*

Example:

```
Router(config-ipsla-mplslm)# reaction monitor 2
Router(config-ipsla-mplslm-react)#
```

Configures an MPLS LSP monitor instance reaction and enters IP SLA MPLS LSP monitor reaction configuration mode.

Step 5 **react** {**connection-loss** | **timeout**}

Example:

```
Router(config-ipsla-mplslm-react)# react connection-loss
```

Specifies that a reaction occurs if there is a one-way connection loss or timeout for the monitored operation. The reaction applies when the condition comes up for any of the automatically created operations.

Step 6 **action logging**

Example:

```
Router(config-ipsla-mplslm-react-cond)# action logging
```

Specifies that an event be logged as a result of the reaction condition and threshold.

Step 7 **threshold type** {**consecutive** *occurrences* | **immediate**}

Example:

```
Router(config-ipsla-mplslm-react-cond)# threshold type consecutive 10
```

Specifies that the designated action is taken after the specified number of consecutive violations or immediately. The valid range of *occurrences* is 1 to 16.

Step 8 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

What to do next

- Schedule the MPLS LSP monitoring instance operations.

Scheduling an MPLS LSP Monitoring Instance on a Source PE Router

Perform this task to schedule the operations in an MPLS LSP monitoring instance.

Procedure

Step 1 **configure**

Example:

```
Router# configure
```

Enters XR Config mode.

Step 2 **ipsla**

Example:

```
Router(config)# ipsla
```

Enters IP SLA configuration mode and configures IP service level agreements.

Step 3 **mpls lsp-monitor**

Example:

```
Router(config-ipsla)# mpls lsp-monitor
Router(config-ipsla-mplslm)#
```

Enters MPLS LSP monitor mode. From this mode you can configure an LSP monitor instance, configure a reaction for an LSP monitor instance, or schedule an LSP monitor instance.

Step 4 **schedule monitor** *monitor-id*

Example:

```
Router(config-ipsla-mplslm)# schedule monitor 2
Router(config-ipsla-mplslm-sched)#
```

Enters IP SLA MPLS LSP monitor schedule configuration mode to schedule the MPLS LSP monitor instance.

Step 5 **frequency** *seconds*

Example:

```
Router(config-ipsla-mplslm-sched)# frequency 600
```

(Optional) Specifies the frequency at which the schedule period is run. The default value is same as schedule period. The schedule period is specified using the **schedule period** command. You must specify this value before scheduling an MPLS LSP monitor instance start time.

Step 6 **schedule period** *seconds*

Example:

```
Router(config-ipsla-mplslm-sched)# schedule period 300
```

Specifies the amount of time, in seconds, during which all of the operations are scheduled to run. All operations are scheduled equally spaced throughout the schedule period.

Use the **frequency** command to specify how often the entire set of operations is performed. The frequency value must be greater than or equal to the schedule period.

You must specify this value before scheduling an MPLS LSP monitor instance start time.

Step 7 **start-time** *hh*:*mm*:*ss* [*day* | *month day*]

Example:

```
Router(config-ipsla-mplslm-sched)# start-time 11:45:00 July 4
```

Specifies the time when the MPLS LSP monitor instance starts collecting information. You must specify the scheduled time; otherwise, no information is collected.

Step 8 Use the **commit** or **end** command.

commit —Saves the configuration changes and remains within the configuration session.

end —Prompts user to take one of these actions:

- **Yes** — Saves configuration changes and exits the configuration session.
- **No** —Exits the configuration session without committing the configuration changes.
- **Cancel** —Remains in the configuration session, without committing the configuration changes.

LSP Path Discovery

LSP Path Discovery (LPD) is an enhancement to MPLS LSP monitor (MPLSLM) that allows operations that are part of an MPLSLM instance to initiate the path discovery process and to process the results. This feature relies on the tree trace capabilities provided by the MPLS OAM infrastructure through the LSPV server.

When multiple paths with equal cost exist between two PE routers, also know as equal cost multipath (ECMP), routers between these PE routers perform load balancing on the traffic, based on characteristics of the traffic being forwarded (for example. the destination address in the packet). In network topologies such as this, monitoring only one (or some) of the available paths among PE routers does not provide any guarantee that traffic will be forwarded correctly.

LPD is configured using the **path discover** command.

Note LPD functionality may create considerable CPU demands when large numbers of path discovery requests are received by the LSPV server at one time.

Configuration Examples for Implementing IP Service Level Agreements

This section provides these configuration examples:

Configuring IP Service Level Agreements: Example

The following example shows how to configure and schedule a UDP jitter operation:

```
configure
ipsla
 operation 101
  type udp jitter
   destination address 12.2.0.2
   statistics hourly
    buckets 5
    distribution count 5
    distribution interval 1
   !
   destination port 400
   statistics interval 120
    buckets 5
   !
  !
 !
 schedule operation 101
  start-time now
  life forever
 !
!
```

```
show ipsla statistics
Fri Nov 28 16:48:48.286 GMT
Entry number: 101
    Modification time: 16:39:36.608 GMT Fri Nov 28 2014
    Start time       : 16:39:36.633 GMT Fri Nov 28 2014
    Number of operations attempted: 10
    Number of operations skipped  : 0
    Current seconds left in Life  : Forever
    Operational state of entry    : Active
    Operational frequency(seconds): 60
    Connection loss occurred      : FALSE
    Timeout occurred              : FALSE
    Latest RTT (milliseconds)     : 3
    Latest operation start time   : 16:48:37.653 GMT Fri Nov 28 2014
    Next operation start time     : 16:49:37.653 GMT Fri Nov 28 2014
    Latest operation return code  : OK
    RTT Values:
      RTTAvg  : 3          RTTMin: 3          RTTMax : 4
      NumOfRTT: 10         RTTSum: 33         RTTSum2: 111
    Packet Loss Values:
      PacketLossSD       : 0          PacketLossDS : 0
      PacketOutOfSequence: 0          PacketMIA    : 0
      PacketLateArrival  : 0          PacketSkipped: 0
      Errors             : 0          Busies       : 0
      InvalidTimestamp   : 0
    Jitter Values :
      MinOfPositivesSD: 1          MaxOfPositivesSD: 1
      NumOfPositivesSD: 2          SumOfPositivesSD: 2
      Sum2PositivesSD : 2
      MinOfNegativesSD: 1          MaxOfNegativesSD: 1
      NumOfNegativesSD: 1          SumOfNegativesSD: 1
      Sum2NegativesSD : 1
      MinOfPositivesDS: 1          MaxOfPositivesDS: 1
      NumOfPositivesDS: 1          SumOfPositivesDS: 1
      Sum2PositivesDS : 1
      MinOfNegativesDS: 1          MaxOfNegativesDS: 1
      NumOfNegativesDS: 1          SumOfNegativesDS: 1
      Sum2NegativesDS : 1
      JitterAve: 1         JitterSDAve: 1      JitterDSAve: 1
      Interarrival jitterout: 0              Interarrival jitterin: 0
    One Way Values :
      NumOfOW: 0
      OWMinSD : 0          OWMaxSD: 0          OWSumSD: 0
      OWSum2SD: 0          OWAveSD: 0
      OWMinDS : 0          OWMaxDS: 0          OWSumDS: 0
      OWSum2DS: 0          OWAveDS: 0
```

Configuring IP SLA Reactions and Threshold Monitoring: Example

The following examples show how to configure IP SLA reactions and threshold monitoring. You can:

- Configure a reaction for attributes that activate a true or false condition, for example, 1, 5, or 6.
- Configure a reaction for attributes that accept a threshold value.
- Configure additional threshold type options.
- Configure either the logging or triggering of action types.

```
configure
ipsla operation 1
  type icmp echo
    timeout 5000
    destination address 223.255.254.254
    frequency 10
    statistics interval 30
    buckets 3
end

configure
ipsla operation 2
  type icmp path-echo
    destination address 223.255.254.254
    frequency 5
end

configure
```

```
ipsla reaction operation 1
  react timeout
   action trigger
   threshold type immediate
 exit
exit
  react rtt
   action logging
   threshold lower-limit 4 upper-limit 5
end
```

Operation 1 checks for timeout occurrence. If applicable, operation 1 generates a trigger event. If the **rtt** keyword exceeds 5, an error is logged.

If operation 1 generates a trigger event, operation 2 is started. The following example shows how to configure a reaction trigger operation by using the **ipsla reaction trigger** command:

```
configure
ipsla reaction trigger 1 2
end
```

Configuring IP SLA MPLS LSP Monitoring: Example

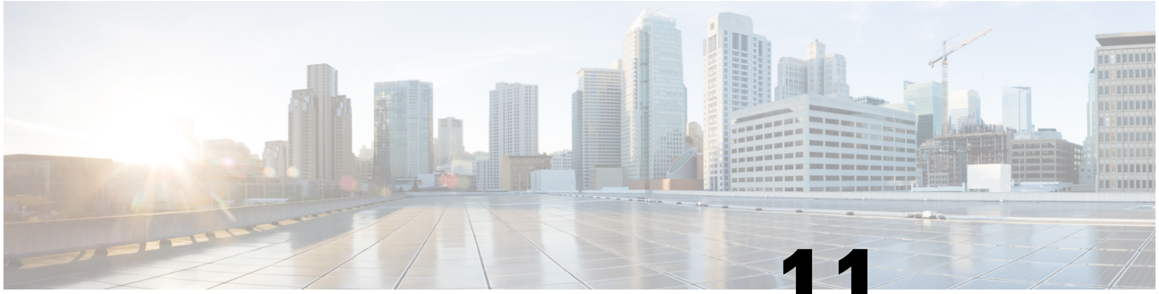
The following example illustrates how to configure IP SLA MPLS LSP monitoring:

```
ipsla
 mpls lsp-monitor
  monitor 1
   type mpls lsp ping
    vrf SANJOSE
    scan interval 300
    scan delete-factor 2
    timeout 10000
    datasize request 256
    lsp selector ipv4 127.0.0.10
    force explicit-null
    reply dscp af
    reply mode router-alert
    ttl 30
    exp 1
    statistics hourly
     buckets 1
    !
   !
  !
  reaction monitor 1
   react timeout
    action logging
    threshold type immediate
   !
   react connection-loss
    action logging
    threshold type immediate
   !
  !
  schedule monitor 1
   frequency 300
   schedule period 120
   start-time 11:45:00 July 4
  !
 !
 mpls discovery vpn
  interval 600
 !
!
```

Configuring LSP Path Discovery: Example

The following example illustrates how to configure LSP Path Discovery:

```
configure
ipsla
 mpls lsp-monitor
  monitor 1
   type mpls lsp ping
    path discover
     path retry 12
     path secondary frequency both 12
```



CHAPTER 11

Collecting Tech-Support Information

This module describes commands that are used for collecting tech-support information.

To use commands of this module, you must be in a user group associated with a task group that includes appropriate task IDs. If the user group assignment is preventing you from using any command, contact your AAA administrator for assistance.

This chapter covers the following topic:

Configuring Custom Profiles

Table 46: Feature History Table

Feature Name	Release	Description
Supporting Custom Profile show tech command	Release 7.5.1	This feature lets you run a customized list of **show** commands and **System Admin show** commands from all core protocols such as BGP, MPLS, Segment Routing and so on You can also generate **tech-support** information that is useful for Cisco Technical Support representatives when troubleshooting a router. This feature introduces the **show tech support custom profile-name** command.

You can group multiple Cisco IOS XR **show** commands, **System Admin show** commands, multiple show tech-support commands from IOS XR and Admin into a custom profile. A profile can be used for protocols such as BGP, MPLS, Segment Routing etc.

Restriction

- The **System Admin show** commands must be enclosed within double quotes.

Configuration Example

To configure the custom profile, perform the following instructions:

1. Create a custom profile using **customshowtech** *profile* command.

   For example,

   ```
   Router# configure
   Router(config)# customshowtech ospf_prof1
   ```

2. Add the existing **Show** commands or **System Admin show** commands to the profile that you created.

   For example,

   ```
   Router(config-cst-ospf_prof1)#command show ospf neighbor
   Router(config-cst-ospf_prof1)#command show ospf trace
   Router(config-cst-ospf_prof1)#admincommand "show version"
   Router(config-cst-ospf_prof1)#command show tech-support routing ospf
   ```

3. Use Commit.

   For example,

   ```
   Router(config-cst-profile)#commit
   ```

You can use the **show running-config customshowtech** *profile name* command to view whether the commit is successful.

For example,

```
Router(config-cst-ospf_prof1)#show running-config customshowtech
        ospf_prof1
Thu Oct 28 17:42:53.897 UTC
customshowtech ospf_prof1
 command show ospf neighbor
 command show ospf trace
 admincommand "show version"
 command show tech-support routing ospf
!
```

Running Configuration Example

Use the **show tech-support custom profile-name** command to run the custom profile.

For example,

```
Router# show tech-support custom profile-name ospf_prof1
```

Verification Example

Verify that the **show tech-support custom profile-name** command generates tech-support information. By default, the output of this command is saved on the router's hard disk in a file with *.tgz* extension. For example, */harddisk:/showtech/name.tgz*.

For example,

```
++ Show tech start time: 2021-Oct-28.174339.UTC ++
Thu Oct 28 17:43:39 UTC 2021 Waiting for gathering to complete
```

```
.......................................
Thu Oct 28 17:45:40 UTC 2021 Compressing show tech output
Show tech output available at 0/RP0/CPU0 :
/harddisk:/showtech/showtech-custom-2021-Oct-28.174339.UTC.tgz
++ Show tech end time: 2021-Oct-28.174540.UTC ++
```

Simplified Tech-Support Data Collection

Table 47: Feature History Table

Feature Name	Release	Description
Collect Comprehensive Tech-Support Information	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I
Supporting Custom Profile show tech command	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I
Supporting Custom Profile show tech command	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

Feature Name	Release	Description
Collect Comprehensive Tech-Support Information	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
Collect Comprehensive Tech-Support Information	Release 24.2.11	You can now collect a comprehensive list of troubleshooting data and restore network operations quickly in case of a network disruption. This release allows you to collect more tech-support data than you could in previous releases by executing the **show tech-support custom** command.

Feature Name	Release	Description
Supporting Custom Profile show tech command	Release 24.2.11	We have simplified the process to collect technical support information for traffic, control-plane, and system by consolidating multiple commands for each of these parameters under the following options: • **traffic** - Generates tech-support information related to network traffic. • **control-plane** - Generates tech-support information related to the control-plane. • **system** - Generates tech-support information related to the system (router). This release adds the keywords **traffic**, **control-plane**, and **system** to the **show tech-support custom** command.

When diagnosing a potential problem in the network or when collecting information related to a specific area such as network traffic, control-plane, or the system, you no longer need to run several show commands or show tech-support commands. Instead, you can run a minimum set of CLIs to generate or gather entire tech-support data related to a specific area. Data is collected from all Route Processors (RPs), Line Cards (LCs), and Fabric Cards (FCs) on the router.

The **show tech-support custom traffic** command generates tech-support information related to network traffic, the **show tech-support custom control-plane** command generates tech-support information related to the control-plane, and the **show tech-support custom system** command generates tech-support information related to the system (router).

Cisco IOS XR Release 24.2.11 collects a comprehensive list of tech-support data in addition to providing internal checks to ensure the router does not collect the same command output more than once.

Note The command execution may take a few to several minutes. Moreover, the output is usually large and varies in size depending on the number of LCs on the router. So, before executing the CLI command, ensure that you have enough disk space available.

Guidelines for Gathering Technical Support Information Using Custom Options

The guidelines for gathering technical support information are as follows:

- We do not recommend parallel execution of the **show tech-support custom** command with different options.

- We do not recommend using **show tech-support custom** command in automated scripts.
- We do not recommend executing the **show tech-support custom** command at the same time as PAM (Platform Automated Monitoring), EEM (Embedded Event Manager), EDCD (Event-Driven CLI Database) scripts.

Example

The following codeblocks show the **show tech-support custom** command options generating tech-support information. By default, the output of this command is saved on the hard disk of the router in a file with *.tgz* extension. For example, *`/harddisk:/showtech/name.tgz`*.

```
Router# show tech-support custom traffic
++ Show tech start time: 2023-Jun-16.195852.UTC ++
Fri Jun 16 19:58:52 UTC 2023 Waiting for gathering to complete
..........................................................................................
Fri Jun 16 20:05:45 UTC 2023 Compressing show tech output
Show tech output available at 0/RP0/CPU0 :
/harddisk:/showtech/showtech-custom-traffic-2023-Jun-16.195852.UTC.tgz
++ Show tech end time: 2023-Jun-16.200546.UTC ++
```

```
Router# show tech-support custom system
Fri Jan 12 11:57:04.991 UTC
++ Show tech start time: 2024-Jan-12.115709.UTC ++
Fri Jan 12 11:57:09 UTC 2024 Waiting for gathering to complete
..........................................................................................
Fri Jan 12 12:09:39 UTC 2024 Compressing show tech output
Show tech output available at 0/RP0/CPU0 :
/harddisk:/showtech/showtech-custom-system-2024-Jan-12.115709.UTC.tgz
++ Show tech end time: 2024-Jan-12.120946.UTC ++
```

```
Router# show tech-support custom control
Fri Jan 12 13:21:48.433 UTC
++ Show tech start time: 2024-Jan-12.132153.UTC ++
Fri Jan 12 13:21:53 UTC 2024 Waiting for gathering to complete
..........................................................................................
Fri Jan 12 13:28:37 UTC 2024 Compressing show tech output
Show tech output available at 0/RP0/CPU0 :
/harddisk:/showtech/showtech-custom-control-2024-Jan-12.132153.UTC.tgz
++ Show tech end time: 2024-Jan-12.132842.UTC ++
```

CHAPTER 12

Online Diagnostics for NPU

Cisco 8000 Series Routers support the Online Diagnostics feature that enables you to run tests to verify the hardware functionality when connected to a live network. When a problem is detected, diagnostic test results help in isolating the location of the problem, enabling you to take appropriate measures to resolve the issue in less time.

Table 48: Feature History Table

Feature Name	Release	Description
Online diagnostics for NPU, NPU slices, and fabric cards	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100, K100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 8712-MOD-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
Online diagnostics for NPU slices and fabric cards	Release 24.2.11	Introduced in this release on: Fixed Systems (8200); Centralized Systems (8600); Modular Systems (8800 [LC ASIC: Q100, Q200]) You can now use the online diagnostics functionality to test the health of fabric cards and all the slices in an NPU. This feature can help you detect fabric, and slice level failures.

Feature Name	Release	Description
Online diagnostics for NPU	Release 7.5.2/Release 7.3.5	You can now use the online diagnostic feature to verify if the router NPUs are operational. NPU failure logs are captured in the system log output. You can also generate tech support information that is useful for Cisco Technical Support representatives when troubleshooting a router. This feature introduces the following commands: • **diagnostic monitor interval** • **diagnostic monitor location disable** • **diagnostic monitor syslog** • **diagnostic monitor threshold** • **show diagnostic trace location** • **show diagnostic result**

Online Diagnostics for NPU

The diagnostic tests check different hardware components in a system and verify the data paths and control signals. The online diagnostics tests use the CPU to send packets to the Network Processing Unit (NPU) through the Punt switch. If a failure is detected, an NP Datalog is automatically generated to help diagnose the problem.

The default interval for the NPU loopback test is one minute, and the default threshold is 3.

The following is a sample system log output:

```
LC/0/6/CPU0:Oct 30 18:52:37.737 UTC: 8000_online_diag[123]: %DIAG-DIAG-3-GOLDXR_FAIL :
SFNPULoopback: Online diagnostic packet drops detected on NPU 0, slice 0. Please collect
"show tech-support online-diags".
LC/0/6/CPU0:Oct 30 18:52:37.737 UTC: 8000_online_diag[123]: %DIAG-DIAG-3-GOLDXR_FAIL : Use
 "show diagnostic result location <location> detail" to monitor online diagnostic results.
```

Online diagnostic tests can be categorized based on the way they are executed. They are the following:

Types of Online diagnostic test	Description
Dynamic diagnostics	Online Diagnostics are enabled when the system starts and the system datapath is operational. When the system is in use and linked to a live network, these tests run in the background as a non-disruptive test.
On-demand diagnostics	Tests that are conducted as needed using a diagnostic start command from the command-line interface (CLI). These tests are useful when a hardware fault is suspected. You can use these diagnostics tests to determine the status and troubleshoot the hardware issues.

Online diagnostics for NPU slices

From Release 24.2.11, the online diagnostics functionality is extended to test all the slices in an NPU, enabling you to detect slice level failures. The default rate at which the test packets are transmitted to each NPU slice is increased to 60 packets per minute.

The default interval is one second for the per-slice test, and the default threshold is 3.

The following is a sample system log output:

```
LC/0/6/CPU0:Oct 30 18:52:37.737 UTC: 8000_online_diag[271]: %DIAG-DIAG-3-GOLDXR_FAIL :
SFNPUSlice: Online diagnostic packet drops detected on NPU 0, slice 0. Please collect "show
 tech-support online-diags".
LC/0/6/CPU0:Oct 30 18:52:37.737 UTC: 8000_online_diag[271]: %DIAG-DIAG-3-GOLDXR_FAIL : Use
 "show diagnostic result location <location> detail" to monitor online diagnostic results.
```

Online diagnostics for fabric cards

From Release 24.2.11, you can also test the health of fabric cards (FC) using the online diagnostics functionality. The test packets are transmitted to fabric cards at the default rate of 50 packets per minute.

The default interval for the fabric test is 30 seconds, and the default threshold is 6.

The following is a sample system log output:

```
LC/0/6/CPU0:Oct 30 18:52:37.737 UTC: npu_drvr[283]: %FABRIC-NPU_DRVR-4-FABRIC_ONLINE_DIAG_FAIL
 : Online diagnostic packet drops detected on NPU 0, slice 0. Please collect "show
tech-support online-diags".
LC/0/6/CPU0:Oct 30 18:52:37.737 UTC: npu_drvr[283]: %FABRIC-NPU_DRVR-4-FABRIC_ONLINE_DIAG_FAIL
 : Use "show diagnostic result location <location> detail" to monitor online diagnostic
results.
```

Data plane health check utility

Data plane health check utility is a monitoring tool that helps you determine the health of the data plane components including

- fabric cards
- NPUs, and

- lookup engines

This utility can detect fabric memory corruption and packet loss that may happen due to broken internal links.

Table 49: Feature History Table

Feature Name	Release	Description
Lookup engine test in data plane health check utility	Release 25.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q100, Q200]); Centralized Systems (8600 [ASIC:Q200]) ; Modular Systems (8800 [LC ASIC: Q100, Q200]) This feature detects mis-routed packets due to hardware defects causing bit corruption in route prefix lookups. This feature implements a new lookup engine test mode in the data plane health check utility to detect such mis-routed packets. This test requires a router reload upon completion. This feature introduces these changes: **CLI:** Two new keywords, **lookup** and **all-modules**, are added to the **monitor dataplane-health module** command.
Monitor data plane health	Release 25.3.1	Introduced in this release on: Fixed Systems (8700 [ASIC:K100], 8010 [ASIC: A100])(select variants only*) *This feature is now supported on: • 8011-4G24Y4H-I • 8712-MOD-M

Feature Name	Release	Description
Monitor data plane health	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
Monitor data plane health	Release 24.2.11	Introduced in this release on: Fixed Systems (8200); Modular Systems (8800 [LC ASIC: Q100, Q200]) You can now easily detect fabric memory corruption and packet loss by checking the health of data plane components including fabric and NPUs on a distributed system using our on-demand diagnostic utility. This functionality introduces the following commands: • **monitor dataplane-health** • **show dataplane-health status**

You can start the diagnosis using the CLI command **monitor dataplane-health**. The detailed error report helps you identify the faulty card.

Starting from IOS XR Release 25.4.1, the data plane health check utility introduces a new lookup engine test mode to detect packets routed to wrong interfaces using the **monitor dataplane-health module lookup** command. The test analyzes the packet statistics in your network and reports hardware failure for dropped packets indicating a traffic loss caused by corruption in the route prefix lookup. A router reload is required after this test is completed. Also, to run the health check for all modules namely the fabric path, NPU path, and lookup engine, use the **monitor dataplane-health module all-modules**.

You can also use the **show dataplane-health status** command to check the status of a data plane health test. It provides information on whether the test is still running or if it's completed, along with a summary of the results.

Note Do not use the data plane health check utility on a router that carries live traffic, as this utility affects the system performance.

Use cases for data plane health check

You can use the data plane health check utility in the following scenarios:

- Before router deployment – After installing the FC or LC on the router, you can run the utility to check for issues, and then proceed to router provisioning.
- After router deployment – If traffic loss is observed, but the packet drop analysis does not provide a root cause, then isolate the router and run the utility to check for issues.

Limitations for data plane health check

- Avoid using the **show controllers npu debugshell** CLI command.
- Avoid system reload (or LC reload) when the Data Plane Health Check utility is being executed.
- The **monitor dataplane-health module fabric** command is supported only on distributed routers.
- You must archive the report file before subsequent runs, as this file is overwritten on re-execution of the command.

  The previous log file is archived as `/harddisk:/dph_mon/dataplane_health_fabric_mode_report.txt.bak`.

  The test report is available at `/harddisk:/dph_mon/dataplane_health_fabric_mode_report.txt`.

Monitor and verify data plane health

This example shows how to execute the data plane health check utility for fabric module.

```
RP/0/RP0/CPU0:Router# monitor dataplane-health
Wed Feb 28 15:08:15.659 EST
RP/0/RP0/CPU0:Feb 28 15:08:15.687 EST: dph_mon_bg[337]: %PLATFORM-DPH_MONITOR-6-STARTED :
Dataplane health monitoring started. Please check
harddisk:/dph_mon/dataplane_health_fabric_mode_report.txt for details of the result.
THIS COMMAND IMPACTS SYSTEM PERFORMANCE AND SHOULD IDEALLY BE RUN ON A ROUTER THAT IS
ISOLATED.
Progressive details of the test logged in
/harddisk:/dph_mon/dataplane_health_fabric_mode_report.txt
Previous log file is archived as
/harddisk:/dph_mon/dataplane_health_fabric_mode_report.txt.bak
Please save the archive with a different file name as needed
########################################################################################
Module:fabric
Patterns used: 0xf0,0x0f,0x00,0x55,0xff,
Duration per pattern: 10 seconds
Pause time between each slice/pattern test: 2 seconds
Best effort time for test completion: 1044 seconds
Depending on the overall dataplane health state, it may take additional time to complete
```

```
########################################################################################
Dataplane health monitoring will run in the background
Wait for completion log message from the process "dph_mon_bg"
   OR
Use "show dataplane-health status" regularly to check for completion

RP/0/RP0/CPU0:Feb 28 15:08:15.687 EST: dph_mon_bg[337]: %PLATFORM-DPH_MONITOR-6-STARTED :
Dataplane health monitoring started. Please check
harddisk:/dph_mon/dataplane_health_fabric_mode_report.txt for details of the result.

RP/0/RP0/CPU0:Feb 28 15:28:54.965 EST: dph_mon_bg[337]: %PLATFORM-DPH_MONITOR-6-COMPLETED
: Dataplane health monitoring completed. Please check
harddisk:/dph_mon/dataplane_health_fabric_mode_report.txt for details of the result.
```

This example shows how to run fabric, NPU, and lookup engine tests together to check the health of these modules. The test includes a warning to reload the router after the test completion.

Warning Run the test on a router in a non-production environment as this test impacts the system performance.

```
RP/0/RP0/CPU0:Router# monitor dataplane-health module all-modules
THIS COMMAND IMPACTS SYSTEM PERFORMANCE AND SHOULD IDEALLY BE RUN ON A ROUTER THAT IS
ISOLATED.
THIS COMMAND INCLUDES THE TESTING OF THE LOOKUP ENGINE MODULE.
TESTS ON THE LOOKUP ENGINE REQUIRE A RELOAD AFTER TEST COMPLETION.
Progressive details of the test logged in
/harddisk:/dph_mon/dataplane_health_fabric_mode_report.txt
Previous log file is archived as
/harddisk:/dph_mon/dataplane_health_fabric_mode_report.txt.bak
Please save the archive with a different file name as needed
########################################################################################
Module:fabric, lookup
Patterns used: 0xf0,0x55,0xff,
Duration per pattern: 10 seconds
Pause time between each slice/pattern test: 2 seconds
Best effort time for test completion: 492 seconds
Depending on the overall dataplane health state, it may take additional time to complete
########################################################################################
Dataplane health monitoring will run in the background
Wait for completion log message from the process "dph_mon_bg"
            OR
Use "show dataplane-health status" regularly to check for completion
RP/0/RP0/CPU0:Router#
```

Note Make sure to reload the router once the test is completed.

This example shows how to run lookup engine test. The test includes a warning to reload the router after the test completion.

Warning Run the test on a router in a non-production environment as this test impacts the system performance.

```
RP/0/RP0/CPU0:ios# monitor dataplane-health module lookup
Fri Oct 24 13:09:20.906 UTC
THIS COMMAND IMPACTS SYSTEM PERFORMANCE AND SHOULD IDEALLY BE RUN ON A ROUTER THAT IS
```

```
ISOLATED.
THIS COMMAND INCLUDES THE TESTING OF THE LOOKUP ENGINE MODULE.
TESTS ON THE LOOKUP ENGINE REQUIRE A RELOAD AFTER TEST COMPLETION.
Progressive details of the test logged in
/harddisk:/dph_mon/dataplane_health_lookup_mode_report.txt
Previous log file is archived as
/harddisk:/dph_mon/dataplane_health_lookup_mode_report.txt.bak
Please save the archive with a different file name as needed
########################################################################################
Module:lookup
Best effort time for test completion: 32 seconds
Depending on the overall dataplane health state, it may take additional time to complete
########################################################################################
Dataplane health monitoring will run in the background
Wait for completion log message from the process "dph_mon_bg"
            OR
Use "show dataplane-health status" regularly to check for completion
```

Note Make sure to reload the router once the test is completed.

These examples show how to check the status of a data plane health test:

Example 1: Output with data plane health check in progress

```
RP/0/RP0/CPU0:Router# show dataplane-health status
Mon Jan 29 22:39:48.336 UTC
Dataplane health monitoring in progress..
```

Example 2: Output with successful data plane health check

```
RP/0/RP0/CPU0:Router# show dataplane-health status
Mon Jan 29 23:10:21.564 UTC
Dataplane health monitoring completed
Summary of results (Module: fabric):
############################################################################
Output summary legend:
ERROR: Tests were not run for this slice due to some errors
GOOD: Tests were successful for this slice
LOSS: Packet loss was observed for this slice
CORRUPT: Packet corruption was observed for this slice
############################################################################
  LC    NP     Slice        GOOD        LOSS     CORRUPT       ERROR
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
   0     0         0     2526099           0           0           0
                   1     2526856           0           0           0
                   2     2526529           0           0           0
         1         0     2526590           0           0           0
                   1     2526918           0           0           0
                   2     2526421           0           0           0
         2         0     2526665           0           0           0
                   1     2525818           0           0           0
                   2     2526286           0           0           0
-------------------------------------------------------------------------------
   1     0         0     2526754           0           0           0
                   1     2526328           0           0           0
                   2     2526695           0           0           0
         1         0     2525892           0           0           0
                   1     2526988           0           0           0
                   2     2526215           0           0           0
**********************************************************************************
```

```
DATAPATH CHECK IS CLEAN (mode: fabric).
**********************************************************************************
```

Example 3: Output with failures

```
RP/0/RP0/CPU0:Router# show dataplane-health status
Wed Jan 10 21:37:01.601 UTC
Dataplane health monitoring completed
Summary of results (Module: fabric):
############################################################################
Output summary legend:
ERROR: Tests were not run for this slice due to some errors
GOOD: Tests were successful for this slice
LOSS: Packet loss was observed for this slice
CORRUPT: Packet corruption was observed for this slice
############################################################################
  LC    NP     Slice        GOOD        LOSS     CORRUPT       ERROR
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
   1     0         0     2526253           0           0           0
                   1     2527136           0           0           0
                   2     2526235           0           0           0
         1         0     2527166           0           0           0
                   1     2527217           0           0           0
                   2     2526424           0           0           0
-------------------------------------------------------------------------------
   2     0         0     2526733           0           0           0
                   1     2526948           0           0           0
                   2     2526554           0           0           0
         1         0     2526294           0           0           0
                   1     2526220           0           0           0
                   2     2526085           0           0           0
-------------------------------------------------------------------------------
   3     0         0     2525876           0           0           0
                   1     2526642           0           0           0
                   2     2525957           0           0           0
         1         0     2526491           0           0           0
                   1     2526263           0           0           0
                   2     2526200           0           0           0
         2         0     2526804           0           0           0
                   1     2526135           0           0           0
                   2     2526328           0           0           0
-------------------------------------------------------------------------------
   4     0         0      493934           0       11501           0
                   1           0           0           0           0
                   2           0           0           0           0
         1         0      493605           0       11591           0
                   1           0           0           0           0
                   2           0           0           0           0
-------------------------------------------------------------------------------
   5     0         0      505389           0          30           0
                   1           0           0           0           0
                   2           0           0           0           0
         1         0      505358           0          23           0
                   1           0           0           0           0
                   2           0           0           0           0
-------------------------------------------------------------------------------
   6     0         0     2526307           0           0           0
                   1     2525905           0           0           0
                   2     2526142           0           0           0
         1         0     2526755           0           0           0
                   1     2526603           0           0           0
                   2     2526607           0           0           0
**********************************************************************************
```

```
Corruption detected:(LC4/0 <-> FC2/0) (LC4/1 <-> FC2/0) (LC5/0 <-> FC3/0) (LC5/1 <-> FC3/0)


**********************************************************************************
FAILURES DETECTED IN DATAPATH for fabric mode.
Please run "monitor dataplane-health module no-fabric"
Please check /harddisk:/dph_mon/dataplane_health_fabric_mode_report.txt
**********************************************************************************
```

Example 4: Output for health check with all-modules option

```
RP/0/RP0/CPU0:Router# show dataplane-health status
Dataplane health monitoring completed

Datapath test on all requested LC/NPU/slice completed
Summary of results (Module: fabric/lookup):
####################################################################################
Output summary legend:
ERROR: Tests were not run for this slice due to some errors
GOOD: Tests were successful for this slice
LOSS: Packet loss was observed for this slice
CORRUPT: Packet corruption was observed for this slice
LE-FAULTS: Lookup Engine test results - On supported devices, 0 for no-failures,
           or a bit mask of faulty engines; n/a on unsupported devices

###################################################################################
-----------------------------------------------------------------------------------
  LC     NP     Slice        GOOD        LOSS     CORRUPT       ERROR      LE-FAULTS
-----------------------------------------------------------------------------------
-----------------------------------------------------------------------------------
 LC4     0         0     1663101           0           0           0        0x0000
                   1     1661542           0           0           0        0x0200
                   2     1662258           0           0           0        0x0000
         1         0     1667547           0           0           0        0x0000
                   1     1661658           0           0           0        0x0000
                   2     1661153           0           0           0        0x0000
         2         0     1661087           0           0           0        0x0000
                   1     1661474           0           0           0        0x0000
                   2     1660756           0           0           0        0x0000
         3         0     1661936           0           0           0        0x0000
                   1     1661197           0           0           0        0x0000
                   2     1661644           0           0           0        0x0000
****************************************************************************#******
DATAPATH CHECK IS CLEAN, LOOKUP ENGINE ERROR(s) DETECTED (mode: fabric/lookup).
***********************************************************************************
```

Example 5: Output for health check with lookup option

```
RP/0/RP0/CPU0:Router# show dataplane-health status

Thur Nov 20 18:06:25.132 UTC
Dataplane health monitoring completed

Datapath test on all requested LC/NPU/slice completed
Summary of results (Module: lookup):
###################################################################################
Output summary legend:
LE-FAULTS: Lookup Engine test results - On supported devices, 0 for no-failures,
           or a bit mask of faulty engines; n/a on unsupported devices

##################################################################################
----------------------------------------------------------------------------------
  LC    NP     Slice    LE-FAULTS
---------------------------------
```

```
----------------------------------
   4     0         0       0x0000
                   1       0x0200
                   2       0x0000
         1         0       0x0000
                   1       0x0000
                   2       0x0000
         2         0       0x0000
                   1       0x0000
                   2       0x0000
         3         0       0x0000
                   1       0x0000
                   2       0x0000

******************************************************************************
LOOKUP ENGINE ERROR(s) DETECTED (mode: lookup).
******************************************************************************
```

Example 6: Output for health check with lookup option on unsupported devices

```
RP/0/RP0/CPU0:Router# show dataplane-health status

Wed Feb 11 20:13:45.429 UTC
Dataplane health monitoring completed

Datapath test on all requested LC/NPU/slice completed
Summary of results (Module: lookup):
###################################################################################
Output summary legend:
LE-FAULTS: Lookup Engine test results - On supported devices, 0 for no-failures,
           or a bit mask of faulty engines; n/a on unsupported devices
###################################################################################
-----------------------------------------------------------------------------------
  LC    NP     Slice   LE-FAULTS
-----------------------------------------------------------------------------------
 RP0     0         0         n/a
                   1         n/a
************************************************************************************

************************************************************************************
DATAPATH CHECK IS CLEAN (mode: lookup).
************************************************************************************
```

Note If the data plane-health monitor does not report any issues, it suggests that the system is functioning correctly, and there is no need to proceed with further data plane health verification.

If any failure is detected during the data plane health check, you must proceed with additional verification.

Additional Troubleshooting

The following is a sample output that illustrates failures that are detected in the datapath.

```
RP/0/RP0/CPU0:Router# monitor dataplane-health

Fri Sep 29 12:53:51.595 UTC

THIS COMMAND IMPACTS SYSTEM PERFORMANCE AND SHOULD IDEALLY BE RUN ON A ROUTER THAT IS
ISOLATED.
```

```
Details of the test results are logged in harddisk:/dataplane_health_detail_report.txt

Estimated time for completion: 783 seconds

Ensure that the terminal/vty session timeout is greater than 783 seconds

Testing in progress (suggest not to break the tests)

....................................................................................................................................................................................


Processing further to find failed fabric elements. This will take more time...

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Datapath test on all requested LC/NPU/slice completed

Summary of results:

############################################################################

Output summary legend:

ERROR: Tests were not run for this slice due to some errors

GOOD: Tests were successful for this slice

LOSS: Packet loss was observed for this slice

CORRUPT: Packet corruption was observed for this slice

############################################################################

-------------------------------------------------------------------------------

  LC    NP     Slice        GOOD        LOSS     CORRUPT       ERROR

-------------------------------------------------------------------------------

-------------------------------------------------------------------------------

   1     0         0      476214           0         637           0

                   1           0           0           0           0

                   2           0           0           0           0

         1         0      475860           0         670           0

                   1           0           0           0           0

                   2           0           0           0           0

-------------------------------------------------------------------------------

   2     0         0     2383553           0           0           0

                   1     2383747           0           0           0

                   2     2383616           0           0           0

         1         0     2383280           0           0           0

                   1     2383737           0           0           0
                   2     2383343           0           0           0

         2         0     2383937           0           0           0

                   1     2383913           0           0           0

                   2     2384017           0           0           0

**********************************************************************************

Corruption detected:(LC1/0 <-> FC7/0) (LC1/1 <-> FC7/0)
**********************************************************************************

FAILURES DETECTED IN DATAPATH.

Please run "monitor dataplane-health module no-fabric"

Please check harddisk:/dataplane_health_detail_report.txt

**********************************************************************************

RP/0/RP0/CPU0:Router# monitor dataplane-health
Fri Feb 16 08:50:58.115 UTC
THIS COMMAND IMPACTS SYSTEM PERFORMANCE AND SHOULD IDEALLY BE RUN ON A ROUTER THAT IS
ISOLATED.
Progressive details of the test logged in
/harddisk:/dph_mon/dataplane_health_fabric_mode_report.txt
########################################################################################
Module:fabric
Patterns used: 0xf0,0x0f,0x00,0x55,0xff,
Duration per pattern: 10 seconds
Pause time between each slice/pattern test: 2 seconds
Best effort time for test completion: 522 seconds
Depending on the overall dataplane health state, it may take additional time to complete
########################################################################################
Dataplane health monitoring will run in the background
Wait for completion log message from the process "dph_mon_bg"
                        OR
Use "show dataplane-health status" regularly to check for completion

RP/0/RP0/CPU0:Router# show dataplane-health status
Fri Feb 16 09:04:12.156 UTC
Dataplane health monitoring completed
Summary of results (Module: fabric):
############################################################################
Output summary legend:
ERROR: Tests were not run for this slice due to some errors
GOOD: Tests were successful for this slice
LOSS: Packet loss was observed for this slice
CORRUPT: Packet corruption was observed for this slice
############################################################################
  LC    NP     Slice        GOOD        LOSS     CORRUPT       ERROR
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
   0     0         0          85       10977           0           0
                   1           0           0           0           0
                   2           0           0           0           0
         1         0        1735        8865           0           0
                   1           0           0           0           0
                   2           0           0           0           0
**********************************************************************************
```

```
*********************************************************************************
FAILURES DETECTED IN DATAPATH (mode: fabric).
Please run "monitor dataplane-health module no-fabric" to check if the issue is on the LCs
 or FCs
Please check /harddisk:/dph_mon/dataplane_health_fabric_mode_report.txt
*********************************************************************************
```

The sample output indicates that there are failures detected in the datapath. To further isolate the issue, verify if the packet corruption is caused by a fabric card or line card. Execute the following command to run health check on the line cards excluding the fabric card.

```
RP/0/RP0/CPU0:Router# monitor dataplane-health module no-fabric

Fri Sep 29 14:09:28.506 UTC

THIS COMMAND IMPACTS SYSTEM PERFORMANCE AND SHOULD IDEALLY BE RUN ON A ROUTER THAT IS
ISOLATED.

Details of the test results are logged in harddisk:/dataplane_health_detail_report.txt

Estimated time for completion: 783 seconds

Ensure that the terminal/vty session timeout is greater than 783 seconds

Testing in progress (suggest not to break the tests)

..................................................................................................................................................................................................................


Datapath test on all requested LC/NPU/slice completed

Summary of results:

############################################################################

Output summary legend:

ERROR: Tests were not run for this slice due to some errors

GOOD: Tests were successful for this slice

LOSS: Packet loss was observed for this slice

CORRUPT: Packet corruption was observed for this slice

############################################################################

-------------------------------------------------------------------------------

  LC     NP     Slice        GOOD        LOSS     CORRUPT       ERROR

-------------------------------------------------------------------------------

-------------------------------------------------------------------------------

   1      0         0     2383412           0           0           0

                    1     2383031           0           0           0

                    2     2383484           0           0           0

          1         0     2383883           0           0           0

                    1     2383973           0           0           0
```

```
                   2     2383349           0           0           0

-------------------------------------------------------------------------------

   2     0         0     2383160           0           0           0

                   1     2384196           0           0           0

                   2     2383879           0           0           0

         1         0     2383135           0           0           0

                   1     2383196           0           0           0

                   2     2383668           0           0           0

         2         0     2383414           0           0           0

                   1     2384360           0           0           0

                   2     2383732           0           0           0

-------------------------------------------------------------------------------

   6     0         0     2383933           0           0           0

                   1     2384205           0           0           0

                   2     2383746           0           0           0

         1         0     2383215           0           0           0

                   1     2383578           0           0           0

                   2     2382921           0           0           0

**********************************************************************************

 DATAPATH CHECK IS CLEAN.

**********************************************************************************

RP/0/RP0/CPU0:Router# monitor dataplane-health module no-fabric
Fri Feb 16 09:08:39.412 UTC
THIS COMMAND IMPACTS SYSTEM PERFORMANCE AND SHOULD IDEALLY BE RUN ON A ROUTER THAT IS
ISOLATED.
Progressive details of the test logged in
/harddisk:/dph_mon/dataplane_health_no_fabric_mode_report.txt
########################################################################################
Module:no-fabric
Patterns used: 0xf0,0x0f,0x00,0x55,0xff,
Duration per pattern: 10 seconds
Pause time between each slice/pattern test: 2 seconds
Best effort time for test completion: 522 seconds
Depending on the overall dataplane health state, it may take additional time to complete
########################################################################################
Dataplane health monitoring will run in the background
Wait for completion log message from the process "dph_mon_bg"
                        OR
Use "show dataplane-health status" regularly to check for completion

RP/0/RP0/CPU0:Router# show dataplane-health status
Fri Feb 16 09:11:01.540 UTC
```

```
Dataplane health monitoring completed
Summary of results (Module: no-fabric):
############################################################################
Output summary legend:
ERROR: Tests were not run for this slice due to some errors
GOOD: Tests were successful for this slice
LOSS: Packet loss was observed for this slice
CORRUPT: Packet corruption was observed for this slice
############################################################################
  LC    NP     Slice        GOOD        LOSS     CORRUPT       ERROR
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
   0     0         0       4836          0           0           0
                   1       4908          0           0           0
                   2       5634          0           0           0
         1         0       4196          0           0           0
                   1       7638          0           0           0
                   2       7362          0           0           0
**********************************************************************************
DATAPATH CHECK IS CLEAN..

**********************************************************************************
```

The above sample output indicates that there are no errors or corruption on the line cards, and hence the fabric card must be faulty.

If there is any corruption detected after running the **monitor dataplane-health module no-fabric** command, then contact the Cisco Technical Assistance Center (TAC).

If packet loss is detected after running the **monitor dataplane-health module** command, perform the following steps for further verification:

- Run the following command:

```
RP/0/RP0/CPU0:Router# show controllers npu driver location all

Fri Sep 29 13:11:16.738 EDT

==============================================

NPU Driver Information

==============================================

Driver Version: 1

SDK Version: 1.55.0.41


Functional role: Active,     Rack: 8808, Type: lcc, Node: 0/5/CPU0

Driver ready      : Yes

NPU first started : Fri Sep 29 08:11:58 2023

Fabric Mode: FABRIC/8FC

NPU Power profile: Medium

Driver Scope: Node

Respawn count     : 1
```

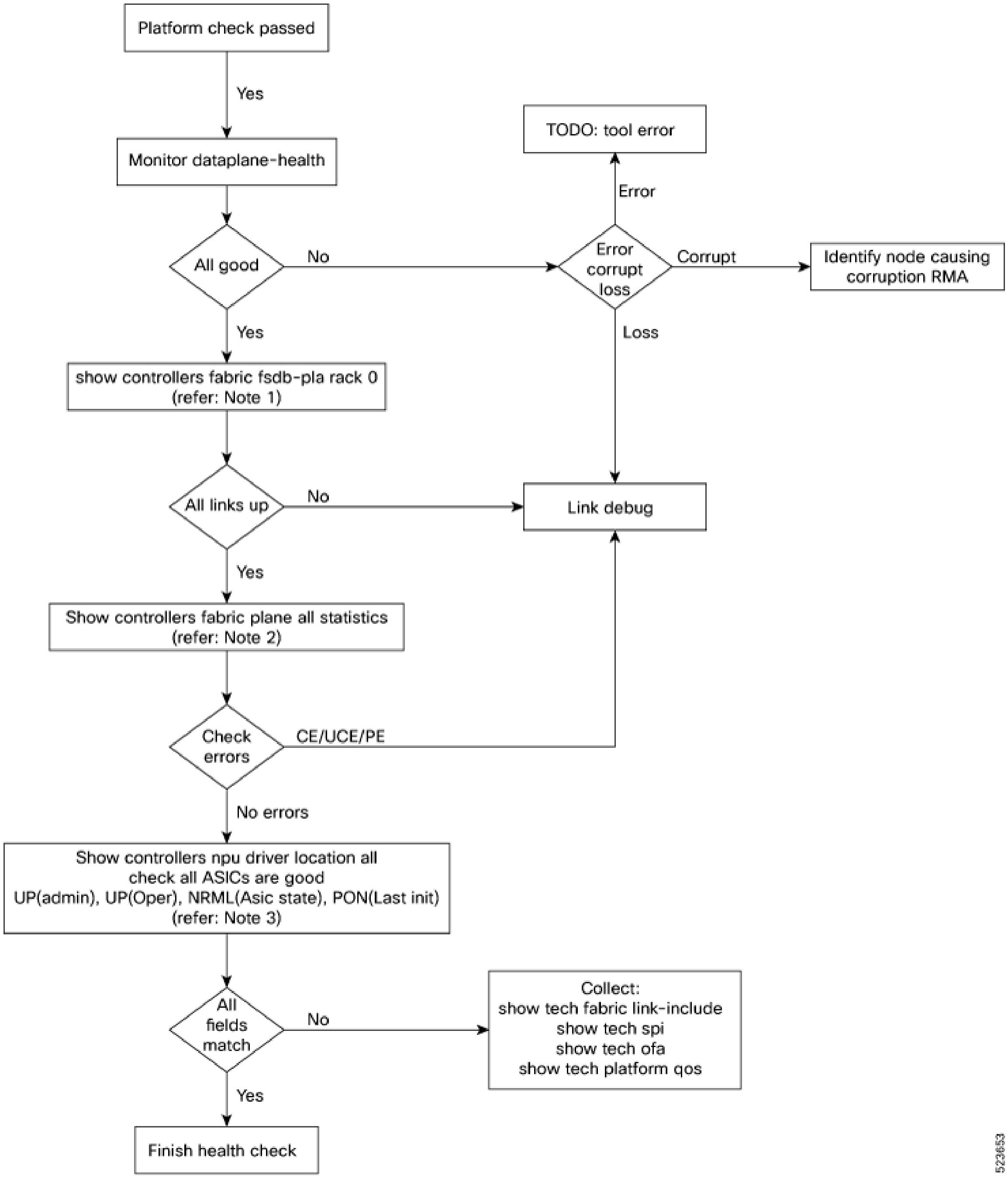
```
Availablity masks :

        card: 0x1,     asic: 0x7,    exp asic: 0x7

Weight distribution:

        Unicast: 80,      Multicast: 20

+----------------------------------------------------------------+

| Process | Connection | Registration | Connection | DLL         |

| /Lib    | status     | status       | requests   | registration|

+----------------------------------------------------------------+

| FSDB    | Active     | Active       |           1|  n/a        |

| FGID    | Inactive   | Inactive     |           0|  n/a        |

| AEL     | n/a        | n/a          |         n/a|  Yes        |

| SM      | n/a        | n/a          |         n/a|  Yes        |

+----------------------------------------------------------------+


Asics :

HP - HotPlug event, PON - Power On reset

HR - Hard Reset,    WB  - Warm Boot

+------------------------------------------------------------------------------+

| Asic inst. | fap|HP|Slice|Asic|Admin|Oper | Asic state | Last |PON|HR |  FW  |

|  (R/S/A)   | id |  |state|type|state|state|            | init |(#)|(#)|  Rev |

+------------------------------------------------------------------------------+

| 0/5/0      |  20| 1| UP  |npu | UP  | UP  |NRML        |HPON   |  1|  0|0x0000|

| 0/5/1      |  21| 1| UP  |npu | UP  | UP  |NRML        |PON   |  1|  0|0x0000|

| 0/5/2      |  22| 1| UP  |npu | UP  | UP  |NRML        |PON   |  1|  0|0x0000|

+------------------------------------------------------------------------------+
......
.......
```

HPON flag indicates an error. Collect logs, and contact Cisco TAC.

Troubleshooting Flowchart

If a HPON error is detected, follow the steps provided in the following troubleshooting flowchart:

Figure 9: Troubleshooting Flowchart



Link Debugging

Identify Reachability Issues

When the datapath monitoring tool reports packet loss, if the status of all ASICs is normal, and if you have not done hard resets, perform the following troubleshooting steps:

- To identify issues in the connection between the line card and fabric cards, check the fabric links using the **show controllers fabric fsdb-pla rack 0** command:

```
RP/0/RP0/CPU0:Router# show controllers fabric fsdb-pla rack 0
Fri Feb  2 05:59:07.624 UTC
Description:
   planes       : p0-p7
   plane mask   : Asic #0-3
   Asic value  1: destination reachable via asic
               .: destination unreachable via asic
               x: asic not connected to LC (for S3)
               -: plane not configured (for S2) or asic missing
Rack: 0, Stage: s123
=============================
Destination   p0     p1     p2     p3     p4     p5     p6     p7     Reach-mask  Oper
 Up
Address       mask   mask   mask   mask   mask   mask   mask   mask   links/asic
links/asic
Fapid(R/S/A)  0123   0123   0123   0123   0123   0123   0123   0123  Mn/Mx Total Mn/Mx
  Total
--------------------------------------------------------------------------------------
0(0/0/0)      11     11     --     --     --     --     --     --     6/6    22  12/12
     48
1(0/1/0)      11     11     --     --     --     11     --     --     4/6    22  10/12
     48
4(0/7/0)      ..     ..     --     ..     --     ..     --     ..     0/0    0   12/12
     48
```

The above sample output indicates that the line cards in slot 0, and 1 do not have any connectivity issues with fabric cards. However, the "**..**" state for line card in slot 7 indicates that links are down between the line card and fabric card pair.

In such situations, perform the following troubleshooting steps:

1. Remove the line card from the particular slot (for example, slot 7) on the front panel.
2. Check the particular backplane connector (for example, connector 4) for any bent pins.
3. Check the FC connector (for example, FC3 connector 5) for any damage.
   - If there are no bent pins or FC connector damages, reinsert the line card, and run the **show controller fabric fsdb-pla rack 0** command again. Check if the status displays all "**11**" and not "**..**".
   - If you notice bent pins on the line card connector, capture it in a photo, and open a Cisco Support (TAC) case for Return Material Authorization (RMA).
   - If you notice any damage on the FC connector, capture it in a photo, and open a Cisco Support (TAC) case for RMA.

- If some links are down even after the visual inspection of fabric links, collect logs and contact Cisco TAC.

Note In some scenarios, the connectivity between line card ASICs and fabric card ASICs can be UP, but few links between the cards can be down. So, it is essential to verify the link status.

- To check the status of links between the line card and fabric cards, examine the **Reach-mask links/asic min/max** column output of the **show controllers fabric fsdb-pla rack 0** command. In the previous sample output, you can infer that for LC1, two links out of six links are down due to link connectivity issues.

  If some links are down even after the visual inspection of fabric links, collect logs and contact Cisco TAC.

Note 1

If the min and max values in the output under **links/asic min/max** are not equal, that means the links between the LC ASIC and FC ASIC are down.

- Run the following command to get more details about the links status:

```
RP/0/RP0/CPU0:Router# show controllers npu link-info rx 0 255 topo instance
 all location all | ex EN/DN | ex NC
Fri Feb  2 06:59:18.003 UTC

Node ID: 0/2/CPU0
-----------------------------------------------------------------------------
Link ID    Log  Link   Asic  EN/                     Far-End         Far-End
           Link Speed  Stage Oper                    Link (FSDB)     Link (HW)
                (Gbps)       Status
-----------------------------------------------------------------------------
0/2/0/16      - 50.00 FIA    EN/DN  ...........    0/FC2/7/157     NC
0/2/0/17      - 50.00 FIA    EN/DN  ...........    0/FC2/8/156     NC

Node ID: 0/RP0/CPU0
-----------------------------------------------------------------------------
Link ID    Log  Link   Asic  EN/                     Far-End         Far-End
           Link Speed  Stage Oper                    Link (FSDB)     Link (HW)
                (Gbps)       Status
-----------------------------------------------------------------------------
0/FC2/7/20      - 50.00 FIA    EN/DN  ...........    0/2/0/178     NC
0/FC2/8/33      - 50.00 FIA    EN/DN  ...........    0/2/0/166     0/2/0/188
```

  In this example, links between LC2 and FC2 are down.

Identify UCE Drops of Fabric Plane

If all links are UP, then monitor the Uncorrectable Errors (UCE). UCE means that there is packet corruption between a line card and a fabric card.

- To identify packet corruption, run the following command:

```
RP/0/RP0/CPU0:Router# show controllers fabric plane all statistics
Fri Feb  2 07:21:13.447 UTC

    Flags: E-D  - Exceeded display width.
                  Check detail option.

                     In                   Out        CE        UCE        PE
Plane             Packets               Packets   Packets     Packets   Packets
--------------------------------------------------------------------------------
 0                835649707985         733726217071      0         0         0
```

```
1              835649703742        733726213241      0          0          0
2              835649542985        733763942830      0      43921          0
```

Note 2

You may notice a few errors initially at card insertion. So, run this command multiple times with a 30-sec interval. If you notice the **UCE packets** column count increasing, then it is evident that there are links on that plane (plane number = FC slot number) with UCE errors.

The sample output indicates that there are Uncorrectable Errors (UCE) on FE ASICs on fabric card in slot 2 (FC2).

- To identify the line card and fabric cards between which the errors are detected, use the **show controllers npu stats link all instance all location all** command.

```
RP/0/RP0/CPU0:Router# show controllers npu stats link all instance all
location all
Fri Feb  2 07:38:04.699 UTC

Node ID: 0/0/CPU0

                       In Data      Out Data        CE        UCE      CRC
                        Frames        frames    Frames     Frames   Errors
-------------------------------------------------------------------------
0/0/0/0-1                        0            0        0         0        0
0/0/0/2-3                        0            0        0         0        0
0/0/0/4-5                        0            0        0         0        0
0/0/0/6-7                        0            0        0         0       89
0/0/0/8-9                        0            0        0         0        0
```

Errors are detected on fabric links 6, 7.

- If the **monitor dataplane-health** command still displays packet loss despite no issues being detected after link debugging, collect all logs and contact Cisco TAC before draining and reloading the router.

Collect Logs of Fabric and Line Cards and Contact Cisco TAC

Run the following show commands to collect logs for the impacted LC and all FCs.

- **show tech fabric link-include**
- **show tech ofa**
- **show tech interface**
- **show tech optics**

In addition, run the following debugshell commands, and collect logs for all NPUs on all LCs, and all FEs on all FCs.

```
show controllers npu debugshell <np_num/unit_num> "script
print_get_counters" location 0/x/CPU0
```

```
show controllers npu debugshell <np_num> "script sf_oq_debug" location
0/x/CPU0
```

```
show controllers npu debugshell <np_num> "script sf_fabric_debug" location
 0/x/CPU0
```

```
show controller npu stats counters-all instance all loc 0/x/CPU0
```

Note 3

To collect logs for an FC card, obtain the **np_num** from the **fap_id** column of **show controllers npu driver loc 0/RP0/CPU0** command output (0/RP0/CPU0 is the location of FC).

Note 4

- Starting from Cisco IOS XR Release 24.4.1, the **show controllers npu stats counters-all** command is deprecated and will not be supported in future releases. Instead, you can use **show controllers npu debug-utils get-counters instance** <> loc <> command.
- The **show controllers npu stats counters-all detail instance** <> **loc** <> command is now replaced by **show controllers npu stats counters-detail instance** <>**loc** <> command.

Packet Tracer

Packet Tracer is a debugging framework that is used to trace custom flows through the router.

Table 50: Feature History Table

Feature Name	Release	Description
Packet Tracer	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) *This feature is supported on: • 8011-12G12X4Y-A • 8011-12G12X4Y-D

Feature Name	Release	Description
Packet Tracer	Release 25.3.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100]); Modular Systems (8800 [LC ASIC: P100]) We now extend the Packet Tracer support on P100 based line cards and fixed chasis. *This feature is now supported on: • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM • 8212-48FH-M • 8711-32FH-M
Packet Tracer	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100]) Packet Tracer is a framework that enables you to trace custom flows through the router for service validation or troubleshooting. This feature utilizes the existing XR packet tracing infrastructure to facilitate debugging of packet flows at the ASIC and hardware levels. **CLI:** This feature introduces the following commands. • **packet-trace start** • **packet-trace stop** • **packet-trace condition** • **clear packet-trace conditions all** • **clear packet-trace counters all** • **show packet-trace status [detail]** • **show packet-trace result** • **packet-trace description**

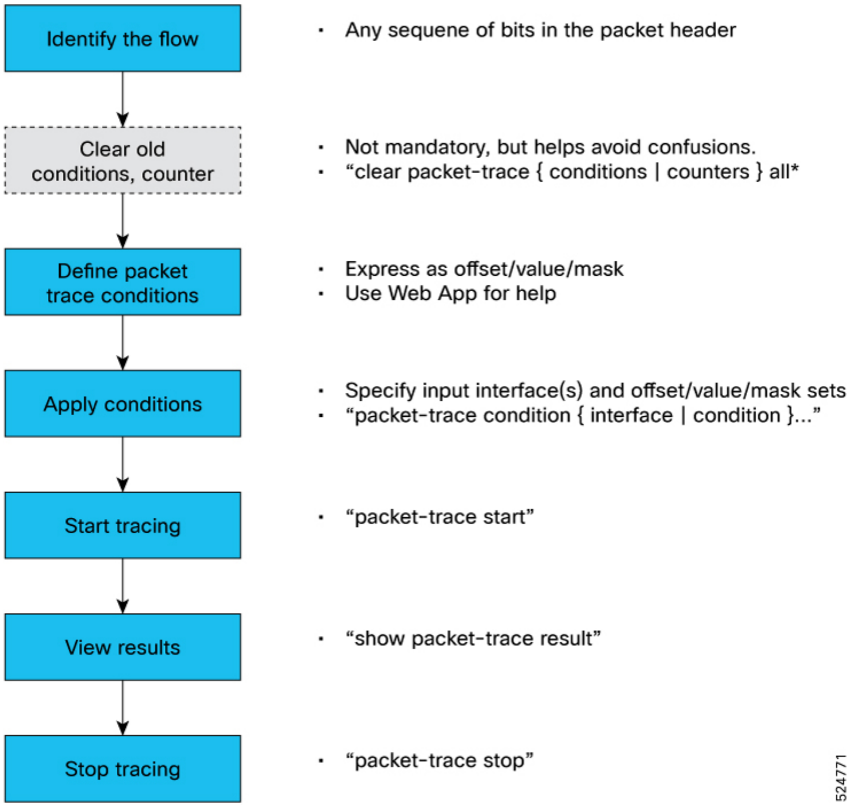
How packet tracer work

Packet tracing is accomplished using the packet tracer framework, which facilitates the tracking and analysis of packet flow. The framework's user interface is accessed through packet tracer commands.

When packet tracing is enabled on an interface, the Network Processor (NP) evaluates incoming packets to see if they meet specified conditions. If a packet meets the defined conditions, a flag is set in its internal header. This flag enables the tracing of the packet as it moves through all elements in the data path within the router. In Cisco IOS XR Release 25.1.1, the packet tracer captures a maximum of one packet on ingress and one packet on egress. To capture the next packet, you need to stop the current tracing session and start a new one.

The figure below illustrates the configuration flow for Packet Tracer.

Figure 10: Packet Tracer configuration flow


Identify the flow
• Any sequene of bits in the packet header
Clear old conditions, counter
• Not mandatory, but helps avoid confusions.
• "clear packet-trace { conditions | counters } all*
Define packet trace conditions
• Express as offset/value/mask
• Use Web App for help
Apply conditions
• Specify input interface(s) and offset/value/mask sets
• "packet-trace condition { interface | condition }..."
Start tracing
• "packet-trace start"
View results
• "show packet-trace result"
Stop tracing
• "packet-trace stop"
524771

Guidelines for configuring packet tracer

Packet tracer conditions

Packet tracer conditions consist of the following two entities:

- Physical interface(s) on which the packets are expected to be received.

  For example:

  ```
  packet-trace condition interface hu0/5/0/6
  ```

  Cisco 8000 series routers does not support per interface filters. Instead, you must use:

  ```
  packet-trace condition interface all
  ```

Note When defining conditions, only physical or bundle interfaces are allowed. If you are tracing on sub-interfaces, you must consider the dot1q or Q-in-Q encapsulation when specifying the Offset/Value/Mask definition.

- Offset/Value/Mask triplets that define a flow of interest.

  Defining a flow as a set of Offset/Value/Mask triplets allows the packet tracer framework to remain entirely protocol- agnostic. These triplets can represent any segment of any header within the protocol stack. Ensure that both the value and mask do not exceed 4 bytes in size. You can configure conditions at any offset within the first 128 bytes of a given packet, using a value and mask of up to 4 bytes each.

Note The **offset 0** is the start of the ethernet frame.

Configure packet tracer

Before you begin

To configure packet tracer and start tracing, you must first define the conditions. Use the Condition Generator Web User Interface tool (https://github.com/xr-packet-tracer) or any other tool of your choice, to define conditions.

For more information about defining these conditions and using the condition generator tool, see https://xrdocs.io/asr9k//tutorials/xr-embedded-packet-tracer/.

Procedure

Step 1 Clear packet tracing.

Use the **clear packet-trace conditions all** command to clear all buffered packet-trace conditions.

```
router# clear packet-trace conditions all
```

Step 2 Define conditions as per your network structure.

Use the Condition Generator Web User Interface tool (https://github.com/xr-packet-tracer) or any other tool of your choice, to define conditions.

Example:

Below are a few examples of conditions that have been generated.

```
condition interface all
condition 1 offset 0 value 0xaabbcc mask 0xffffff
condition 2 offset 3 value 0xdd mask 0xff
```

Note

- You must also use the **interface all** condition along with the defined conditions, as this feature is enabled on all NPUs rather a specific interface.

Step 3 Configure the defined conditions.

Example:

Below is an example to configure packet trace for the defined conditions.

```
router# packet-trace condition interface all
router# packet-trace condition 1 offset 0 value 0xaabbcc mask 0xffffff
router# packet-trace condition 2 offset 3 value 0xdd mask 0xff
```

Step 4 Check the status of tracing.

Apply the conditions and verify if they are configured correctly.

Example:

Use the **show packet-trace status** command to check the status of tracing.

```
router# show packet-trace status
```

```
Tue Sep 11 18:29:07.249 UTC
------------------------------------------------------------
Packet Trace Master Process:
Buffered Conditions:
1 offset 0  value 0xaabbcc mask 0xffffff
2 offset 3  value 0xdd     mask 0xff
Status: Inactive
```

Note
To view more detailed status, such as processes that are registered with the packet tracer framework on every card in the router, use the **show packet-trace status detail** command.

The output displays the status as either **Active** or **Inactive**. The status will remain **Inactive** until tracing starts.

Step 5 Start packet tracing.

Use the **packet-trace start** command to start packet tracing.

```
router# packet-trace start
```

Step 6 Check the status of tracing.

Example:

Use the **show packet-trace status** command to check the status of tracing.

```
router# show packet-trace status
```

The status should now display as **Active**.

```
Tue Sep 11 18:32:08.250 UTC
------------------------------------------------------------
Packet Trace Master Process:
Buffered Conditions:
1 offset 0 value 0xaabbcc mask 0xffffff
2 offset 3 value 0xdd mask 0xff
Status: Active
```

Step 7 View the packet tracing results.

Example:

Use the **show packet-trace result** command to view the results of packet tracing.

```
router# show packet-trace result
```

The following output is displayed.

```
Tue Sep 11 18:26:37.595 UTC
T: D - Drop counter; P - Pass counter
Location     | Source       | Counter               | T | Last-Attribute         | Count
------------   ------------   ----------------- ------------------------     ------------
0/RP0/CPU0     NP0            INGRESS                 P                              1
0/RP0/CPU0     NP0            EGRESS                  P                              1
```

The count value of 1 indicates that the incoming packet has matched the configured filter in both the ingress and egress pipelines.

Step 8 (Optional) View packet trace counters and their descriptions.

Use the **show packet-trace descriptions** command to view all counters registered with the packet tracer framework along with their descriptions.

```
router# show packet-trace descriptions
```

Step 9 (Optional) View decoded packet trace data.

Use the **show packet-trace decode** command to view decoded packet trace information.

Example:

To view raw binary packet trace data, use the **show packet-trace decode raw** command.

```
router# show packet-trace decode raw instance 0 location 0/RP0/CPU0
```

The following output is displayed.

```
Tue Sep 11 18:29:23.717 UTC
Decoding packet trace results for NPU id 0 on node id 0x2000
=== Trace decode results for NP0 ===
{
"packet_traces": [
{
"slice-id": 0,
"termination-input": [
{
"raw":
"00aabbccddeeff01940000020108004500002e00000000403d4e7fc800000a6400000a000102030405060708090a0b0c0d0e0f10111213141516171819000000000194"
}
],
"termination-output": [
{
"raw":
"00aabbccddeeff01940000020108004500002e00000000403d4e7fc800000a6400000a000102030405060708090a0b0c0d0e0f10111213141516171819000000000194"
}
],
"raw": "00000061e0013a0987fd80000028077fffff7fffff7fffff0085c89ffff4000000"
}
],
"ifg1_tm-pd-output": [
{
"raw": "00001d8fd6ff6c100000001cff8000677c7b5e5bfc1c6bd7000024000000000000"
}
],
"ifg0_rxpp-output-npu-header": [
{
"raw": "00001d8fd6ff6c100000001cff8000677c7b5e5bfc1c6bd7000024000000000000"
}
],
"ifg1_rxpp-output-npu-header": [
{
"raw":
"00c1b76c4f3dfff77fffb67e50000000000000000000000000000000000000000000000000000000000000201"
}
],
"rxpp-termination-macro-stack": [
{
"raw": "000000000000000000"
}
],

},
{
"slice-id": 1,
"termination-input": [
{
"raw": "0"
}
],
"termination-output": [
{
"raw": "0"
}
],

"ifg0_rxpp-output-npu-header": [
{
"raw": "0"
}
],
"rxppp-forwarding-macro-keys": [
{}
],
"rxppp-forwarding-macro-results": [
{}
],
"transmit-input": [
{
"raw": "0"
}
],
"transmit-pre-npe": [
{
"raw": "0"
}
],
"transmit-post-npe": [
{
"raw": "0"
}
],
"transmit-post-ene": [
{
"raw": "0"
}
],
"txpp-macro-stack": [
{
"raw": "0"
}
],
"txpp-npe-macro-keys": [
{}
],
"txpp-npe-macro-results": [
{}
]
}
```

```
]
}
```

Step 10 Stop packet tracing.

Use the **packet-trace stop** command to stop packet tracing.

```
router# packet-trace stop
```

CHAPTER 13

Traffic Monitoring

For a router to function smoothly and effortlessly, monitoring traffic is an essential task. Traffic Monitoring helps you get count of packets going inwards and outwards.

This chapter describes the process to monitor packet drops as part of troubleshooting process.

In routing, the information is passed around in the form of packets. Packets contains unit size data. Sometimes,due to network congestion, or outdated software/hardware the packets fail to reach at their destined location.You can identify packet loss in the form of incomplete information or missing information.

Traffic Statistics with Packet Drop Location

Table 51: Feature History Table

Feature Name	Release	Description
Traffic Statistics with Packet Drop Location	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on Cisco 8011-4G24Y4H-I routers.

Feature Name	Release	Description
Traffic Statistics with Packet Drop Location	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100, K100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 8712-MOD-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
Traffic Statistics with Packet Drop Location	Release 24.2.11	Introduced in this release on: Fixed Systems(8200); Centralized Systems (8600); Modular Systems (8800 [LC ASIC: Q100, Q200]) We help you save debugging time to locate packet drops by automatically detecting nonzero traffic drops from the commands running in the background and giving you the exact location of the packet drop. In earlier releases, you used multiple show commands with their respective locations to detect packet drops. This feature introduces the **show drops all** command.

Earlier, finding the exact location of packet drop was a long and tedious process as there are multiple node locations. You were executing show commands with their different locations to detect the packet drop location.

Starting Cisco IOS XR Software Release 24.2.11, finding a packet drop location has become easy and quick. You can use the **show drops all** command to know the exact packet drop location. This command shows all nonzero traffic-drops in the node at one place. This command automatically runs the required IOS XR debug commands in the background and removes insignificant information from the command output.

The following commands outputs are integrated in the **show drops all** command:

- **show arp traffic**
- **show controllers npu stats traps-all instance** *instance* **location** *location*
- **show controllers npu stats voq ingress interface** *interface name* **instance all location** *location*
- **show cef drops location** *location*

- **show lpts pifib hardware police location** *location*
- **show spp node-counters location** *location* | **inc drop**
- **show controllers npu stats counters-drop instance** *instance* **location** *location*

Monitor Packet Drops

Use **show drops all ongoing location all** to see ongoing drops on the system. This command output shows ongoing drops since the last time command was executed.

```
Router# show drops all ongoing location all

-----------------------------------------------
Printing Drop Counters for node 0/RP0/CPU0
-----------------------------------------------

-------------------
MODULE cef
-------------------

cef ipv4 drops
------------------
No route drops       : 18

-------------------
MODULE spp
-------------------

      Local Linux Packet drops:             683
         Total Socket RX Drops:             683

-----------------------------------------------
Printing Drop Counters for node 0/0/CPU0
-----------------------------------------------

-------------------
MODULE npu
-------------------

Trap Type                                     NPU  Trap  Punt       Punt  Punt  Punt
Configured Hardware   Policer Avg-Pkt Packets              Packets
                                            ID  ID    Dest       VoQ  VLAN  TC   Rate(pps)
  Rate(pps)  Level   Size    Accepted             Dropped
==========================================================================================

INJECT_UP_L3_LOOKUP_FAIL                      0    109  BOTH_RP_CPU 310   1538  6    5000
      4878       IFG     1520    18                   0

-------------------
MODULE spp
-------------------

      Local Linux Packet drops:              86
         Total Socket RX Drops:             104
   Drop wrong Mcast pkt on SIM:              18
```

Use **show drops all location all** to see the drops from all location or node that have happened in the system.

```
Router# show drops all location all

-----------------------------------------------
```

```
Printing Drop Counters for node 0/RP0/CPU0
-----------------------------------------------

-------------------
MODULE cef
-------------------

cef ipv4 drops
------------------
No route drops       : 18

-------------------
MODULE spp
-------------------

      Local Linux Packet drops:             683
         Total Socket RX Drops:             683

-----------------------------------------------
Printing Drop Counters for node 0/0/CPU0
-----------------------------------------------

-------------------
MODULE npu
-------------------

Trap Type                                     NPU  Trap  Punt       Punt  Punt  Punt
Configured Hardware   Policer Avg-Pkt Packets              Packets
                                           ID   ID    Dest      VoQ   VLAN  TC   Rate(pps)
  Rate(pps)  Level   Size    Accepted            Dropped
===========================================================================================

INJECT_UP_L3_LOOKUP_FAIL                      0    109  BOTH_RP_CPU 310   1538  6    5000
      4878       IFG     1520    18                    0

-------------------
MODULE spp
-------------------

      Local Linux Packet drops:              86
         Total Socket RX Drops:             104
   Drop wrong Mcast pkt on SIM:              18
```

Aggregated drop counters for software packet path node counters

The Software Packet Path (SPP) handles the movement and management of packets in software across network devices.

Table 52: Feature History Table

Feature Name	Release	Description
Aggregated drop counters for software packet path node counters	Release 25.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100, K100]), 8010 [ASIC: A100]); Centralized Systems (8600 [ASIC: Q200]); Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) You can use the **show spp node-counters drops** command to view aggregated drop counters for the SPP. This displays a summary of dropped packets at the node level to identify and troubleshoot drop issues efficiently. The feature introduces these changes: Modified Command: **CLI** The **show spp node-counters** command was modified to introduce the **drops** keyword.

The **show spp node-counters** command displays node counters for the SPP. These counters include forwarded packets, injected packets, and dropped packets for different nodes and clients.

From Release 25.4.1, the **show spp node-counters** command includes the *drops* keyword. You can use this keyword to filter the output and display aggregated dropped packets for each node in the SPP.

Enter the **show spp node-counters drops** command to view aggregated drop counters for the SPP.

```
Router# show spp node-counters drops
0/0/CPU0:
socket/rx
aggr-drop-counter: 371
-------------------------------
socket/tx
aggr-drop-counter: 372
```

You can also collect aggregated drop counter statistics programmatically using the **Cisco-IOS-XR-spp-oper.yang** data model.

Sample Netconf get request:

```
<rpc message-id="9a7097c8-f1b4-44d3-9e42-38158dd33b41"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <filter>
      <spp xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-spp-oper">
        <nodes>
          <node>
          <node-name>0/RP0/CPU0</node-name>
                <spp-node-drop-counters>
                <spp-node-drop-counter>
                </spp-node-drop-counter>
                </spp-node-drop-counters>
          </node>
        </nodes>
      </spp>
    </filter>
  </get>
</rpc>
```

Sample Netconf Get Response:

```
<?xml version="1.0"?>
<rpc-reply message-id="9a7097c8-f1b4-44d3-9e42-38158dd33b41"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <spp xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-spp-oper">
      <nodes>
        <node>
          <node-name>0/RP0/CPU0</node-name>
          <spp-node-drop-counters>
            <spp-node-drop-counter>
              <spp-node-entry>0</spp-node-entry>
              <spp-node-name>pd_span_drop</spp-node-name>
              <aggr-drop-counters>1</aggr-drop-counters>
            </spp-node-drop-counter>
         #203
            <spp-node-drop-counter>
              <spp-node-entry>1</spp-node-entry>
              <spp-node-name>socket/rx</spp-node-name>
              <aggr-drop-counters>194</aggr-drop-counters>
            </spp-node-drop-counter>
         #201
            <spp-node-drop-counter>
              <spp-node-entry>2</spp-node-entry>
              <spp-node-name>socket/tx</spp-node-name>
              <aggr-drop-counters>3</aggr-drop-counters>
            </spp-node-drop-counter>
         #207
            <spp-node-drop-counter>
              <spp-node-entry>3</spp-node-entry>
              <spp-node-name>device/classify</spp-node-name>
              <aggr-drop-counters>1</aggr-drop-counters>
            </spp-node-drop-counter>
         #86
          </spp-node-drop-counters>
        </node>
      </nodes>
    </spp>
  </data>
</rpc-reply>

5.983184ms elapsed
```

Restrictions for aggregated drop counters for software packet path node counters

- Cisco-IOS-XR-spp-oper.yang data model does not support get requests at the spp-node-name level.

System Log Alerts for Packet Loss

Table 53: Feature History Table

Feature Name	Release Information	Description
System Log Alerts for Packet Loss	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I
System Log Alerts for Packet Loss	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

Feature Name	Release Information	Description
System Log Alerts for Packet Loss	Release 24.1.1	You can quickly get notified about any traffic impacting errors within the router's Network Processing Unit (NPU). These notifications are error log messages on the router console for NPU interrupts that affect traffic. To diagnose traffic loss, follow the recommended actions in the log. Previously, the only way to identify NPU errors that impacted traffic was to run the **show asic-error** command. This feature introduces the following changes: **CLI:** This feature introduces the **hw-module profile packet-loss-alert** command. **YANG Model:** New xpaths for `Cisco-IOS-XR-npu-hw-profile-cfg.yang` data model (see GitHub, YANG Data Models Navigator)

Network packet loss can significantly impact the overall experience of end-users, particularly when using real-time applications such as voice or video. Therefore, network administrators need to address such issues promptly to ensure the integrity of these services.

Now you can configure your router to provide immediate alerts in the event of packet loss, along with configuring the duration of packet loss to raise the alert. This feature enables network administrators to quickly identify the specific router within the network that is experiencing the problem. It facilitates a swift and precise response to rectify the issue and maintain optimal network performance.

Note Only line cards and routers with the Q100, Q200, P100, or G100 based Silicon One ASIC support this feature.

Configuration Example

Execute the **hw-module profile packet-loss-alert** command to enable this feature, as shown below:

```
Router# configure
Router(config)# hw-module profile packet-loss-alert 3Min
Router(config)# commit
```

You can configure the duration of packet loss to raise the alert to either 3 minutes or 5 minutes.

Running Configuration

```
Router# show running-config hw-module profile
hw-module profile packet-loss-alert 3Min
```

System Log Alert Generated for Packet Loss

When you enable this feature, the router generates a system log message whenever there’s a packet loss for the configured duration. To diagnose the reason for the packet loss, follow the recommended action in the log.

```
LC/0/3/CPU0:Nov 4 21:12:47.062 UTC: npu_drvr[213]:
%FABRIC-NPU_DRVR-3-ASIC_ERROR_TRAFFIC_IMPACT : [10118] : npu[2]: Potential PACKET_LOSS due
 to error, please check configuration to see if drop is expected; if not, collect showtech
 fabric link-include and follow the TAC guideline for this message
```

Monitor interface

Table 54: Feature History Table

Feature Name	Release	Description
Monitor interface	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is now supported on Cisco 8011-4G24Y4H-I routers.

Feature Name	Release	Description
Monitor interface	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100, K100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) The **filter physical** keyword was introduced, along with new columns `InDrops` and `OutDrops` in the output, to provide enhanced monitoring capabilities for physical interfaces. **CLI:** - The **filter physical** keyword is added to the **monitor interface** command. *This feature is supported on: - 8212-48FH-M - 8711-32FH-M - 8712-MOD-M - 88-LC1-36EH - 88-LC1-12TH24FH-E - 88-LC1-52Y8H-EM
Monitor interface	Release 7.5.4	The keyword **full-name** is added, allowing users to display the full names of interfaces, which is particularly useful for interfaces with long or descriptive names. **CLI:** - The **full-name** keyword is added to the **monitor interface** command.

Feature Name	Release	Description
Monitor interface	Release 7.0.12	Introduced in this release on: Fixed Systems(8200); Centralized Systems (8600); Modular Systems (8800 [LC ASIC: Q100, Q200]) Monitor interface is introduced to enable real-time monitoring of interface counters on Cisco routers. This feature introduces the **monitor interface** command.

The **monitor interface** command is used to monitor network interface counters in real-time.

This command provides valuable insights into the performance and status of both physical and virtual interfaces on a router.

By leveraging various arguments, you can customize the output to display

- detailed statistics,
- full interface names, and
- filter for physical interfaces.

The **monitor interface** command is essential for network administrators to diagnose and troubleshoot interface-related issues effectively.

Enhancing FIB hardware programming failure recovery in non-OOR scenarios

Forwarding Information Base (FIB) hardware programming failure recovery in Cisco network devices refers to the process by which the router

- detects and responds to failures encountered when updating or programming entries in the hardware FIB
- ensures the integrity and reliability of the packet forwarding process by attempting corrective actions when hardware programming fails, and
- minimizes disruption to data traffic by systematically managing and recovering from such failures.

Table 55: Feature History Table

Feature Name	Release Information	Description
Enhancing FIB hardware programming failure recovery in non-OOR scenarios	Release 25.3.1	Introduced in this release on: Cisco 8000 with ASIC Q100, Q200, P100, K100, and A100: Fixed Systems (8200 [ASIC: Q100, Q200, P100], 8700 [ASIC: P100, K100], 8010 [ASIC: A100]); Centralized Systems (8600 [ASIC: Q200]); Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) FIB hardware programming failure recovery enhancement increases network stability and reduces performance impact during hardware programming issues in non-OOR situations by taking these actions: • Reduces churn by limiting the recovery attempts to twice, instead of retrying every 15 seconds. • Removes the errored object from the forwarding tree. • Displays syslog messages with details about the hardware failure and its cause. • Attempts recovery by deleting and then re-creating the affected hardware programming entry.

The enhancement to FIB's hardware programming failure recovery reduces churn by limiting recovery attempts, thus improving traffic congestion. It also provides detailed syslog messages for debugging purposes.

Before Release 25.3.1, the router attempted to recover from hardware programming failures every 15 seconds, leading to excessive churn.

From Release 25.3.1, the recovery process is limited to two attempts:

- **First attempt**:
  - The router triggers the first recovery attempt 15 seconds after it detects the failure.
  - It removes the errored object from the forwarding tree.
  - It deletes and re-creates the hardware programming entry.
  - It generates a syslog message detailing the failure and the recovery attempt.
- **Second attempt**:
  - The router triggers the second recovery attempt at the third minute if the first attempt fails.
  - It performs another deletion and re-creation of the entry.
  - It generates another syslog message.

If both attempts fail, the router enters an unrecoverable state for the affected entry and remains in this state until routing protocols remove the errored entry from the FIB.

After each recovery attempt, the Cisco IOS XR software displays a syslog message detailing the hardware failure:

- **First attempt**: The router retries to recover from the failure and displays the following system log message:

  ```
  %ROUTING-3-PLATF_UPD_FAIL: FIB platform update failed:
  Obj=FIB_DATA_TYPE_LEAF[ptr=0x22336399dc28,refc=0x2,flags=0x1000003]
  Action=FAIL-RETRY Proto=ipv4. Cerr='FIB' detected the 'try again' condition
  'Temporary failure. Try again later’
  ```

- **Second attempt**: The router stops retrying and displays the following system log message:

  ```
  %ROUTING-3-PLATF_UPD_FAIL: FIB platform update failed:
  Obj=FIB_DATA_TYPE_LEAF[ptr=0x22336399dc28,refc=0x2,flags=0x1000003]
  Action=REPEATED-FAIL-STOP-RETRY Proto=ipv4. Cerr='FIB' detected the 'try again' condition
  'Temporary failure. Try again later’
  ```

Traffic class latency histogram

The Transmit Packet Processor (TXPP) is a specialized hardware component in Cisco network devices that

- processes and manages the transmission of packets through the Network Processing Unit (NPU)
- calculates, updates, and maintains key packet metrics such as latency and transmission statistics, and
- enables precise monitoring and analysis of traffic behavior for various Traffic Classes (TCs).

Table 56: Feature History Table

Feature Name	Release Information	Description
Traffic class latency histogram	Release 25.3.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200]); Centralized Systems (8600 [ASIC: Q200]); Modular Systems (8800 [LC ASIC: Q200]) You can now monitor packet delays between ingress and egress on specific ports using a new visual representation featuring latency histograms. These histograms provide detailed insights into packet delays and jitter, enabling you to identify bottlenecks, optimize traffic flows, and enhance network efficiency. The feature offers latency analysis for each TC across multiple levels, including Network Processing Unit (NPU), slice, Inter-Frame Gap (IFG), and specific TCs. **CLI:** This feature introduces the following commands: • **show controllers npu packet-latency instance** • **clear controllers npu packet-latency instance** **YANG Model:** • `Cisco-IOS-XR-8000-platforms-npu-latency-histogram-oper.yang` • `Cisco-IOS-XR-8000-platforms-npu-latency-histogram-oper-sub1.yang` • `Cisco-IOS-XR-8000-platforms-npu-latency-histogram-act.yang` (see GitHub, YANG Data Models Navigator)

Within the NPU architecture, TXPP provides essential functions such as

- measuring packet delays between ingress and egress points to support latency histograms
- enabling precise analysis of latency distribution for each TC across multiple levels, including NPU, slice, IFG, and specific TCs
- updating real-time metrics for network performance monitoring, and
- integrating with other monitoring tools to facilitate efficient troubleshooting and performance optimization.

From Release 25.3.1, TC latency histogram is enabled by default and provides a detailed insights into latency distribution and network performance.

The Cisco IOS XR software collects aggerageted packet latency every 30 seconds, and exports 30 records every 30 seconds, ensuring timely and detailed telemetry.

You can view the cumulative packet count for each TC since either system bootup or the last clear using the **show controllers npu packet-latency instance** command.

You can manually clear the cumulative packet count for each TC since either system bootup or the last clear using the **clear controllers npu packet-latency instance** command.

Limitations for traffic class latency histogram

These are the limitation for traffic class latency histogram:

- The Cisco IOS XR Software does not support SNMP based monitoring.
- You must manually clear histogram data using **clear controllers npu packet-latency instance** commands, as the system does not automatically reset data over time.

How traffic class latency histogram works

Workflow

1. TXPP calculates the per-packet latency, which is the difference between ingress and egress timestamp, and categorizes the latency into one of the 8 histogram bins based on the configured range.
2. The Cisco IOS XR Software aggregates packet latency every 30 seconds and exports 30 records every 30 seconds to collect data for all enabled NPUs, slices, IFGs, and TCs.
3. When you run the **show controllers npu packet-latency instance** command the Cisco IOS XR Software displays the TC latency histogram data in the desired granularity.

View Traffic class latency histogram

Perform these steps to view or clear the cumulative packet count in the latency histogram bin:

Procedure

Step 1 Run the **show controllers npu packet-latency instance** command to view the cumulative packet count in the latency histogram bin for all TC or each TC since either system bootup or the last clear.

Example:

You can view the cumulative packet count in the latency histogram bin with granularity for slice, IFG, location, and traffic class.

The following output displays cumulative packet count in the latency histogram bin for all TCs on slice 2, IFG 1, and location 0/2/CPU0:

```
Router# show controllers npu packet-latency instance 0 slice 2 ifg 1 location 0/2/CPU0

----------------------------------------------------------------------------------------------------
Packet Latency Histogram Bin Range In nanosecond: Node ID: 0/2/CPU0
Time             NPU  Slice IFG TC        5000        6000        7000        8000        9000
   10000       15000    4000000000  Max_delay
----------------------------------------------------------------------------------------------------
14:43:50.043 EDT   0   2    1    0            0           0           0           0           0
       0           0           0           0
14:43:50.043 EDT   0   2    1    1            0           0           0           0           0
```

```
       0           0           0           0
14:43:50.043 EDT   0   2    1    2            0  1627324790      387009           0           0
       0           0           0        6400
14:43:50.043 EDT   0   2    1    3            0           0           0           0           0
       0           0           0           0
14:43:50.043 EDT   0   2    1    4            0           0           0           0           0
       0           0           0           0
14:43:50.043 EDT   0   2    1    5            0           0           0           0           0
       0           0           0           0
14:43:50.043 EDT   0   2    1    6            0   102965743    28788249       10708           0
       0           0           0        7936
14:43:50.043 EDT   0   2    1    7        38680          15           1           1           0
       0           0           0        7680
```

Step 2 Run the **clear controllers npu packet-latency instance** command to manually clear the cumulative packet count in the latency histogram bin for all TC or each TC since either system bootup or the last clear.

Example:

The following output displays manually clearing the cumulative packet count for all TC on slice 2, IFG 1, and location 0/2/CPU0 since either system bootup or the last clear:

```
Router# clear controller npu packet-latency instance 0 slice 2 ifg 1 location 0/2/CPU0
```

Port rate histogram

A port rate histogram is a network analytics tool that:

- divides port traffic rates into discrete bins representing specific utilization thresholds,
- displays the frequency of port traffic for each threshold over defined time intervals, and
- enables granular monitoring, analysis, and troubleshooting of network port utilization.

Table 57: Feature History Table

Feature Name	Release Information	Feature Description
Port rate histogram	Release 25.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100](select variants only*); Centralized Systems (8600 [ASIC:Q200])(select variants only*); Modular Systems (8800 [LC ASIC: Q200, P100])(select variants only*) Port rate histogram enables rapid detection of network performance issues by providing detailed, real-time port utilization analysis. The feature displays port traffic rates in a histogram, breaking down data into specific time intervals and utilization bins. Data is collected and displayed using the CLI, allowing you to monitor and compare all network ports on supported line cards, and detect anomalies and performance issues efficiently. This feature is supported on: • 8202-32FH-M • 8608-SYS • 88-LC0-36FH • 88-LC0-36FH-M • 88-LC0-34H14FH • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM The feature introduces these changes: CLI • **hw-module profile port-rate-histogram** • **show controllers npu port-rate-histogram**

The port rate histogram view provides a detailed display of port rate utilization. This view helps monitor traffic usage on high-speed network ports and analyze it at a granular level. The histogram breaks down data into specific time intervals, allowing for fine-grained observation and in-depth analysis.

Benefits of port rate histogram

The port rate histogram offers several benefits.

- **Microburst monitoring:** It provides microburst monitoring capabilities and identifies rapid and short-duration spikes in traffic.

- **Congestion visibility:** It offers a better histogram view of bursts and congestion in the network.
- **Identification of low port utilization:** This helps identify ports with low utilization.
- **Load balancing improvement:** It improves load balancing by checking port utilization at a granular level for bundle members and Equal-Cost Multi-Path (ECMP) members. The histogram identifies congestion that may require load balancing or proper traffic management.
- **Efficient data handling:** The histogram holds up to one snapshot per port with different bin thresholds. The system updates the local data structure every 30 seconds.

Configuration guidelines for port rate histogram

Consider these guidelines and best practices for configuring port rate histogram:

- The system automatically clears histogram data when you remove or swap a port.
- Histogram data is not polled if a port's link is down.
- Ensure that you clear histogram data when the router or the line card is reloaded.

Restrictions for port rate histogram

These restrictions apply to port rate histogram:

- Hardware limitation: The port rate histogram is supported only on Q200 and P100 based ASICs and is not supported on Q100, A100, and K100 based ASICs.
- The histogram does not support virtual interfaces.
- Process restart is not supported.
- Fabric card is not supported.
- Due to memory limitations, the system maintains only a single snapshot per port for both transmit (Tx) and receive (Rx) counters.

Configuring port rate histogram

Use this task to enable and monitor port rate histogram data to analyze granular port utilization and traffic patterns on supported routers and line cards.

Procedure

Step 1 Enable the port rate histogram feature for the desired ports or line cards by using the below command:

```
hw-module profile port-rate-histogram location node-id enable
```

Example:

```
Router# hw-module profile port-rate-histogram location 0/7/CPU0 enable
```

Step 2 View detailed port rate histogram data for specific interfaces or port ranges.

Use the show command to display average samples and bin data:

```
show controllers npu port-rate-histogram avg-samples location node-id
```

Example:

```
Router# show controllers npu port-rate-histogram avg-samples location 0/0/CPU0

Sample Output:
Tue Nov 25 16:50:20.173 EST
==============================================
Port rate Histogram
==============================================

HW Polling Period     : 1  Seconds
HW Snapshot Interval  : 32 Microseconds
Description:
==============================================
Cumulative  : Accumulated samples since the port is up
Delta       : Delta samples between the cli's
avg-samples    - aggregation of samples from various time window
avg-percentage - aggregation of samples percentage from various time window
+==================================================================================================+
Interface     Directions  Durations    5<-35%           35-<60%         60-<80%         80-<90%
     90-<95%         95-<99%         >99%
+==================================================================================================+
+--------------------------------------------------------------------------------------------------+
last Rx snapshot timestamp:1764107377549798104 [Local time: 2025-11-25 16:49:37]
+--------------------------------------------------------------------------------------------------+
0/0/0/0         Rx         30sec          0               0               0               0
       0               0               0
0/0/0/0         Rx         5mins          0               0               0               0
       0               0               0
0/0/0/0         Rx         1hour          0               0               0               0
       0               0               0
0/0/0/0         Rx         24hours        0               0               0               0
       0               0               0
0/0/0/0         Rx         Delta          0               0               0               0
       0               0               0
0/0/0/0         Rx         Cumm           0               0               0               0
       0               0               0
+--------------------------------------------------------------------------------------------------+
last Tx snapshot timestamp:1764107377549798104 [Local time: 2025-11-25 16:49:37]
+--------------------------------------------------------------------------------------------------+
0/0/0/0         Tx         30sec          0               0               0               0
       0               0               0
0/0/0/0         Tx         5mins          0               0               0               0
       0               0               0
0/0/0/0         Tx         1hour          0               0               0               0
       0               0               0
0/0/0/0         Tx         24hours        0               0               0               0
       0               0               0
0/0/0/0         Tx         Delta          0               0               0               0
       0               0               0
0/0/0/0         Tx         Cumm           0               0               0               0
       0               0               0
+--------------------------------------------------------------------------------------------------+
last Rx snapshot timestamp:1764107377549782105 [Local time: 2025-11-25 16:49:37]
+--------------------------------------------------------------------------------------------------+
0/0/0/1         Rx         30sec          0               0               0               0
       0               0               0
0/0/0/1         Rx         5mins          0               0               0               0
       0               0               0
0/0/0/1         Rx         1hour          0               0               0               0
       0               0               0
```

```
0/0/0/1         Rx         24hours        0               0               0               0
       0               0               0
0/0/0/1         Rx         Delta          0               0               0               0
       0               0               0
0/0/0/1         Rx         Cumm           0               0               0               0
       0               0               0
+-------------------------------------------------------------------------------------------------+
```

CHAPTER 14

Monitoring Fabric Links

This chapter describes methods for monitoring fabric links and troubleshooting them to ensure optimal network performance.

Fabric Link Keepalive Monitoring

Table 58: Feature History Table

Feature Name	Release Information	Feature Description
Fabric Link Keepalive Monitoring	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I

Feature Name	Release Information	Feature Description
Fabric Link Keepalive Monitoring	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
Fabric Link Keepalive Monitoring	Release 24.2.11	This feature allows you to monitor and identify the fabric links that are down due to failure to receive keep-alive messages. If a fabric link doesn't receive the keep-alive message, the Cisco IOS XR software performs a port-reset action and tries to activate the fabric link. This feature is enabled by default. You also have the option to disable the maximum port-reset threshold value of five, which causes the link to flap again, but we recommend you avoid using this command unless you have evaluated its impact on your traffic flow. This feature introduces the **hw-module fabric-tsmon-port-reset disable** command, which disables the maximum port-reset threshold value.

Fabric links in a router are high-speed connections that interconnect the internal components of a modular router, facilitating efficient data transfer and communication within the routing infrastructure.

The connection between a line card (LC) and a fabric card (FC) in a modular network device is a high-speed interface that enables the forwarding of data packets from the network ports on the line card through the switch fabric on the fabric card for routing to their destination.

These are the control messages exchanged between LC and FC specific NPU devices.

Prior to Release 24.2.11, if a fabric link doesn't receive the keep-alive message, it continues to flap until it becomes active or is manually shut down by admin.

From Release 24.2.11, if the fabric link doesn't receive the keep-alive messages, the Cisco IOS XR software performs the following steps to establish the connection between the fabric links:

1. Failure to receive keep-alive messages triggers a keep-alive (tsmon) interrupt.
2. The Cisco IOS XR software performs a port reset and tries to recover the fabric link.
3. If five interrupts occur continuously or randomly within 24 hours, the Cisco IOS XR software will permanently shut down the fabric link on the sixth interrupt and display the following syslog message.

   ```
   Router# Jan 19 11:31:37.899 UTC: npu_drvr[248]: %FABRIC-NPU_DRVR-3-NPU_CPA_GEN_ERR_INFO
    : TS_MON flap crossed threshold: Asic 2, link 180, event 20
   ```

Note The syslog message is displayed only after the fabric link is permanently shut down, which is after the sixth interrupt.

By default, each port is configured with a threshold that allows for five interrupts within a 24-hour period. If a sixth interrupt occurs within this timeframe, the Cisco IOS XR software shuts down the flapping fabric link. In the section View Fabric Link Interrupt, on page 325, you can see that the fabric link is shut down during the sixth tsmon interrupt within 24 hours.

You also have the option to disable the port-reset threshold value using the **hw-module fabric-tsmon-port-reset disable** command, which disables the fabric link shutdown action. For more information, see the section Disable the Port Reset Threshold Value, on page 326.

After disabling the port-reset threshold value, the Cisco IOS XR software doesn't shutdown the fabric link even after interrupts cross the threshold value; instead, it keeps flapping the fabric link and logs the asic-errors for the interrupt.

View Fabric Link Interrupt

If a fabric link flaps or shuts down due to failure to receive keep-alive messages, you can use the **show asic-errors** command to see the interrupt information:

```
Router# show asic-errors npu 8 all location 0/rp0/CPU0
************************************************************
*                       Link Errors                        *
************************************************************
8000, 8808-FC1, 0/FC4, npu[8]
Name            : slice[5].ifg[0].mac_pool8[1].rx_link_status_down.rx_link_status_down6
Block ID        : 0x288
Addr            : 0x100
Leaf ID         : 0x51002006
Thresh/period(s): 10/day
Error count     : 1
Last N errors   : 1
--------------------------------------------------------------
First N errors.
@Time, Error-Data
------------------------------------------
Jun  3 22:13:18.973101
        Error description: Id:193 Bit:0x6 Action:none LINK_DN: (Slc/Ifg/FstSer):(5/0/14)
RxLnDnSt:1 RxRmtLnStDn:0 PcsLnStDn:1 PcsAlnStDn:1 HiBer:0 HiSerIntr:0 SigLos:0 1 0 0 0 0 0
 0 FifoOverflow:0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  Link:174
```

```
NTS-22:13:18.973636 STS-22:13:18.973101 TEMP_SENSOR:43.5C
--------------------------------------------------------------
8000, 8808-FC1, 0/FC4, npu[8]
Name            : ts_mon.slice_interrupt_register[5].link_down_interrupt
Block ID        : 0x40f
Addr            : 0x105
Leaf ID         : 0x81e020a0
Thresh/period(s): 5/day
Error count     : 6
Last clearing   : Mon Jun  3 22:13:18 2024
Last N errors   : 6
--------------------------------------------------------------
First N errors.
@Time, Error-Data
------------------------------------------
Jun  3 22:13:18.945593
        Error description: Id:192 Bit:0x0 Action:port-reset LINK_FABRIC_DN:
(Slc/Ifg/FstSer):(5/0/14) Link:174 NTS-22:13:18.945716 STS-22:13:18.945593
--------------------------------------------------------------
```

The above sample displays only a part of the actual output; the actual output displays more details.

Disable the Port Reset Threshold Value

To disable the port-reset threshold value, use the **hw-module fabric-tsmon-port-reset disable** command.

You can't activate the fabric links that are down due to port-reset action crossing thresholds using the **hw-module fabric-tsmon-port-reset disable** command. The fabric links that are down must be debugged with CISCO.

Caution We recommend that you troubleshoot and resolve the reason for the fabric port shutdown instead of using the **hw-module fabric-tsmon-port-reset disable** command to prevent the fabric port shutdown.

```
Router# configuration terminal
Router(config)# hw-module fabric-tsmon-port-reset disable
Router(config)# commit
```

Fabric link health and capacity checks

The fabric link health and capacity checks introduce a single command dashboard that:

- eliminates the need for multiple CLIs to diagnose the fabric subsystem.
- monitors and diagnose fabric link quality, and
- allows for network bandwidth estimation and planning.

Table 59: Feature History Table

Feature Name	Release Information	Feature Description
Fabric link health and capacity monitoring with a single command	Release 25.3.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) In this release, we've introduced a single command that monitors fabric link quality, monitors capacity, and allows you to diagnose hardware and software issues. Previously, users encountered issues like links stopping, flapping, or dropping packets due to poor quality. These issues often required collecting multiple Command Line Interface (CLI) outputs and using external scripts for analysis. The feature introduces these changes: **CLI** • **show controller fabric dashboard** • **show controller fabric dashboard capacity** • **show controller fabric dashboard link-all**

The **show controller fabric dashboard** command offers these options to display fabric link health and capacity information:

Purpose	Command	Description
Display NPU capacity and problematic links	**show controller fabric dashboard**	This command displays each NPU's capacity and details only down or potentially bad fabric links. It indicates if the issue is a software state or a physical link quality problem.
Display NPU capacity only	**show controller fabric dashboard capacity**	This command displays only each NPU's available fabric capacity.
Display all fabric links information	**show controller fabric dashboard link-all**	This command provides each NPU's capacity and quality information for all fabric links.

Benefits of single command for fabric link health

These are the benefits of having a single Command Line Interface (CLI) for checking the fabric health:

- Simplifies diagnosis: eliminates the need for multiple CLIs to diagnose the fabric health.
- Aids planning: allows for network bandwidth estimation and planning.
- Identifies issues: identifies the down or potentially bad fabric links.
- Pinpoints root cause: helps identify whether fabric link issues originates from MAC (Media Access Control) problems or software protocol issues.

Restrictions for fabric link checks

These restrictions apply to fabric link checks:

- The **show controller fabric dashboard** command is only available on Cisco 8800 Series Routers.
- Periodic monitoring of fabric links can increase CPU usage on the active Route Processor (RP). You may need to adjust the caching timer interval based on CPU usage in fully loaded systems.

CHAPTER 15

Inbuilt Traffic Generator for Network Diagnostics

A traffic generator is a tool that is used to

- inject traffic onto a network for other devices to consume
- test the network to optimize performance, and
- troubleshoot network issues.

The Cisco 8000 Series Fixed Routers and Line Cards provides an inbuilt traffic generator in the Network Processing Unit (NPU) . This chapter shows you how to set up and run the inbuilt traffic generator.

Overview of the Inbuilt Traffic Generator

Table 60: Feature History Table

Feature Name	Release Information	Description
Inbuilt Traffic Generator for Network Diagnostics	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I

Feature Name	Release Information	Description
Inbuilt Traffic Generator for Network Diagnostics	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
Inbuilt Traffic Generator for Network Diagnostics	Release 24.2.11	By introducing an inbuilt traffic generator in the Network Processing Unit (NPU) of line cards (LCs) of distributed systems and route processors (RPs) of fixed routers, we've ensured that the traffic generator is always available for network diagnostics. You also don't face compatibility issues because the traffic generator is inbuilt and easy to maintain. Previously, connecting an external traffic generator was necessary to inject packets to test networks. This feature introduces these changes: **CLI:** • **diagnostic packet-generator create** • **diagnostic packet-generator start** • **diagnostic packet-generator stop** • **diagnostic packet-generator delete** • **show diagnostic packet-generator status**

The inbuilt traffic generator is implemented within the Cisco Silicon One Application-Specific Integrated Circuit (ASIC), the Network Processing Unit (NPU) of linecards, and fixed routers of the Cisco 8000 Series Routers.

Caution Don’t run the inbuilt traffic generator on a live network unless you are fully aware of the impact of packets injected. Injecting packets into a live network may result in network outages.

The inbuilt traffic generator works in three modes:

- **Ingress mode:** In this mode, the traffic generator injects ingress packets to an interface as if received from an upstream router. The NPU processes these packets normally and sends them to the egress interface based on looking up the Forwarding Information Base (FIB) programmed in the NPU. The NPU may encapsulate the packet and rewrite the packet headers before sending the packet out of the egress interface.
- **Egress mode:** In this mode, the traffic generator injects packets directly to an egress interface. If the egress interface is on another line card, the router sends the injected packet through the fabric and to another NPU where the egress interface is located. The router sends these packets as-is from the egress interface. So, you must be careful to formulate these packets with a valid Layer 2 header so that the downstream routers can process the packets.
- **Raw mode:** This mode is an advanced traffic generator mode where the user has full control. In this mode, the user can control the packets within the NPU by defining the inject headers of the packet.

Caution Improper use of raw mode could cause unexpected behavior, such as NPU lock-up. It is recommended that only Cisco engineers create traffic generator instances in raw mode.

Restrictions of the Inbuilt Traffic Generator

The following restrictions apply to the inbuilt traffic generator:

- The traffic generator does not support Link Layer Topology Discovery (LLTD) protocol for a scapy packet.
- The packet counters displayed in the show command outputs of the traffic generator CLIs does not increment if the router drops the packet.
- On linecards and fixed routers with Q100 based Silicon One ASICs, the traffic generator does not support capturing ingress packets.
- The inbuilt traffic generator does not retain the instance or the data if the line card or slot gets reloaded.
- When running the traffic generator in ingress mode, the **input packets** counter does not reflect an increase in the **show interface** command output for the ingress interface. However, the **input packets** counter of the **show interface accounting** command output does show an increment for the same ingress interface.
- Conversely, in egress mode, the **output packets** counter is incremented in the **show interface** command output for the egress interface. But the **output packets** counter of the **show interface accounting** command output does not display this increment for the same egress interface.

Guidelines for the Inbuilt Traffic Generator

Follow these guidelines while using the inbuilt traffic generator:

- You can create and control multiple traffic generator sessions at the same time, each with a different name.
- You can turn on packet capture when you create the traffic generator session. In this case, the router punts the traffic packets to the CPU instead of sending them out on the wire.
- All variations of the Cisco Silicon One ASIC support this inbuilt traffic generator.
- Per NPU, you can configure a maximum of 585 flows with packet size smaller or equal to 112 bytes and upto 8 flows for larger packets.
- You can configure a maximum traffic rate of 6.8 Mpps or 33.2 Gbps.
- You can configure upto 8 different configurable transmit rates shared across flows, that is, from 1 pps to 6.8 Mpps. Since the hardware doesn't support every single rate from 1 to 6.8Mpps, it will round off any unsupported rate you configured to the nearest hardware rate, with a maximum difference of 2%. You can view the rate applied in hardware with the **show diagnostic packet-generator status** command.
- You can configure a maximum packet size of 608 bytes on line cards and routers with the Q100 or Q200 based Silicon One ASICs.
- You can view per flow transmit statistics.
- The traffic generator can inject packets in egress, or ingress, or both directions at the same time.

Guidelines to Define Packets for the Traffic Generator

Few guidelines to keep in mind while defining packets to be generated by the traffic generator:

- You can define packets either at command line or within a file. The maximum length for packets at command line is 255 characters. For larger packets, define the packets within a file.
- While defining the packet within a file, you can use a pcap (packet capture) file with a .pcap suffix, or a text file.

  Pcap is a common format for storing packet captures. A pcap file includes an exact copy of every byte of every packet as seen on the network, including OSI layers 2-7.

  The packets defined in a text file can either be a hexadecimal string or as a python scapy script. Scapy is a packet manipulation tool, written in python, that can forge or decode packets, send them on the wire, capture them, and so on.
- Ensure to remove white-spaces and line-breaks from the hexadecimal or scapy specification of the packet.

Set-up and Run the Inbuilt Traffic Generator

The following steps show you how to set-up and run the inbuilt traffic generator:

1. **Create traffic generator instance:** In this step, you'll create the traffic generator instance with the required rate, duration, traffic generator mode, packet format, location, and so on, using the **diagnostic packet-generator create** command.

   By default, packet capture is turned off. You can turn on packet capture in this step using the optional keyword **capture**.

   The following examples show you how to create the traffic generator instance in ingress and egress mode:

   - **Ingress mode:**

     ```
     Router# diagnostic packet-generator create t1 rate 100 duration 60 packet
     IP(src="32.0.0.1",dst="22.0.0.1",ttl=64)/UDP()/Raw(load="a"*100) ingress interface
     FourHundredGigE0/0/0/1 capture location 0/RP0/CPU0
     OK
     ```

   - **Egress mode:**

     ```
     Router# diagnostic packet-generator create t1 rate 100 duration 60 packet
     Ether(src="A:B:C:D:E:F",dst="1:2:3:4:5:6")/IP(src="32.0.0.1",dst="109.0.0.101",ttl=64)/Raw(load="f"*100)
      egress interface fourHundredGigE0/0/0/0 capture location 0/RP0/CPU0
     OK
     ```

2. **Start traffic generator instance:** The traffic generator is in an inactive state until you execute the **diagnostic packet-generator start** command to start injecting packets.

   ```
   Router# diagnostic packet-generator start t1 location 0/RP0/CPU0
   OK
   ```

3. **Verify the status of the traffic generator:** Execute the **show diagnostic packet-generator status** command to view the traffic generator status and the packet generated.

   ```
   Router# show diagnostic packet-generator status t1 location 0/RP0/CPU0
   0/RP0/CPU0:
   Name           Run_State    Type    Capture  Set_Rate(pps)  Applied_Rate(pps)
   Duration(sec)  TC  Phy_Interface  NPU  Slice  IFG  Packets        Bytes
   ======================================================================================
   T1             Running      Ingress True     100           101              60
         0   FH0/0/0/1      0    4      1    209            45144
   --------------------------------------------------------------------------------------
   Packet Details:
   ###[ Ethernet ]###
     dst       = 78:bf:d2:07:10:08
     src       = 00:00:00:00:00:01
     type      = IPv4
   ###[ IP ]###
        version   = 4
        ihl       = 5
        tos       = 0x0
        len       = 128
        id        = 1
        flags     =
        frag      = 0
        ttl       = 64
        proto     = udp
        chksum    = 0x446b
        src       = 32.0.0.1
        dst       = 22.0.0.1
        \options   \
   ###[ UDP ]###
           sport     = domain
           dport     = domain
           len       = 108
   ```

```
        chksum    = 0xc3a5
###[ DNS ]###
           id        = 24929
           qr        = 0
           opcode    = 12
           aa        = 0
           tc        = 0
           rd        = 1
           ra        = 0
           z         = 1
           ad        = 1
           cd        = 0
           rcode     = format-error
           qdcount   = 24929
           ancount   = 24929
           nscount   = 24929
           arcount   = 24929
           qd        = ''
           an        = ''
           ns        = ''
           ar        = ''
###[ Raw ]###
              load      =
'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa'
```

4. **View captured packet:** If you have enabled packet capture, the router punts the packets, that are ready to be sent out, to the CPU of the egress line card. You can view the captured packets with the **show captured packets** command. This CLI limits the display of the packet to 256 bytes.

```
Router# show captured packets traps all location 0/RP0/CPU0
Captured packets are also saved in /harddisk:/nputrap_pcap_traps_all_loc_RP0.txt
***** Packet 1 (len 236) *****
Captured at: 2024-01-17 18:51:33.486

Internal header + packet:
4e41500100014e41500100117102080000a0c00ffff003cbffff3ffff0000028e
0ca000000000000000004e41500100014e41500100118100663271020803143c
000008ffff000030003c000011121314151617050000000000123456789abd00ca
fedead00810000000710326200000c00000000000000000000000078bfd20710
08000000000001080045000080000100004011446b2000000116000001003500
35006cc3a5616161616161616161616161616161616161616161616161616161
6161616161616161616161616161616161616161616161616161616161616161
616161616161616161616161

DMAC: 4e:41:50:01:00:01
SMAC: 4e:41:50:01:00:11
etype: NPU-PUNT
Punt header:
next_hdr: 0x1 (Ethernet)
fwd_hdr_type: 0x1 (Ethernet)
next header offset: 0
punt_source: 0xa (OUTBOUND_MIRROR)
Trap code: 12 (Refer to the traps output)
Ingress port details:
--------------------------
src_sys_port: 0xffff
src_logical_port: 0x3ffff
is_l2_punt: Yes
--------------------------
Egress port details:
--------------------------
dst_sys_port: 0x3c
```

```
dst_logical_port: 0x3ffff
--------------------------
switch_or_vrf_id: 0x0 (default)
timestamp: 0x28e0ca000000000
counterstamp: 0x0



Ethernet payload

4e41500100014e415001001181006632710208031 43c000008ffff000030003c
00001112131415161705000000001 23456789abd00cafedead00810000007103
26200000c0000000000000000000000000 78bfd2071008000000000001080045
00008000010000401 1446b200000011600000100350035006cc3a56161616161
6161616161616161616161616161616161616161616161616161616161616161
6161616161616161616161616161616161616161616161616161616161616161
6161

DMAC: 4e:41:50:01:00:01
SMAC: 4e:41:50:01:00:11
etype: 802.1Q
802.1Q hdr [User field:03 CFI:00 VLAN-ID:1586]
etype: NPU-PUNT

***** Packet 1 end *****
```

5. **Stop and delete traffic generator instance:** After completing the traffic testing, execute the **diagnostic packet-generator stop** command to stop injecting packets. Following that, free up resources by deleting the traffic generator instance with the **diagnostic packet-generator delete** command.

```
Router# diagnostic packet-generator stop t1 location 0/RP0/CPU0
OK
Router# diagnostic packet-generator delete t1 location 0/RP0/CPU0
OK
```

CHAPTER 16

Implementing Audit Monitoring

This chapter explains the audit monitoring and logging capabilities available on Cisco 8000 Series routers and how to configure audit monitoring.

Audit monitoring

Audit monitoring is a security and compliance feature that

- integrates with the Linux audit daemon to monitor system events as defined by audit rules
- writes logs in local log files to record changes in the monitored resources, and
- allows forwarding of audit logs to a remote syslog server.

Linux audit daemon is a user-space component of the Linux auditing system that

- tracks and logs system calls, file accesses, user actions, and other events as specified by audit rules, and
- provides administrators with insights to detect suspicious behavior and maintain system integrity.

An audit rule is a router configuration that

- specifies which files, directories, or system events should be monitored, and
- can be activated on the router by enabling their respective audit rule-groups.

An audit rule-group is a collection of predefined audit rules that

- can be enabled or disabled together as a single unit, and
- allows monitoring specific categories of system events, files, or directories, such as changes in user group configuration files, DNS client files, or XR software

Table 61: Feature History Table

Feature name	Release information	Feature description
Implementing audit monitoring	Release 25.3.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q100, Q200, P100], 8700 [ASIC: P100, K100], 8010 [ASIC: A100]); Centralized Systems (8600 [ASIC:Q200]) ; Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) You can enhance your router's security and compliance by enabling audit monitoring. This feature lets you configure predefined rules that enable the router to monitor, log, and optionally forward audit logs to a remote syslog server for centralized analysis and incident response. The feature introduces these changes: CLI: • **linux security audit** • **show linux security audit**

Audit logs for security monitoring

Network administrators enable audit rule-groups to track changes to sensitive files, monitor system calls, and observe other critical activities.

Audit rules establish what to watch, while audit logs capture and document every relevant occurrence, ensuring a complete and actionable history of system activity.

For example, an audit rule that monitors changes to **/etc/passwd** file creates an audit log entry each time this file is modified.

How audit monitoring works

Summary

These are the key components involved in this feature:

- Network Administrator: The user who initiates configurations via CLI.
- Linux audit daemon : The process that monitors system activity according to the installed rules and writes audit event logs.
- Local rsyslog daemon: The process that forwards logs to a remote syslog server.
- Remote syslog server: The external server that maintains the logs generated by the router.

The Linux audit daemon is the core service that actually performs audit monitoring and logging, based on the audit rules configured by the network administrator. It operates at the operating system level on each node, such as line cards and route processors.

Workflow

These stages describe how audit monitoring and logging works.

1. The network administrator enables audit monitoring via CLI.
2. The router software receives the configurations, applies the relevant audit rules, and ensures these rules are distributed to all appropriate nodes.
3. On each node, the Linux audit daemon actively monitors system events as defined by the audit rules and writes the logs to a local log file at **/var/log/audit/audit.log**.
4. If the network administrator has enabled log forwarding, the audit logs are sent to the local rsyslog daemon, which then forwards the logs to a remote syslog server.

Guidelines for audit monitoring

Granularity of audit rules

- You can enable or disable audit rules only at the group level, not individually within a group.
- Regularly review the status of audit rules and audit log forwarding to ensure monitoring remains effective.

Security of audit logs and syslog servers

- Allow only users with appropriate administrative privileges to configure or view Linux security audit settings.
- Protect access to audit logs and syslog servers to prevent unauthorized access or tampering.

Log forwarding to remote syslog servers

- Confirm that the remote syslog server is reachable and properly configured before enabling log forwarding.
- The router does not support Transport Layer Security (TLS) for forwarding audit logs. It forwards audit logs to remote syslog servers using User Datagram Protocol (UDP). Use only trusted network segments for remote syslog servers.

Notes about audit log storage

- The router stores audit logs locally at **/var/log/audit/audit.log**, unless you enable log forwarding.
- By default, the system rotates up to five audit log files, each up to 8 MB in size.

Configure audit monitoring

Follow this procedure to configure audit monitoring for specific system events by enabling the relevant audit rule groups.

Procedure

Step 1 Execute the **linux security audit monitor** *<group-keyword>* command, to enable a group of audit rules.

Example:

```
Router# conf t
Router(config)# linux security audit monitor xr-software
Router(config)# linux security audit monitor user-group-config-files
Router(config)# commit
```

Step 2 Run the **show linux security audit monitor status** command, to verify the general status of all active audit rule groups.

Example:

```
Router# show linux security audit monitor status
key name: xr-software                status: enabled
rules:
-a always,exit -F arch=b64 -F dir=/pkg/bin -F perm=wa -k xr_bin_changes
-a always,exit -F arch=b64 -F dir=/pkg/sbin -F perm=wa -k xr_sbin_changes
-a always,exit -F arch=b64 -F dir=/pkg/lib -F perm=wa -k xr_lib_changes
--------------------------------------------------------------
key name: user-group-config-files    status: enabled
rules:
-a always,exit -F arch=b64 -F path=/etc/passwd -F perm=wa -k passwd_changes
-a always,exit -F arch=b64 -F path=/etc/shadow -F perm=wa -k shadow_changes
-a always,exit -F arch=b64 -F path=/etc/group -F perm=wa -k group_changes
-a always,exit -F arch=b64 -F path=/etc/sudoers -F perm=wa -k sudoers_changes
--------------------------------------------------------------
```

Step 3 (Optional) Execute the **linux security audit logging syslog** command, to enable forwarding of audit logs.

Example:

```
Router# conf t
Router(config)# linux security audit logging syslog
Router(config)# commit
```

Step 4 (Optional) Execute the **logging** *remote-server-ip* **vrf** *vrf-name* command, to configure the remote syslog server.

Example:

```
Router# conf t
Router(config)# logging 10.0.1.2 vrf default port 514
Router(config)# commit
```

Step 5 (Optional) Run the **show linux security audit logging syslog** command, to verify whether audit log forwarding is enabled and to view the configured remote syslog server.

Example:

```
Router# show linux security audit logging syslog
status: enabled
syslog-server(s):
ipaddr: 10.0.1.2 vrf: default port: 514
ipaddr: 2001:db8::1 vrf: default port: 514
```