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

First Published: 2019-05-01

Last Modified: 2019-12-01

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

CONTENTS

Preface

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

Changes to this Document

Note *This software release has reached end-of-life status. For more information, see the* End-of-Life and End-of-Sale Notices.

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

Date	**Summary**
May 2019	Initial release of this document.
December 2019	Republished for 6.6.3.

Communications, Services, and Additional Information

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

Cisco Bug Search Tool

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

CHAPTER 1

New and Changed Feature Information

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

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

Feature	Description	Changed in Release	Where Documented
TWAMP-Light	This feature was introduced in this release.	Release 6.6.25	TWAMP-Light, on page 90

CHAPTER 2

Implementing System Logging

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

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

Feature History for Implementing System Logging

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

Implementing System Logging

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

Syslog Message Format

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

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

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

Table 1: Format of Syslog Messages

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

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

Syslog Message Severity Levels

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

Table 2: Syslog Message Severity Levels

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

Prerequisites for Configuring System Logging

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

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

Configuring System Logging

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

Configuring Logging to the Logging Buffer

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

Configuration Example

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

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

Configuring Logging to a Remote Server

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

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

Table 3: Features for Managing Syslog Messages

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

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

Configuration Example for Logging to Syslog Server

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

```
Router# configure
Router(config)# logging 209.165.201.1 vrf default
Router(config)# logging facility kern (optional)
Router(config)# logging hostnameprefix 203.0.113.1 (optional)
Router(config)# logging source-interface HundredGigE 0/0/0/3 (optional)
Router(config)# commit
```

Configuration Example for Logging to SNMP Server

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

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

For more information on SNMP server configurations, see the *Configuring Simple Network Management Protocol* chapter in the *System Management Configuration Guide for Cisco NCS 5500 Series Routers*

Related Topics

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

Configuring Logging to Terminal Lines

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

Configuration Example

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

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

Modifying Logging to Console Terminal

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

Configuration Example

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

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

Modifying Time Stamp Format

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

Configuration Example

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

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

Suppressing Duplicate Syslog Messages

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

Configuration Example

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

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

Archiving System Logging Messages to a Local Storage Device

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

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

Table 4: Commands Used to Set Syslog Archive Attributes

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

Configuration Example

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

```
Router#conf t
Router(config)#logging archive
Router(config-logging-arch)#device disk1
Router(config-logging-arch)#frequency weekly
Router(config-logging-arch)#severity warnings
Router(config-logging-arch)#archive-length 6
Router(config-logging-arch)#archive-size 50
Router(config-logging-arch)#file-size 10
Router(config-logging-arch)#commit
```

Platform Automated Monitoring

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

PAM Events

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

The format of the .tgz file is: *PAM-<platform>-<PAM event>-<node-name>-<PAM process>-<YYYYMMDD>-<checksum>.tgz*.For example, *PAM-ncs5500-crash-xr_0_RP0_CPU0-ipv4_rib-2016Aug16-210405.tgz* is the file collected when PAM detects a process crash.

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

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

The sections below describe the main PAM events:

Crash Monitoring

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

```
RP/0/RP0/CPU0:Aug 16 21:04:06.442 : logger[69324]: %OS-SYSLOG-4-LOG_WARNING : PAM detected
 crash for ipv4_rib on 0_RP0_CPU0.
All necessary files for debug have been collected and saved at
0/RP0/CPU0 :
harddisk:/cisco_support/PAM-ncs5500-crash-xr_0_RP0_CPU0-ipv4_rib-2016Aug16-210405.tgz
Please copy tgz file out of the router and send to Cisco support. This tgz file will be
removed after 14 days.)
```

Traceback Monitoring

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

```
RP/0/RP0/CPU0:Aug 16 21:42:42.320 : logger[66139]: %OS-SYSLOG-4-LOG_WARNING : PAM detected
 traceback for ipv4_rib on 0_RP0_CPU0.
All necessary files for debug have been collected and saved at
0/RP0/CPU0 :
harddisk:/cisco_support/PAM-ncs5500-traceback-xr_0_RP0_CPU0-ipv4_rib-2016Aug16-214242.tgz
Please copy tgz file out of the router and send to Cisco support. This tgz file will be
```

```
removed after 14 days.)
```

Memory Usage Monitoring

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

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

```
RP/0/RP0/CPU0:Aug 17 05:13:32.684 : logger[67772]: %OS-SYSLOG-4-LOG_WARNING : PAM detected
 significant memory increase
(from 13.00MB at 2016/Aug/16/20:42:41 to 28.00MB at 2016/Aug/17/04:12:55) for
pam_memory_leaker on 0_RP0_CPU0.
All necessary files for debug have been collected and saved at
0/RP0/CPU0 :
harddisk:/cisco_support/PAM-ncs5500-memory_leak-xr_0_RP0_CPU0-pam_memory_leaker-2016Aug17-051332.tgz

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

CPU Monitoring

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

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

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

```
RP/0/RP0/CPU0:Aug 16 00:56:00.819 : logger[68245]: %OS-SYSLOG-4-LOG_WARNING : PAM detected
 CPU hog for cpu_hogger on 0_RP0_CPU0.
All necessary files for debug have been collected and saved at 0/RP0/CPU0 :
harddisk:/cisco_support/PAM-ncs5500-cpu_hog-xr_0_RP0_CPU0-cpu_hogger-2016Aug16-005600.tgz
(Please copy tgz file out of the router and send to Cisco support. This tgz file will be
removed after 14 days.)
RP/0/RP0/CPU0:Jun 21 15:33:54.517 : logger[69042]: %OS-SYSLOG-1-LOG_ALERT : PAM detected
ifmgr is hogging CPU on 0_RP0_CPU0!
```

File System Monitoring

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

```
RP/0/RP0/CPU0:Jun 20 13:59:04.986 : logger[66125]: %OS-SYSLOG-4-LOG_WARNING : PAM detected
 /misc/config is full on 0_1_CPU0
(please clean up to avoid any fault caused by this). All necessary files for debug have
been collected and saved at
0/RP0/CPU0 : harddisk:/cisco_support/PAM-ncs5500-disk_usage-xr_0_1_CPU0-2016Jun20-135904.tgz

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

Disable and Re-enable PAM

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

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

- From Cisco IOS XR Command Line Interface:

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

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

- From router shell prompt:

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

Disable PAM

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

For local RP:

```
Router# disable-pam
```

For all RPs:

```
Router# disable-pam
```

Re-enable PAM

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

For local RP:

```
Router# enable-pam
```

For all RPs:

```
Router# enable-pam
```

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

Data Archiving in PAM

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

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

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

Files Collected by PAM Tool

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

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

Event Name	Commands and Files Collected by PAM
Process crash	• **show install active** • **show platform** • **show version** • core (gz) file • core.txt file

Event Name	Commands and Files Collected by PAM
Process traceback	• **show dll** • **show install active** • **show logging** • **show platform** • **show version**
Memory leak	• **show install active** • **show platform** • **show version** • core (gz) file • dumpcore running • continuous memory usage snapshots
Show logging event	• **show install active** • **show logging** • **show platform** • **show version** • core (gz) file • core.txt file
CPU hog	• **follow process** • **pstack** • **show dll** • **show install active** • **show platform** • **show version** • **top -H** • core (gz) file • CPU usage snapshots

Event Name	Commands and Files Collected by PAM
Disk usage	• **show install active** • **show platform** • **show version** • console log • core (gz) file • Disk usage snapshots

CHAPTER 3

Implementing Alarm Log Correlation

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

Implementing Alarm Log Correlation

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

Prerequisites for Implementing Alarm Log Correlation

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

Information About Implementing Alarm Log Correlation

Alarm Logging and Debugging Event Management System

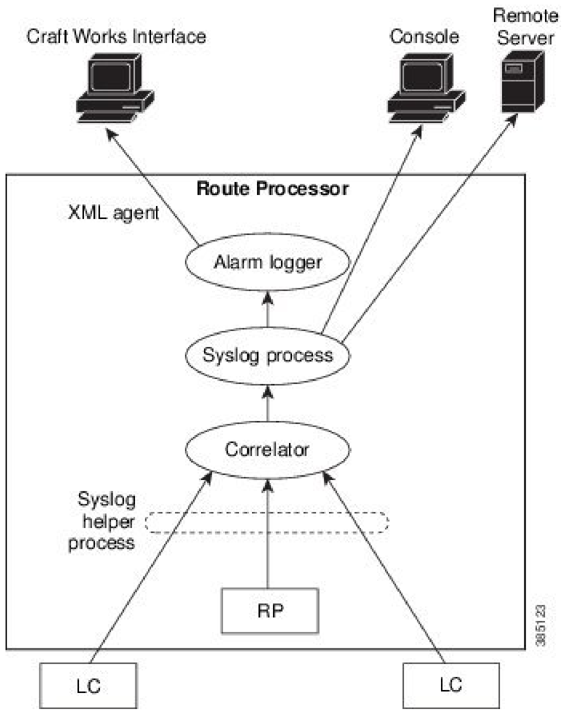
Cisco IOS XR Software Alarm Logging and Debugging Event Management System (ALDEMS) is used to monitor and store alarm messages that are forwarded by system servers and applications. In addition, ALDEMS correlates alarm messages forwarded due to a single root cause.

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

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

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

Figure 1: ALDEMS Component Communications



Correlator

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

System Logging Process

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

Alarm Logger

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

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

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

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

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

Configuring Alarm Log Correlation

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

Configuring Logging Correlation Rules

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

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

Configuration Example

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

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

Configuring a Logging Correlation Rule Set

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

Configuration Example

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

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

Correlating a Root Cause and Non Root Cause Alarms

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

Configuration Example

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

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

Configuring Hierarchical Correlation Rule Flags

Hierarchical correlation is when a single alarm is both a root cause for one correlation rule and a non-root cause for another rule, and when alarms are generated resulting in a successful correlation associated with both rules. What happens to a non-root-cause alarm depends on the behavior of its correlated root-cause alarm. There are cases in which you want to control the stateful behavior associated with these hierarchies and to implement flags, such as reparenting and reissuing of non-bistate alarms. For detailed information about hierarchical correlation and correlation flags, see Hierarchical Correlation, on page 22

Configuration Example

This example shows how to configure hierarchical correlation rule flags.

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

Configuring Logging Suppression Rules

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

Configuration Example

This example shows how to configure logging suppression rules.

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

Modifying Logging Events Buffer Settings

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

Configuration Example

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

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

Modifying Logging Correlation Buffer Settings

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

Configuration Example

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

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

Enabling Alarm Source Location Display Field for Bistate Alarms

Bistate alarms are generated by state changes associated with system hardware. The bistate alarm message format is similar to syslog messages. You can optionally configure the output to include the location of the actual alarm source, which may be different from the process that logged the alarm. For more information about bistate alarms see, Bistate Alarms, on page 22

Configuration Example

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

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

Configuring SNMP Correlation Rules

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

Configuration Example

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

```
RP/0/RP0/CPU0:Router# configure terminal
RP/0/RP0/CPU0:Router(config)# snmp-server correlator buffer-size 1024 (optional)
RP/0/RP0/CPU0:Router(config)# snmp-server correlator rule test rootcause A varbind A1 value
 regex RA1  nonrootcause  trap B varbind B1 index regex RB1
RP/0/RP0/CPU0:Router(config)# snmp-server correlator apply rule test host ipv4 address
1.2.3.4
RP/0/RP0/CPU0:Router(config)# commit
```

Configuring SNMP Correlation Ruleset

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

Configuration Example

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

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

Alarm Logging Correlation-Details

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

Correlation Rules

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

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

- Message category
- Message group
- Message code

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

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

Alarm Severity Level and Filtering

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

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

Table 5: Alarm Severity Levels for Event Logging

Severity Level	System Condition
0	Emergencies

Severity Level	System Condition
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.

CHAPTER 4

Onboard Failure Logging

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

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

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

Caution OBFL is activated by default in FRUs and can be deactivated by stopping the **obflmgr** process. Do not deactivate OBFL without specific reasons, because the OBFL data is used to diagnose and resolve problems in FRUs.

Prerequisites

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

Information About OBFL

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

- Boot time
- FRU part serial number

- OS version
- Temperature and voltage at boot
- Temperature and voltage history
- Total run time

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

Baseline Data Collection

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

Table 6: Data Types for Baseline Data Collection

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

Event-Driven Data Collection

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

Table 7: Data Types for Event-Driven Data Collection

	Details	
Environmental Factors	Temperature Value	Inlet and hot point temperature value change beyond the threshold set in the hardware inventory XML files.
	Voltage Value	An environmental reading is logged when the following temperature or voltage events occur: • Exceed the normal range • Change more than 10% • Return within range for more than five minutes. On reboot, these environmental readings are consolidated into a single environmental history record that shows the duration and extent out of normal range for a consecutive set of environmental readings.
Calendar Time	Cleared	The time when OBFL logging was cleared.
	Disabled	The time when OBFL logging was disabled.
	Reset to 0	The time when total line card runtime is reset to zero.

Supported Cards and Platform

FRUs that have sufficient nonvolatile memory available for OBFL data storage support OBFL. The following table provides information about the OBFL support for different FRUs on the Cisco NCS 5500 Series router.

Table 8: OBFL Support on Cisco NCS 5500 Series Router

Card Type	Cisco NCS 5500 Series Router
Route processor	Supported
Fabric cards	Supported
Line card	Supported
Power supply cards	Not Supported
Fan tray	Supported
System Controller	Supported

CHAPTER 5

Implementing Performance Management

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

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

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

Prerequisites for Implementing Performance Management

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

- You must install and activate the Package Installation Envelope (PIE) for the manageability software.
- You must be in a user group associated with a task group that includes the proper task IDs. The command reference guides include the task IDs required for each command. If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.
- You must have connectivity with a TFTP server.

Information About Implementing Performance Management

PM Functional Overview

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

- PM statistics server
- PM statistics collectors

PM Statistics Server

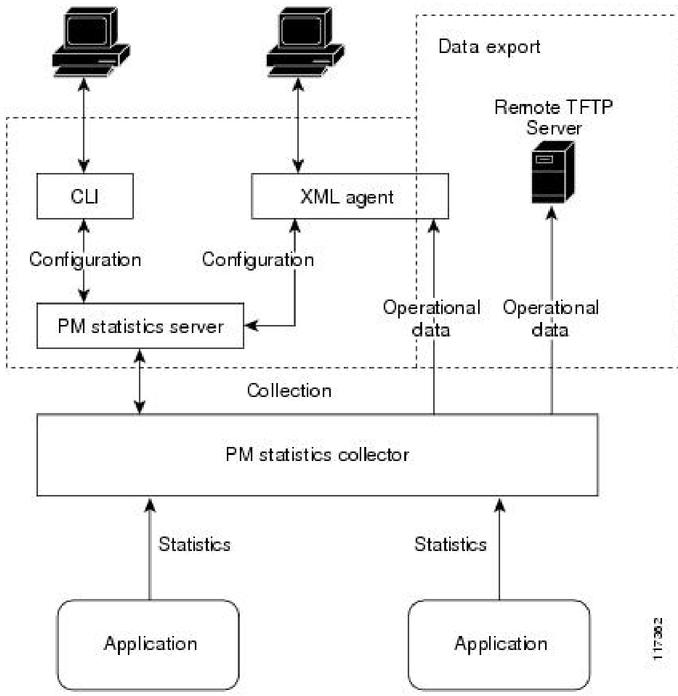
The PM statistics server is the front end for statistic collections, entity instance monitoring collections, and threshold monitoring. All PM statistic collections and threshold conditions configured through the command-line interface (CLI) or through XML schemas are processed by the PM statistics server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

PM Statistics Collector

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

Figure 2: PM Component Communications, on page 31 illustrates the relationship between the components that constitute the PM system.

Figure 2: PM Component Communications



PM Benefits

The PM system provides these benefits:

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

PM Statistics Collection Overview

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

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

Table 9: Entity Classes and Associated Instances

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

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

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

Binary File Format for Exporting PM Statistics

This sample describes the binary file format:

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

```
                                   StatCounterValue :8 Bytes ( for all counters)
                                   Repeat for Each StatCounterName
                               Repeat for Each Sample No(Time Interval)
                       Repeat for All Instances
               Repeat for All SubTypes
       Repeat for All Entities
```

Binary File ID Assignments for Entity, Subentity, and StatsCounter Names

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

Table 10: Binary Format Values and Keys

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

Note <ifName>—The length is variable. The first two bytes contain the size of the Instance ID; this is followed by the Instance ID string (that is, an Interface name).

<IpAddress>—4 bytes that contain the IP address.

<NodeProcessID>—64-bit Instance ID. The first 32 bits contain the node ID, and the second 32 bits contain the process ID.

<NodeID>—32-bit instance ID that contains the Node ID.

<process_instance>—The length is variable. The first two bytes contain the size of Instance ID followed by Instance ID string (that is, a process name).

Note The numbers in parenthesis (the numbers that are associated with each entity and subentity in Table 10: Binary Format Values and Keys, on page 33) denote the entity and subEntity IDs that are displayed in the TFTP File.

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

Table 11: Supported StatsCounters for Entities and Subentites

Entity	Subentity	StatsCounters
Node (1)	CPU (1)	NoProcesses
	Memory (2)	CurrMemory, PeakMemory
	Process (3)	PeakMemory, NoThreads
Interface (2)	Generic Counters (1)	InPackets, InOctets, OutPackets, OutOctets, InUcastPkts, InMulticastPkts, InBroadcastPkts, OutUcastPkts, OutMulticastPkts, OutBroadcastPkts, OutputTotalDrops, InputTotalDrops, InputQueueDrops, InputUnknownProto, OutputTotalErrors, OutputUnderrun, InputTotalErrors, InputCRC,InputOverrun, InputFrame
	Data Rate Counters (2)	InputDataRate, InputPacketRate, OutputDataRate, OutputPacketRate, InputPeakRate, InputPeakPkts, OutputPeakRate, OutputPeakPkts, Bandwidth
	Basic Counters	InPackets, InOctets, OutPackets, OutOctets, InputTotalDrops, InputQueueDrops,InputTotalErrors, OutputTotalErrors, OutputQueueDrops, OutputTotalErrors
BGP (3)	Peer (1)	InputMessages, OutputMessages, InputUpdateMessages, OutputUpdateMessages, ConnEstablished, ConnDropped, ErrorsReceived, ErrorsSent
MPLS (4)	LDP (4)	TotalMsgsSent, TotalMsgsRcvd, InitMsgsSent, InitMsgsRcvd, AddressMsgsSent, AddressMsgsRcvd, AddressWithdrawMsgsSent, AddressWithdrawMsgsRcvd, LabelMappingMsgsSent, LabelMappingMsgsRcvd, LabelWithdrawMsgsSent, LabelWithdrawMsgsRcvd, LabelReleaseMsgsSent, LabelReleaseMsgsRcvd, NotificationMsgsSent, NotificationMsgsRcvd KeepAliveMsgsSent, KeepAliveMsgsRcvd
OSPF (5)	v2protocol (1)	InputPackets, OutputPackets, InputHelloPackets, OutputHelloPackets, InputDBDs, InputDBDsLSA, OutputDBDs, OutputDBDsLSA, InputLSRequests, InputLSRequestsLSA, OutputLSRequests, OutputLSRequestsLSA, InputLSAUpdates, InputLSAUpdatesLSA, OutputLSAUpdates, OutputLSAUpdatesLSA, InputLSAAcks, InputLSAAcksLSA, OutputLSAAcks, OutputLSAAcksLSA, ChecksumErrors

Entity	Subentity	StatsCounters
	v3protocol (2)	InputPackets, OutputPackets, InputHelloPackets, OutputHelloPackets, InputDBDs, InputDBDsLSA, OutputDBDs, OutputDBDsLSA, InputLSRequests, InputLSRequestsLSA, OutputLSRequests, OutputLSRequestsLSA, InputLSAUpdates, InputLSAUpdatesLSA, OutputLSAUpdates, OutputLSAUpdatesLSA, InputLSAAcks, InputLSAAcksLSA, OutputLSAAcks, OutputLSAAcksLSA

Filenaming Convention Applied to Binary Files

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

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

How to Implement Performance Management

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

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

Configuration Examples

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

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

This example configures a local disk for PM statistics collection.

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

Configuring PM Statistics Collection Templates

PM statistics collections are configured through PM statistics collection templates. A PM statistics collection template contains the entity, the sample interval, and the number of sampling operations to be performed before exporting the data to a TFTP server. When a PM statistics collection template is enabled, the PM statistics collection gathers statistics for all attributes 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
```

Enabling PM Entity Instance Monitoring

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

Configuration Example

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

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

Configuring PM Threshold Monitoring Templates

The PM system supports the configuration of threshold conditions to monitor an attribute (or attributes) for threshold violations. Threshold conditions are configured through PM threshold monitoring templates. When a PM threshold template is enabled, the PM system monitors all instances of the attribute (or attributes) for the threshold condition configured in the template. If at end of the sample interval a threshold condition is matched, the PM system generates a system logging message for each instance that matches the threshold condition. For the list of attributes and value ranges associated with each attribute for all the entities, see Performance Management: Details, on page 38

Guidelines for Configuring PM Threshold Monitoring Templates

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

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

Configuration Example

This example shows how to create and enable a PM threshold monitoring template. In this example, a PM threshold template is created for the **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 12: 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 13: 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 0/3/0/24 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 0/3/0/24 default`
Interface Generic Counters	Use the **performance-mgmt apply monitor interface generic-counters** command to enable entity instance monitoring for an interface generic counters entity instance. **Syntax:** **`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 0/3/0/24 default`
MPLS LDP	Use the **performance-mgmt apply monitor mpls ldp** command to enable entity instance monitoring for an MPLS LDP entity instance. **Syntax:** **`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 6

Configuring and Managing Embedded Event Manager Policies

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

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

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

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

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

Prerequisites for Configuring and Managing Embedded Event Manager Policies

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

Information About Configuring and Managing Embedded Event Manager Policies

Event Management

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

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

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

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

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

System Event Processing

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

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

Embedded Event Manager Scripts

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

In addition the onboard Tcl scripts that come with the IOS XR operating system, users may write their own TCL-based policies. Cisco provides enhancements to the Tcl language in the form of Tcl command extensions that facilitate the writing of EEM policies. For more information about EEM Tcl command extensions, see Embedded Event Manager Policy Tcl Command Extension Categories, on page 53

Writing an EEM script includes the following steps:

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

Embedded Event Manager Policy Tcl Command Extension Categories

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

Table 14: Embedded Event Manager Tcl Command Extension Categories

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

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

Cisco File Naming Convention for Embedded Event Manager

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

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

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

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

Table 15: Two-Character Abbreviation Specification

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

Embedded Event Manager Built-in Actions

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

This table describes each EEM handler request or action.

Table 16: Embedded Event Manager Built-In Actions

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

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

Application-specific Embedded Event Management

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

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

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

Event Detection and Recovery

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

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

- System EEM events
- Application-defined events

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

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

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

These event detectors are supported:

System Manager Event Detector

The System Manager Event Detector has four roles:

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

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

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

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

Table 17: 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 18: Syslog Event Detector Event Monitoring Requests

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

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

None Event Detector

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

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

Watchdog System Monitor Event Detector

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

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

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

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

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

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

Table 19: Watchdog System Monitor Event Detector Requests

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

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

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

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

Distributed Event Detectors

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

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

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

Embedded Event Manager Event Scheduling and Notification

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

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

Reliability Statistics

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

Hardware Card Reliability Metric Data

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

Data maintained by the hardware card is as follows:

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

Process Reliability Metric Data

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

Process terminations include the following cases:

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

Data to be maintained by process is as follows:

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

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

How to Configure and Manage Embedded Event Manager Policies

Configuring Environmental Variables

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

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

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

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

Configuration Example

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

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

No. Name                             Value
1   _email_to                        beta@cisco.com
2   _cron_entry                      0-59/2 0-23/1 * * 0-7
3   _email_from                      beta@cisco.com
RP/0/RP0/CPU0:Router#
```

Registering Embedded Event Manager Policies

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

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

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

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

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

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

Configuration Example

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

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

How to Write Embedded Event Manager Policies Using Tcl

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

This section contains these tasks:

Registering and Defining an EEM Tcl Script

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

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

Configuration Example

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

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

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

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

Suspending EEM Policy Execution

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

Configuration Example

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

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

Specifying a Directory for Storing EEM Policies

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

Configuration Example

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

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

Programming EEM Policies with Tcl

Perform this task to help you program a policy using Tcl command extensions. We recommend that you copy an existing policy and modify it. There are two required parts that must exist in an EEM Tcl policy: the event_register Tcl command extension and the body. For detailed information about the Tcl policy structure and requirements, see EEM Policies Using TCL: Details, on page 74

SUMMARY STEPS

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**show event manager policy available** [**system** | **user**] **Example:** `RP/0/RP0/CPU0:Router# show event manager policy available`	Displays EEM policies that are available to be registered.
Step 2	Cut and paste the contents of the sample policy displayed on the screen to a text editor.	—
Step 3	Define the required event_register Tcl command extension.	Choose the appropriate event_register Tcl command extension for the event that you want to detect, and add it to the policy. The following are valid Event Registration Tcl Command Extensions: • event_register_appl • event_register_counter • event_register_stat • event_register_wdsysmon • event_register_oir • event_register_process • event_register_syslog • event_register_timer

	Command or Action	Purpose
		• event_register_timer_subscriber • event_register_hardware • event_register_none

Step 4	Add the appropriate namespace under the ::cisco hierarchy.	Policy developers can use the new namespace ::cisco in Tcl policies to group all the extensions used by Cisco IOS XR EEM. There are two namespaces under the ::cisco hierarchy. The following are the namespaces and the EEM Tcl command extension categories that belongs under each namespace:
		• ::cisco::eem
		  • EEM event registration
		  • EEM event information
		  • EEM event publish
		  • EEM action
		  • EEM utility
		  • EEM context library
		  • EEM system information
		  • CLI library
		• ::cisco::lib
		  • SMTP library
		Note Ensure that the appropriate namespaces are imported, or use the qualified command names when using the preceding commands.

Step 5	Program the must defines section to check for each environment variable that is used in this policy.	This is an optional step. Must defines is a section of the policy that tests whether any EEM environment variables that are required by the policy are defined before the recovery actions are taken. The must defines section is not required if the policy does not use any EEM environment variables. EEM environment variables for EEM scripts are Tcl global variables that are defined external to the 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 "_".

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

	Command or Action	Purpose
Step 8	Check the exit status to determine whether or not to apply the default action for this event, if a default action exists.	A value of zero means that the default action should not be performed. A value of nonzero means that the default action should be performed. The exit status is passed to subsequent policies that are run for the same event.
Step 9	Set Cisco Error Number (_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 15: Two-Character Abbreviation Specification, on page 54) for the first event specified, an underscore character part, and a descriptive field part further identifying the policy. • A filename suffix part defined as .tcl. For more details, see theCisco File Naming Convention for Embedded Event Manager, on page 54. Copy the file to the flash file system on the router—typically disk0:.
Step 11	**configure**	Enters global configuration mode.
Step 12	**event manager directory user** {**library** *path* | **policy** *path*} **Example:** `RP/0/RP0/CPU0:Router(config)# event manager directory user library disk0:/user_library`	Specifies a directory to use for storing user library files or user-defined EEM policies.
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.

	Command or Action	Purpose
		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.

SUMMARY STEPS

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

DETAILED STEPS

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

	Command or Action	Purpose

		`proc test1 {} {`
		`  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]]`

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

Step 5	Copy a user-defined EEM policy file written in Tcl to the directory used for storing user-defined EEM policies on the target router.	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 disk0:/eem_library`	Specifies the EEM user library directory; this is the directory to which the files in step 4 were copied.
Step 8	**event manager directory user policy** *path* **Example:** `RP/0/RP0/CPU0:Router(config)# event manager directory user policy disk0:/eem_policies`	Specifies the EEM user policy directory; this is the directory to which the file in step 5 was copied.
Step 9	**event manager policy** *policy-name* **username** *username* [**persist-time** [*seconds* | **infinite**] | **type** [**system** | **user**]] **Example:** `RP/0/RP0/CPU0:Router(config)# event manager policy libtest.tcl username user_a`	Registers a user-defined EEM policy.

	Command or Action	Purpose
Step 10	**event manager run** *policy* [*argument*] **Example:** `RP/0/RP0/CPU0:Router(config)# event manager run libtest.tcl`	Manually runs an EEM policy.
Step 11	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Creating an EEM User Tcl Package Index

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

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

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

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

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

SUMMARY STEPS

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

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

DETAILED STEPS

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

Step 3	**pkg_mkindex** *directory_name* ***.tcl**
	Example:
	`workstation% pkg_mkindex eem_library *.tcl`
		Use the **pkg_mkindex** command to create the pkgIndex file. The pkgIndex file contains a directory of all the packages contained in the Tcl library files. We recommend that you run the **pkg_mkindex** command inside a directory, because there can be only a single pkgIndex file in any directory and you may have other Tcl files to be grouped together. Running the **pkg_mkindex** command in a directory determines which Tcl package file or files are indexed using a specific pkgIndex.
		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.`

	Command or Action	Purpose
		`package ifneeded xmlrpc 0.3 [list source [file join $dir xmlrpc.tcl]]`
Step 4	Copy the Tcl package files from step 1 and the pkgIndex file from step 3 to the directory used for storing user library files on the target router.	—

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

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

	Command or Action	Purpose
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.

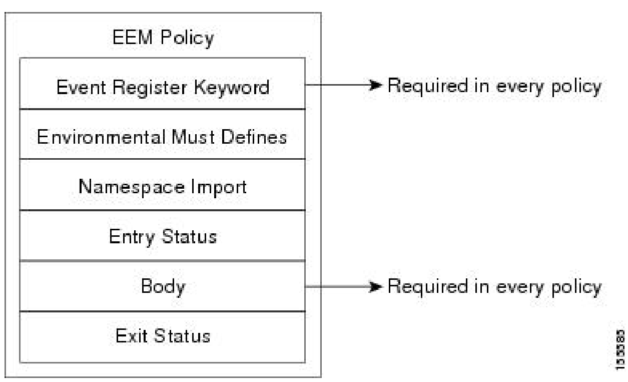
EEM Policies Using TCL: Details

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:

```
XYSSSSSSSSSSSSSEEEEEEEEPPPPPPPPP
```

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

Table 20: _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 Table 21: Error Class Encodings, on page 77 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 21: Error Class Encodings

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

An error return value of zero means SUCCESS.

CHAPTER 7

Implementing IP Service Level Agreements

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

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

Table 22: Feature History Table for IP SLA

Release	Modification
Release 6.3.1	Two-Way Active Measurement Protocol (TWAMP) was introduced.

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

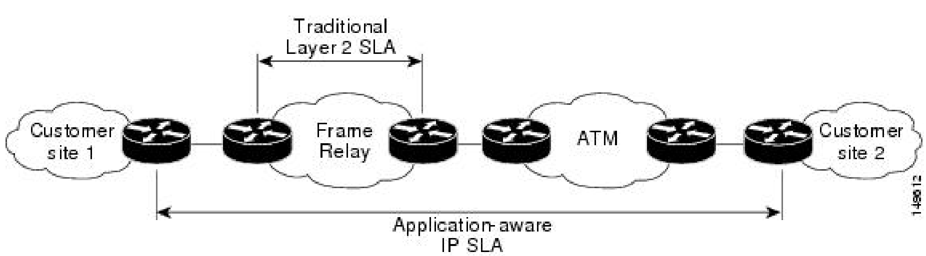
Internet commerce has grown significantly in the past few years as the technology has advanced to provide faster, more reliable access to the Internet. Many companies need online access and conduct most of their business on line and any loss of service can affect the profitability of the company. Internet 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. Figure 4: Scope of Traditional Service Level Agreement Versus IP SLA, on page 80 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 23: 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 24: 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 Implementing IP Service Level Agreements

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

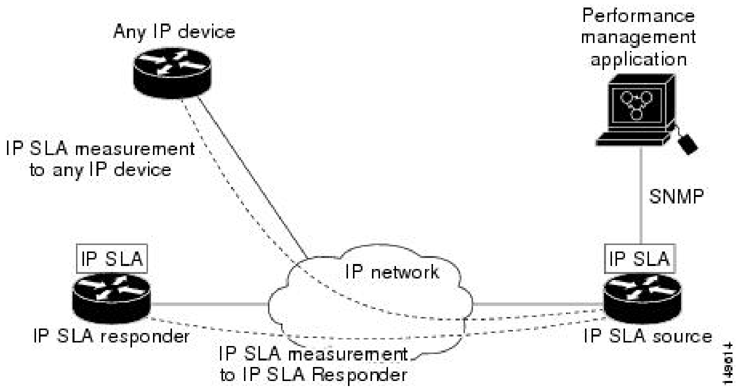
  > **Note** Setting the frequency to less than 60 seconds will increase the number of packets sent. But this could negatively impact the performance of IP SLA operation when scheduled operations have same start time.

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

Measuring Network Performance with IP Service Level Agreements

IP SLA uses generated traffic to measure network performance between two networking devices, such as routers. Figure 5: IP SLA Operations, on page 83 shows how IP SLA starts when the IP SLA device sends a generated packet to the destination device. After the destination device receives the packet and if the operation uses an IP SLA component at the receiving end (for example, IP SLA Responder), the reply packet includes information about the delay at the target device. The source device uses this information to improve the accuracy of the measurements. An IP SLA operation is a network measurement to a destination in the network from the source device using a specific protocol, such as User Datagram Protocol (UDP) for the operation.

Figure 5: IP SLA Operations



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

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

The following topics are covered in this section:

IP SLA Responder and IP SLA Control Protocol

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

Figure 5: IP SLA Operations, on page 83 shows where the IP SLA Responder fits relative to the IP network. The IP SLA Responder listens on a specific port for control protocol messages sent by an IP SLA operation. Upon receipt of the control message, the responder enables the UDP port specified in the control message for the specified duration. During this time, the responder accepts the requests and responds to them. The responder disables the port after it responds to the IP SLA packet or packets, or when the specified time expires. For added security, MD5 authentication for control messages is available.

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

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

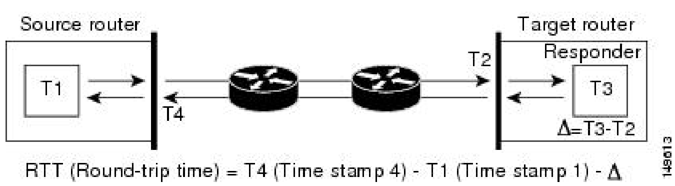
Response Time Computation for IP SLA

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

When enabled, the IP SLA Responder allows the target device to take two time stamps, both when the packet arrives on the interface and again just as it is leaving, and accounts for it when calculating the statistics. This time stamping is made with a granularity of submilliseconds.

Figure 6: IP SLA Responder Time Stamping, on page 84 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



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 25: 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. Table 25: Types of Operations for IP SLA, on page 85 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.

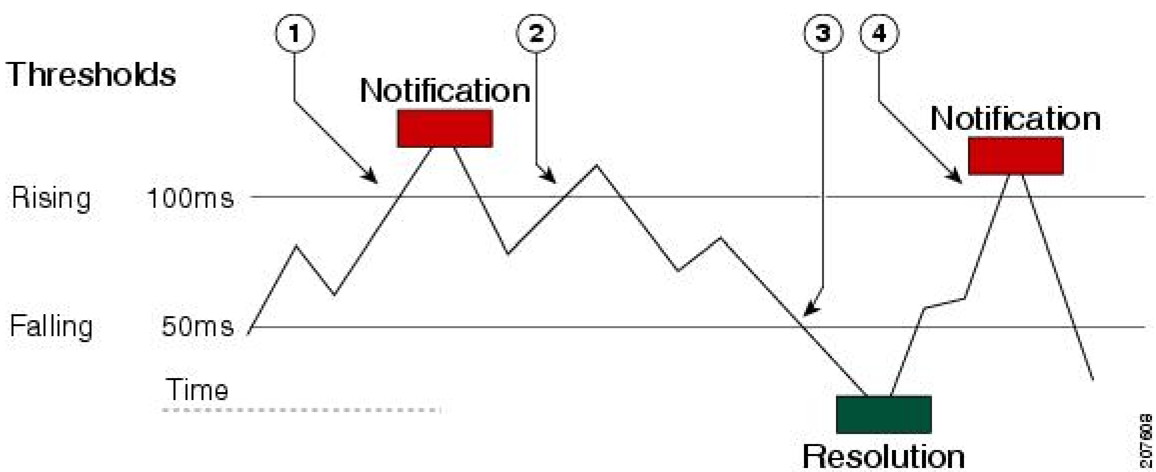
IP SLA Threshold Monitoring and Notifications

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.

Two-Way Active Measurement Protocol (TWAMP)

The Two-Way Active Measurement Protocol (TWAMP) defines a flexible method for measuring round-trip IP performance between any two devices and thereby checks IP SLA compliance.

Advantages of TWAMP

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

Note TWAMP v4 and v6 are supported.

The following topics are the covered in this section:

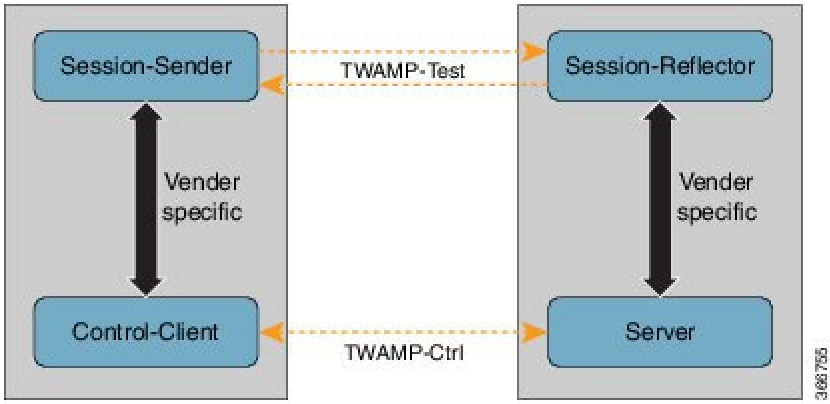
The TWAMP Entities

The TWAMP system consists of 4 logical entities:

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

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

Figure 8: The TWAMP Entities



TWAMP Protocols

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

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

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

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

  - request-session
  - start-session
  - stop-session

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

TWAMP-Control and TWAMP-test stream, have three security modes: unauthenticated, authenticated, and encrypted.

Restrictions of TWAMP on the Router

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

Configuring TWAMP on the Router

Configuration of Session-Server

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

Configuration of Session-Reflector

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

Running Configuration

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

Verification of TWAMP

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

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

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

```
Router# show ipsla twamp session
IP SLAs Responder TWAMP is: Enabled
Recvr Addr: 10.5.139.11
Recvr Port: 7222
Sender Addr: 172.27.111.233
Sender Port: 33243
Session Id: 10.5.139.11:70929508:88F7A620
Connection Id: 0
```

The TWAMP test session based on source ip-address can be verified using the command: **show ipsla twamp session source-ip <source ip-address> source-port <source port-number>**

```
Router# show ipsla twamp session source-ip 172.27.111.233 source-port 33286
IP SLAs Responder TWAMP is: Enabled
Recvr Addr: 10.5.139.11
Recvr Port: 6198
Sender Addr: 172.27.111.233
Sender Port: 33286
Session Id: 10.5.139.11:71804476:F2721505
Connection Id: D
Mode: UnAuthorized
DSCP: 0
Pad Length: 0
Number of Packets Received: 8867
```

TWAMP-Light

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

Note TWAMP-Light v4 and v6 are supported.

Restrictions of TWAMP-Light

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

Configuring TWAMP-Light

This example shows you how to configure TWAMP-Light and the timeout value:

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

Running Configuration

This is a sample running configuration of twamp-light responder:

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

Verification of TWAMP-Light

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

```
Router# show ipsla twamp session
***** TWAMP Sessions *****
No records matching query found
***** TWAMP-LIGHT Sessions *****
Session status: Active
Recvr Addr: any (IPV4)
Recvr Port: 2345
Sender Addr: any (IPV4)
Sender Port: any
Sender VRF Name: any
Session ID: 10
Mode: Unauthenticated
Number of Packets Received: 0
Session timeout: 0
Number of Packets Sent: 0
```

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 92.

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

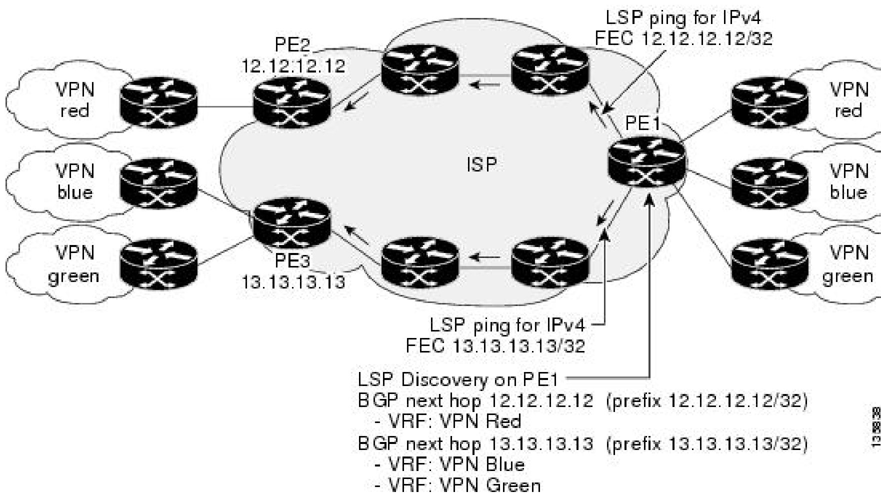
BGP next-hop neighbor discovery is used to find the BGP next-hop neighbors in use by any VRF associated with the source provider edge (PE) router. In most cases, these neighbors are PE routers.

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 9: BGP Next-hop Neighbor Discovery for a Simple VPN, on page 93 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 9: 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 147 and the Configuring an MPLS LSP Monitoring Trace Instance, on page 151.

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.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **ipsla responder**

Example:

```
RP/0/RP0/CPU0:router(config)# ipsla responder
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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 96 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 26: UDP Jitter Operation Parameters, on page 97.

Table 26: 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 95 section.

Configuring and Scheduling a Basic UDP Jitter Operation on the Source Device

You can configure and schedule a UDP jitter operation.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **ipsla operation** *operation-number*

Example:

```
RP/0/RP0/CPU0:router(config)# ipsla operation 432
```

Specifies the operation number. The range is from 1 to 2048.

Step 3 **type udp jitter**

Example:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0:router(config-ipsla-udp-jitter)# exit
RP/0/RP0/CPU0:router(config-ipsla-op)# exit
RP/0/RP0/CPU0:router(config-ipsla)# exit
RP/0/RP0/CPU0:router(config)#
```

Exits from IP SLA configuration mode and operational mode, and returns the CLI to global configuration mode.

Step 10 **ipsla schedule operation** *op-num*

Example:

```
RP/0/RP0/CPU0:router(config)# ipsla schedule operation 432
RP/0/RP0/CPU0:router(config-ipsla-sched)#
```

Schedules the start time of the operation. You can configure a basic schedule.

Step 11 **life** { **forever** | *seconds*}

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-sched)# life 30
```

The **forever** keyword schedules the operation to run indefinitely. The *seconds* argument schedules the lifetime of the operation, in seconds. The default lifetime of an operation is 3600 seconds (one hour).

Step 12 **ageout** *seconds*

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-sched)# ageout 3600
```

(Optional) Specifies the number of seconds to keep the operation in memory when it is not actively collecting information. The default value of 0 seconds means that the operation never times out.

Step 13 **recurring**

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-sched)# recurring
```

(Optional) Specifies that the operation starts automatically at the specified time and for the specified duration every day.

Step 14 **start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*]

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-sched)# start-time 01:00:00
```

Specifies a time for the operation to start. The following keywords are described:

- (Optional) Use the **pending** keyword to configure the operation to remain in a pending (unstarted) state. The default is inactive. If the **start-time** command is not specified, no information is collected until the start time is configured or a trigger occurs that performs an immediate start.
- (Optional) Use the **now** keyword to indicate that the operation should start immediately.
- (Optional) Use the **after** keyword and associated arguments to specify the time after which the operation starts collecting information.

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.

Configuring and Scheduling a UDP Jitter Operation with Additional Characteristics

You can configure and schedule a UDP jitter operation.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **ipsla operation** *operation-number*

Example:

```
RP/0/RP0/CPU0:router(config)# ipsla operation 432
```

Specifies the operation number. The range is from 1 to 2048.

Step 3 **type udp jitter**

Example:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0:router(config-ipsla-udp-jitter)# statistics hourly
RP/0/RP0/CPU0:router(config-ipsla-op-stats)#
```

(Optional) Specifies the statistics collection parameters for UDP jitter operation.

Step 9 **buckets** *hours*

Example:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0:router(config-ipsla-op-stats)# exit
```

Exits from IP SLA statistics configuration mode.

Step 13 **datasize request** *size*

Example:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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.

Step 15 **tos** *number*

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-udp-jitter)# tos 255
```

Specifies the type of service number.

Step 16 **exit**

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-udp-jitter)# exit
RP/0/RP0/CPU0:router(config-ipsla-op)# exit
RP/0/RP0/CPU0:router(config-ipsla)# exit
RP/0/RP0/CPU0:router(config)#
```

Exits from IP SLA configuration mode and operational mode, and returns the CLI to global configuration mode.

Step 17 **ipsla schedule operation** *op-num*

Example:

```
RP/0/RP0/CPU0:router(config)# ipsla schedule operation 432
RP/0/RP0/CPU0:router(config-ipsla-sched)#
```

Schedules the start time of the operation. You can configure a basic schedule.

Step 18 **life** {**forever** | *seconds*}

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-sched)# life 30
```

The **forever** keyword schedules the operation to run indefinitely. The *seconds* argument schedules the lifetime of the operation, in seconds. The default lifetime of an operation is 3600 seconds (one hour).

Step 19 **ageout** *seconds*

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-sched)# ageout 3600
```

(Optional) Specifies the number of seconds to keep the operation in memory when it is not actively collecting information. The default value of 0 seconds means that the operation never times out.

Step 20 **recurring**

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-sched)# recurring
```

(Optional) Specifies that the operation starts automatically at the specified time and for the specified duration every day.

Step 21 **start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*]

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-sched)# start-time 01:00:00
```

(Optional) Specifies a time for the operation to start. The following keywords are described:

- (Optional) Use the **pending** keyword to configure the operation to remain in a pending (unstarted) state. The default is inactive. If the **start-time** command is not specified, no information is collected until the start time is configured or a trigger occurs that performs an immediate start.
- (Optional) Use the **now** keyword to indicate that the operation should start immediately.
- (Optional) Use the **after** keyword and associated arguments to specify the time after which the operation starts collecting information.

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.

Step 23 **show ipsla statistics** [*operation-number*]

Example:

```
RP/0/RP0/CPU0:router # show ipsla statistics 432
```

Displays the current statistics.

Step 24 **show ipsla statistics aggregated** [*operation-number*]

Example:

```
RP/0/RP0/CPU0: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 95 section.

Configuring and Scheduling a UDP Echo Operation on the Source Device

You can enable a UDP echo operation without any optional parameters.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **ipsla operation** *operation-number*

Example:

```
RP/0/RP0/CPU0:router(config)# ipsla operation 432
```

Specifies the operation number. The range is from 1 to 2048.

Step 3 **type udp echo**

Example:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0:router(config-ipsla-udp-echo)# exit
RP/0/RP0/CPU0:router(config-ipsla-op)# exit
RP/0/RP0/CPU0:router(config-ipsla)# exit
RP/0/RP0/CPU0:router(config)#
```

Exits IP SLA operation configuration mode and IP SLA configuration mode. Returns to global configuration mode.

Step 8 **ipsla schedule operation** *op-num*

Example:

```
RP/0/RP0/CPU0:router(config)# ipsla schedule operation 432
RP/0/RP0/CPU0:router(config-ipsla-sched)#
```

Schedules the start time of the operation. You can configure a basic schedule.

Step 9 **life** [**forever** | *seconds*]

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-sched)# life 1
```

The **forever** keyword schedules the operation to run indefinitely. The *seconds* argument schedules the lifetime of the operation, in seconds. The default lifetime of an operation is 3600 seconds (one hour).

Step 10 **ageout** *seconds*

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-sched)# ageout 3600
```

(Optional) Specifies the number of seconds to keep the operation in memory when it is not actively collecting information. The default value of 0 seconds means that the operation never times out.

Step 11 **recurring**

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-sched)# recurring
```

(Optional) Specifies that the operation starts automatically at the specified time and for the specified duration every day.

Step 12 **start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*]

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-sched)# start-time 01:00:00
```

(Optional) Specifies a time for the operation to start. The following keywords are described:

- (Optional) Use the **pending** keyword to configure the operation to remain in a pending (unstarted) state. This is the default value. If the **start-time** command is not specified, no information is collected until the start time is configured or a trigger occurs that performs an immediate start.
- (Optional) Use the **now** keyword to indicate that the operation should start immediately.
- (Optional) Use the **after** keyword and associated arguments to specify the time after which the operation starts collecting information.

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:

```
RP/0/RP0/CPU0:router# show ipsla statistics 432
```

Displays the current statistics.

Step 15 **show ipsla statistics aggregated** [*operation-number*]

Example:

```
RP/0/RP0/CPU0:router# show ipsla statistics aggregated 1
```

Displays the hourly statistical errors and the hourly statistics for all the IP SLA operations or specified operation.

Configuring and Scheduling 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.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **ipsla operation** *operation-number*

Example:

```
RP/0/RP0/CPU0:router(config)# ipsla operation 432
```

Specifies the operation number. The range is from 1 to 2048.

Step 3 **type udp echo**

Example:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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.

Step 11 **tag** *text*

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-udp-echo)# type udp echo tag ipsla
```

(Optional) Creates a user-specified identifier for an IP SLA operation.

Step 12 **exit**

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-udp-echo)# exit
RP/0/RP0/CPU0:router(config-ipsla-op)# exit
RP/0/RP0/CPU0:router(config-ipsla)# exit
RP/0/RP0/CPU0:router(config)#
```

Exits IP SLA operation configuration mode and IPSLA configuration mode. Returns to global configuration mode.

Step 13 **ipsla schedule operation** *op-num*

Example:

```
RP/0/RP0/CPU0:router(config)# ipsla schedule operation 432
RP/0/RP0/CPU0:router(config-ipsla-sched)#
```

Schedules the start time of the operation. You can configure a basic schedule or schedule multiple operations using group scheduling.

Step 14 **life** {**forever** | *seconds*}

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-sched)# life 30
```

The **forever** keyword schedules the operation to run indefinitely. The *seconds* argument schedules the lifetime of the operation, in seconds. The default lifetime of an operation is 3600 seconds (one hour).

Step 15 **ageout** *seconds*

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-sched)# ageout 3600
```

(Optional) Specifies the number of seconds to keep the operation in memory when it is not actively collecting information. The default value of 0 seconds means that the operation never times out.

Step 16 **recurring**

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-sched)# recurring
```

(Optional) Specifies that the operation starts automatically at the specified time and for the specified duration every day.

Step 17 **start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*]

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-sched)# start-time 01:00:00
```

Specifies a time for the operation to start. The following keywords are described:

- (Optional) Use the **pending** keyword to configure the operation to remain in a pending (unstarted) state. The default value is inactive. If the **start-time** command is not specified, no information is collected until the start time is configured or a trigger occurs that performs an immediate start.
- (Optional) Use the **now** keyword to indicate that the operation should start immediately.
- (Optional) Use the **after** keyword and associated arguments to specify the time after which the operation starts collecting information.

Step 18 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 19 **show ipsla statistics enhanced aggregated** [*operation-number*] **interval** *seconds*

Example:

```
RP/0/RP0/CPU0: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 20 **show ipsla statistics** [*operation-number*]

Example:

```
RP/0/RP0/CPU0: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.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **ipsla operation** *operation-number*

Example:

```
RP/0/RP0/CPU0:router(config)# ipsla operation 432
```

Specifies the operation number. The range is from 1 to 2048.

Step 3 **type icmp echo**

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-op)# type icmp echo
```

Defines an ICMP echo operation type.

Step 4 **destination address** *ipv4address*

Example:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0:router(config-ipsla-icmp-echo)# exit
RP/0/RP0/CPU0:router(config-ipsla-op)# exit
RP/0/RP0/CPU0:router(config-ipsla)# exit
RP/0/RP0/CPU0:router(config)#
```

Exits IP SLA operation configuration mode and IP SLA configuration mode. Returns to global configuration mode.

Step 7 **ipsla schedule operation** *op-num*

Example:

```
RP/0/RP0/CPU0:router(config)# ipsla schedule operation 432
RP/0/RP0/CPU0:router(config-ipsla-sched)#
```

Schedules the start time of the operation. You can configure a basic schedule.

Step 8 **life** {**forever** | *seconds*}

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-sched)# life 30
```

The **forever** keyword schedules the operation to run indefinitely. The *seconds* argument schedules the lifetime of the operation, in seconds. The default lifetime of an operation is 3600 seconds (one hour).

Step 9 **ageout** *seconds*

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-sched)# ageout 3600
```

(Optional) Specifies the number of seconds to keep the operation in memory when it is not actively collecting information. The default value of 0 seconds means that the operation never times out.

Step 10 **recurring**

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-sched)# recurring
```

(Optional) Specifies that the operation starts automatically at the specified time and for the specified duration every day.

Step 11 **start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*]

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-sched)# start-time 01:00:00
```

Specifies a time for the operation to start. The following keywords are described:

- (Optional) Use the **pending** keyword to configure the operation to remain in a pending (unstarted) state. The default value is inactive. If the **start-time** command is not specified, no information is collected until the start time is configured or a trigger occurs that performs an immediate start.
- (Optional) Use the **now** keyword to indicate that the operation should start immediately.
- (Optional) Use the **after** keyword and associated arguments to specify the time after which the operation starts collecting information.

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:

```
RP/0/RP0/CPU0:router # show ipsla statistics 432
```

Displays the current statistics.

Configuring 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.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **ipsla operation** *operation-number*

Example:

```
RP/0/RP0/CPU0:router(config)# ipsla operation 432
```

Specifies the operation number. The range is from 1 to 2048.

Step 3 **type icmp echo**

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-op)# type icmp echo
```

Defines an ICMP echo operation type.

Step 4 **vrf** *vrf-name*

Example:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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.

Step 10 **tag** *text*

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-icmp-echo)# tag ipsla
```

(Optional) Creates a user-specified identifier for an IP SLA operation.

Step 11 exit

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-icmp-echo)# exit
RP/0/RP0/CPU0:router(config-ipsla-op)# exit
RP/0/RP0/CPU0:router(config-ipsla)# exit
RP/0/RP0/CPU0:router(config)#
```

Exits IP SLA operation configuration mode and IP SLA configuration mode. Returns to global configuration mode.

Step 12 **ipsla schedule operation** *op-num*

Example:

```
RP/0/RP0/CPU0:router(config)# ipsla schedule operation 432
RP/0/RP0/CPU0:router(config-ipsla-sched)#
```

Schedules the start time of the operation. You can configure a basic schedule.

Step 13 **life** {**forever** | *seconds*}

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-sched)# life 30
```

The **forever** keyword schedules the operation to run indefinitely. The *seconds* argument schedules the lifetime of the operation, in seconds. The default lifetime of an operation is 3600 seconds (one hour).

Step 14 **ageout** *seconds*

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-sched)# ageout 3600
```

(Optional) Specifies the number of seconds to keep the operation in memory when it is not actively collecting information. The default value of 0 seconds means that the operation never times out.

Step 15 **recurring**

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-sched)# recurring
```

(Optional) Specifies that the operation starts automatically at the specified time and for the specified duration every day.

Step 16 **start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*]

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-sched)# start-time 01:00:00
```

Specifies a time for the operation to start. The following keywords are described:

- (Optional) Use the **pending** keyword to configure the operation to remain in a pending (unstarted) state. The default value is inactive. If the **start-time** command is not specified, no information is collected until the start time is configured or a trigger occurs that performs an immediate start.
- (Optional) Use the **now** keyword to indicate that the operation should start immediately.
- (Optional) Use the **after** keyword and associated arguments to specify the time after which the operation starts collecting information.

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:

```
RP/0/RP0/CPU0: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.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **ipsla operation** *operation-number*

Example:

```
RP/0/RP0/CPU0:router(config)# ipsla operation 432
```

Specifies the operation number. The range is from 1 to 2048.

Step 3 **type icmp path-echo**

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-op)# type icmp path-echo
RP/0/RP0/CPU0:router(config-ipsla-icmp-path-echo)#
```

Defines an ICMP path-echo operation type.

Step 4 **destination address** *ipv4address*

Example:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0:router(config-ipsla-icmp-path-echo)# exit
RP/0/RP0/CPU0:router(config-ipsla-op)# exit
RP/0/RP0/CPU0:router(config-ipsla)# exit
RP/0/RP0/CPU0:router(config)#
```

Exits IP SLA operation configuration mode and IP SLA configuration mode. Returns to global configuration mode.

Step 7 **ipsla schedule operation** *op-num*

Example:

```
RP/0/RP0/CPU0:router(config)# ipsla schedule operation 432
RP/0/RP0/CPU0:router(config-ipsla-sched)#
```

Schedules the start time of the operation. You can configure a basic schedule.

Step 8 **life** {**forever** | *seconds*}

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-sched)# life 30
```

The **forever** keyword schedules the operation to run indefinitely. The *seconds* argument schedules the lifetime of the operation, in seconds. The default lifetime of an operation is 3600 seconds (one hour).

Step 9 **ageout** *seconds*

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-sched)# ageout 3600
```

(Optional) Specifies the number of seconds to keep the operation in memory when it is not actively collecting information. The default value of 0 seconds means that the operation never times out.

Step 10 **recurring**

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-sched)# recurring
```

(Optional) Specifies that the operation starts automatically at the specified time and for the specified duration every day.

Step 11 **start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*]

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-sched)# start-time 01:00:00
```

Specifies a time for the operation to start. The following keywords are described:

- (Optional) Use the **pending** keyword to configure the operation to remain in a pending (unstarted) state. The default value is inactive. If the **start-time** command is not specified, no information is collected until the start time is configured or a trigger occurs that performs an immediate start.
- (Optional) Use the **now** keyword to indicate that the operation should start immediately.
- (Optional) Use the **after** keyword and associated arguments to specify the time after which the operation starts collecting information.

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:

```
RP/0/RP0/CPU0:router# show ipsla statistics 432
```

Displays the current statistics.

Configuring and Scheduling 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.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **ipsla operation** *operation-number*

Example:

```
RP/0/RP0/CPU0:router(config)# ipsla operation 432
```

Specifies the operation number. The range is from 1 to 2048.

Step 3 **type icmp path-echo**

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-op)# type icmp path-echo
RP/0/RP0/CPU0:router(config-ipsla-icmp-path-echo)#
```

Defines an ICMP path-echo operation type.

Step 4 **vrf** *vrf-name*

Example:

```
RP/0/RP0/CPU0: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.

Step 5 **lsr-path** *ip-address*

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-imcp-path-echo)# lsr-path 20.25.22.1
```

Specifies that a loose source routing path is to be used.

Step 6 **destination address** *ipv4address*

Example:

```
RP/0/RP0/CPU0: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 7 **frequency** *seconds*

Example:

```
RP/0/RP0/CPU0: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 8 **datasize request** *size*

Example:

```
RP/0/RP0/CPU0: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 9 **tos** *number*

Example:

```
RP/0/RP0/CPU0: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 10 **timeout** *milliseconds*

Example:

```
RP/0/RP0/CPU0: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.

Step 11 **tag** *text*

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-icmp-path-echo)# tag ipsla
```

(Optional) Creates a user-specified identifier for an IP SLA operation.

Step 12 **lsr-path** *ipaddress1* {*ipaddress2* {... {*ipaddress8*}}}

Example:

```
RP/0/RP0/CPU0: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 13 **exit**

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-icmp-path-echo)# exit
```

```
RP/0/RP0/CPU0:router(config-ipsla-op)# exit
RP/0/RP0/CPU0:router(config-ipsla)# exit
RP/0/RP0/CPU0:router(config)#
```

Exits IP SLA operation configuration mode and IP SLA configuration mode. Returns to global configuration mode.

Step 14 **ipsla schedule operation** *op-num*

Example:

```
RP/0/RP0/CPU0:router(config)# ipsla schedule operation 432
RP/0/RP0/CPU0:router(config-ipsla-sched)#
```

Schedules the start time of the operation. You can configure a basic schedule.

Step 15 **life** {**forever** | *seconds*}

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-sched)# life 1
```

The **forever** keyword schedules the operation to run indefinitely. The *seconds* argument schedules the lifetime of the operation, in seconds. The default lifetime of an operation is 3600 seconds (one hour).

Step 16 **ageout** *seconds*

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-sched)# ageout 3600
```

(Optional) Specifies the number of seconds to keep the operation in memory when it is not actively collecting information. The default value of 0 seconds means that the operation never times out.

Step 17 **recurring**

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-sched)# recurring
```

(Optional) Specifies that the operation starts automatically at the specified time and for the specified duration every day.

Step 18 **start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*]

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-sched)# start-time 01:00:00
```

Specifies a time for the operation to start. The following keywords are described:

- (Optional) Use the **pending** keyword to configure the operation to remain in a pending (unstarted) state. The default value is inactive. If the **start-time** command is not specified, no information is collected until the start time is configured or a trigger occurs that performs an immediate start.
- (Optional) Use the **now** keyword to indicate that the operation should start immediately.
- (Optional) Use the **after** keyword and associated arguments to specify the time after which the operation starts collecting information.

Step 19 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 20 **show ipsla statistics** [*operation-number*]

Example:

```
RP/0/RP0/CPU0: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 27: 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

ICMP Path-jitter Operation Parameter	Default	Configured Using
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.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **ipsla operation** *operation-number*

Example:

```
RP/0/RP0/CPU0:router(config)# ipsla operation 432
```

Specifies the operation number. The range is from 1 to 2048.

Step 3 **type icmp path-jitter**

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-op)# type icmp path-jitter
```

Defines an ICMP path-jitter operation type.

Step 4 **destination address** *ipv4address*

Example:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0:router(config-ipsla-icmp-path-jitter)# exit
RP/0/RP0/CPU0:router(config-ipsla-op)# exit
RP/0/RP0/CPU0:router(config-ipsla)# exit
RP/0/RP0/CPU0:router(config)#
```

Exits IP SLA operation configuration mode and IP SLA configuration mode. Returns to global configuration mode.

Step 9 **ipsla schedule operation** *op-num*

Example:

```
RP/0/RP0/CPU0:router(config)# ipsla schedule operation 432
RP/0/RP0/CPU0:router(config-ipsla-sched)#
```

Schedules the start time of the operation. You can configure a basic schedule.

Step 10 **life** {**forever** | *seconds*}

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-sched)# life 30
```

The **forever** keyword schedules the operation to run indefinitely. The *seconds* argument schedules the lifetime of the operation, in seconds. The default lifetime of an operation is 3600 seconds (one hour).

Step 11 **ageout** *seconds*

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-sched)# ageout 3600
```

(Optional) Specifies the number of seconds to keep the operation in memory when it is not actively collecting information. The default value of 0 seconds means that the operation never times out.

Step 12 **recurring**

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-sched)# recurring
```

(Optional) Specifies that the operation starts automatically at the specified time and for the specified duration every day.

Step 13 **start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*]

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-sched)# start-time 01:00:00
```

(Optional) Specifies a time for the operation to start. The following keywords are described:

- (Optional) Use the **pending** keyword to configure the operation to remain in a pending (unstarted) state. The default value is inactive. If the **start-time** command is not specified, no information is collected until the start time is configured or a trigger occurs that performs an immediate start.
- (Optional) Use the **now** keyword to indicate that the operation should start immediately.
- (Optional) Use the **after** keyword and associated arguments to specify the time after which the operation starts collecting information.

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 **show ipsla statistics** [*operation-number*]

Example:

```
RP/0/RP0/CPU0:router# show ipsla statistics 432
```

Displays the current statistics.

Configuring and Scheduling 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.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **ipsla operation** *operation-number*

Example:

```
RP/0/RP0/CPU0:router(config)# ipsla operation 432
```

Specifies the operation number. The range is from 1 to 2048.

Step 3 **type icmp path-jitter**

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-op)# type icmp path-jitter
```

Defines an ICMP path-jitter operation type.

Step 4 **vrf** *vrf-name*

Example:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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.

Step 13 **tag** *text*

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-icmp-path-jitter)# tag ipsla
```

(Optional) Creates a user-specified identifier for an IP SLA operation.

Step 14 **exit**

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-icmp-path-jitter)# exit
RP/0/RP0/CPU0:router(config-ipsla-op)# exit
RP/0/RP0/CPU0:router(config-ipsla)# exit
RP/0/RP0/CPU0:router(config)#
```

Exits IP SLA operation configuration mode and IP SLA configuration mode. Returns to global configuration mode.

Step 15 **ipsla schedule operation** *op-num*

Example:

```
RP/0/RP0/CPU0:router(config)# ipsla schedule operation 432
RP/0/RP0/CPU0:router(config-ipsla-sched)#
```

Schedules the start time of the operation. You can configure a basic schedule.

Step 16 **life** {**forever** | *seconds*}

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-sched)# life 30
```

The **forever** keyword schedules the operation to run indefinitely. The *seconds* argument schedules the lifetime of the operation, in seconds. The default lifetime of an operation is 3600 seconds (one hour).

Step 17 **ageout** *seconds*

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-sched)# ageout 3600
```

(Optional) Specifies the number of seconds to keep the operation in memory when it is not actively collecting information. The default value of 0 seconds means that the operation never times out.

Step 18 **recurring**

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-sched)# recurring
```

(Optional) Specifies that the operation starts automatically at the specified time and for the specified duration every day.

Step 19 **start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*]

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-sched)# start-time 01:00:00
```

Specifies a time for the operation to start. The following keywords are described:

- (Optional) Use the **pending** keyword to configure the operation to remain in a pending (unstarted) state. The default value is inactive. If the **start-time** command is not specified, no information is collected until the start time is configured or a trigger occurs that performs an immediate start.
- (Optional) Use the **now** keyword to indicate that the operation should start immediately.

- (Optional) Use the **after** keyword and associated arguments to specify the time after which the operation starts collecting information.

Step 20 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 21 **show ipsla statistics** [*operation-number*]

Example:

```
RP/0/RP0/CPU0: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)

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **ipsla operation** *operation-number*

Example:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0:router(config-ipsla-mpls-lsp-ping)# target ipv4 10.25.26.10 255.255.255.255
```

or

```
RP/0/RP0/CPU0:router(config-ipsla-mpls-lsp-ping)# target ipv4 10.25.26.10/32
```

or

```
RP/0/RP0/CPU0:router(config-ipsla-mpls-lsp-ping)# target traffic-eng tunnel 12
```

or

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0:router(config-ipsla-mpls-lsp-ping)# reply mode router-alert
```

or

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0:router(config-ipsla-mpls-lsp-ping)# exit
RP/0/RP0/CPU0:router(config-ipsla-op)# exit
RP/0/RP0/CPU0:router(config-ipsla)# exit
RP/0/RP0/CPU0:router(config)#
```

Exits IP SLA MPLS LSP Ping configuration mode and IP SLA configuration mode. Returns to global configuration mode.

Step 13 **ipsla schedule operation** *operation-number*

Example:

```
RP/0/RP0/CPU0:router(config)# ipsla schedule operation 432
RP/0/RP0/CPU0:router(config-ipsla-sched)#
```

Schedules the start time of the operation. You can configure a basic schedule.

Step 14 **start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*]

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-sched)# start-time 01:00:00
```

Specifies a time for the operation to start. The following keywords are described:

- (Optional) Use the **pending** keyword to configure the operation to remain in a pending (unstarted) state. The default value is inactive. If the **start-time** command is not specified, no information is collected until the start time is configured or a trigger occurs that performs an immediate start.
- (Optional) Use the **now** keyword to indicate that the operation should start immediately.
- (Optional) Use the **after** keyword and associated arguments to specify the time after which the operation starts collecting information.

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:

```
RP/0/RP0/CPU0: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)

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **ipsla operation** *operation-number*

Example:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0:router(config-ipsla-mpls-lsp-trace)# target ipv4 10.25.26.10 255.255.255.255
```

```
RP/0/RP0/CPU0:router(config-ipsla-mpls-lsp-trace)# target ipv4 10.25.26.10/32
```

or

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0:router(config-ipsla-mpls-lsp-trace)# exit
RP/0/RP0/CPU0:router(config-ipsla-op)# exit
RP/0/RP0/CPU0:router(config-ipsla)# exit
RP/0/RP0/CPU0:router(config)#
```

Exits IP SLA MPLS LSP Trace configuration mode and IP SLA configuration mode. Returns to global configuration mode.

Step 13 **ipsla schedule operation** *operation-number*

Example:

```
RP/0//CPU0:router(config)# ipsla schedule operation 432
RP/0//CPU0:router(config-ipsla-sched)#
```

Schedules the start time of the operation. You can configure a basic schedule.

Step 14 **start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*]

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-sched)# start-time 01:00:00
```

Specifies a time for the operation to start. The following keywords are described:

- (Optional) Use the **pending** keyword to configure the operation to remain in a pending (unstarted) state. The default value is inactive. If the **start-time** command is not specified, no information is collected until the start time is configured or a trigger occurs that performs an immediate start.
- (Optional) Use the **now** keyword to indicate that the operation should start immediately.
- (Optional) Use the **after** keyword and associated arguments to specify the time after which the operation starts collecting information.

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:

```
RP/0/RP0/CPU0: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.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **ipsla reaction operation** *operation-number*

Example:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0:router(config-ipsla-react)# react connection-loss
RP/0/RP0/CPU0: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.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **ipsla reaction operation** *operation-number*

Example:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0:router(config-ipsla-react)# react jitter-average
RP/0/RP0/CPU0: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.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **ipsla reaction operation** *operation-number*

Example:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0:router(config-ipsla-react)# react packet-loss dest-to-source
RP/0/RP0/CPU0: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.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **ipsla reaction operation** *operation-number*

Example:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0:router(config-ipsla-react)# react rtt
RP/0/RP0/CPU0: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.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **ipsla reaction operation** *operation-number*

Example:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0:router(config-ipsla-react)# react timeout
RP/0/RP0/CPU0: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.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **ipsla reaction operation** *operation-number*

Example:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0:router(config-ipsla-react)# react verify-error
RP/0/RP0/CPU0: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 28: 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 143.
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 142.
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 144.
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 145.

Generating Events for Each Violation

You can generate a trap or trigger another operation each time a specified condition is met.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **ipsla reaction operation** *operation-number*

Example:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0:router(config-ipsla-react)# react timeout
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **ipsla reaction operation** *operation-number*

Example:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0:router(config-ipsla-react)# react connection-loss
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **ipsla reaction operation** *operation-number*

Example:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0:router(config-ipsla-react)# react rtt
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **ipsla reaction operation** *operation-number*

Example:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0:router(config-ipsla-react)# react packet-loss dest-to-source
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **ipsla reaction operation** *operation-number*

Example:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0:router(config-ipsla-react)# react connection-loss
RP/0/RP0/CPU0:router(config-ipsla-react-cond)#
```

Specifies a reaction if there is a connection-loss for the monitored operation.

Step 4 **action** [**logging** | **trigger**]

Example:

```
RP/0/RP0/CPU0: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.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **ipsla**

Example:

```
RP/0/RP0/CPU0:router(config)# ipsla
```

Enters IP SLA configuration mode and configures IP service level agreements.

Step 3 **mpls discovery vpn**

Example:

```
RP/0/RP0/CPU0:router(config-ipsla)# mpls discovery vpn
```

(Optional) Enters MPLS VPN BGP next-hop neighbor discovery configuration mode.

Step 4 **interval** *minutes*

Example:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0:router(config-ipsla-mpls-discovery-vpn)# exit
```

Exits MPLS discovery VPN configuration mode.

Step 6 **mpls lsp-monitor**

Example:

```
RP/0/RP0/CPU0:router(config-ipsla)# mpls lsp-monitor
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0:router(config-ipsla-mplslm)# monitor 1
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **ipsla**

Example:

```
RP/0/RP0/CPU0:router(config)# ipsla
```

Enters IP SLA configuration mode and configures IP service level agreements.

Step 3 **mpls discovery vpn**

Example:

```
RP/0/RP0/CPU0:router(config-ipsla)# mpls discovery vpn
```

(Optional) Enables MPLS VPN BGP next-hop neighbor discovery.

Step 4 **interval** *minutes*

Example:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0:router(config-ipsla-mpls-discovery-vpn)# exit
```

Exits MPLS discovery VPN configuration mode.

Step 6 **mpls lsp-monitor**

Example:

```
RP/0/RP0/CPU0:router(config-ipsla)# mpls lsp-monitor
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0:router(config-ipsla-mplslm)# monitor 1
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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. 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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0//CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **ipsla**

Example:

```
RP/0/RP0/CPU0:router(config)# ipsla
```

Enters IP SLA configuration mode and configures IP service level agreements.

Step 3 **mpls lsp-monitor**

Example:

```
RP/0/RP0/CPU0:router(config-ipsla)# mpls lsp-monitor
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0:router(config-ipsla-mplslm)# reaction monitor 2
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **ipsla**

Example:

```
RP/0/RP0/CPU0:router(config)# ipsla
```

Enters IP SLA configuration mode and configures IP service level agreements.

Step 3 **mpls lsp-monitor**

Example:

```
RP/0/RP0/CPU0:router(config-ipsla)# mpls lsp-monitor
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0:router(config-ipsla-mplslm)# schedule monitor 2
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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:

```
RP/0/RP0/CPU0: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.

Configuring LSP Path Discovery

Perform this task to configure the LSP Path Discovery (LPD) and its required parameters, including echo interval, path, and scan.

Step 1 **configure**

Example:

```
RP/0/RP0/CPU0:router# configure
```

Enters global configuration mode.

Step 2 **ipsla**

Example:

```
RP/0/RP0/CPU0:router(config)# ipsla
```

Enters IP SLA configuration mode and configures IP service level agreements.

Step 3 **mpls lsp-monitor**

Example:

```
RP/0/RP0/CPU0:router(config-ipsla)# mpls lsp-monitor
```

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 **monitor** *monitor-id*

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-mplslm)# monitor 2
```

Configures an MPLS LSP monitor instance.

Step 5 **type mpls lsp ping**

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-mplslm-def)# type mpls lsp ping
```

Verifies the end-to-end connectivity of a label switched path (LSP) and the integrity of an MPLS network.

Step 6 **path discover**

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-mplslm-lsp-ping)# path discover
```

Enables LSP path discovery.

Step 7 **echo interval** *time*

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-mplslm-lsp-lpd)# echo interval 777
```

Configures the interval (in milliseconds) between MPLS LSP echo requests sent during path discovery. Range is 0 to 3600000. Default is 0.

Step 8 **echo maximum lsp selector ipv4** *host address*

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-mplslm-lsp-lpd)# echo maximum lsp selector ipv4 host_one
127.100.100.100
```

Configures a local host IP address (127.x.x.x) that is the maximum selector value to be used during path discovery. Default is 127.255.255.255.

Step 9 **echo multipath bitmap-size** *size*

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-mplslm-lsp-lpd)# echo multipath bitmap-size 50
```

Configures the maximum number of selectors sent in the downstream mapping of an MPLS LSP echo request during path discovery. Range is 1 to 256. Default is 32.

Step 10 **echo retry** *count*

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-mplslm-lsp-lpd)# echo retry 3
```

Configures the number of timeout retry attempts for MPLS LSP echo requests sent during path discovery. Range is 0 to 10. Default is 3.

Step 11 **echo timeout** *value*

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-mplslm-lsp-lpd)# echo timeout 300
```

Configures the timeout value for echo requests during path discovery. Range is 0 to 3600 in milliseconds. Default is 5.

Step 12 **path retry** *range*

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-mplslm-lsp-lpd)# path retry 12
```

Configures MPLS LSP path retry range. Range is 1 to 16. Default is 1.

Step 13 **path secondary frequency** {**both** | **connection-loss** | **timeout**} *value*}

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-mplslm-lsp-lpd)# path secondary frequency both 600
```

Enables secondary frequency for:

- Both timeout and connection loss
- Only connection loss
- Only timeout

Note There is no default value.

Step 14 **scan period** *value*

Example:

```
RP/0/RP0/CPU0:router(config-ipsla-mplslm-lsp-lpd)# scan period 60
```

Configures MPLS LSP scan time period value. Range is 0 to 7200 minutes. Default is 5.

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.

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
```