System Monitoring Configuration Guide for Cisco ASR 9000 Series Routers,IOS XR Release 6.3.x

First Published: 2017-09-01

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

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

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

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

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

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

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

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

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

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: https://www.cisco.com/c/en/us/about/legal/trademarks.html. Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1721R)

© 2017 Cisco Systems, Inc. All rights reserved.

CONTENTS

Preface

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

From Release 6.1.2 onwards, Cisco introduces support for the 64-bit Linux-based IOS XR operating system. Extensive feature parity is maintained between the 32-bit and 64-bit environments. Unless explicitly marked otherwise, the contents of this document are applicable for both the environments. For more details on Cisco IOS XR 64 bit, refer to the Release Notes for Cisco ASR 9000 Series Routers, Release 6.1.2 document.

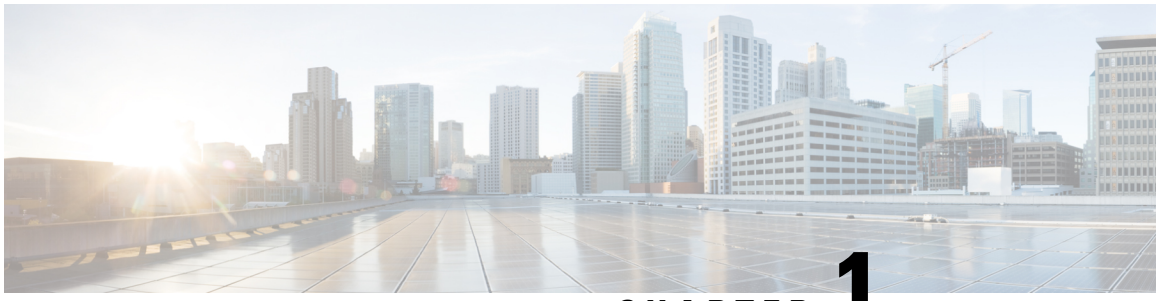
The *System Monitoring Configuration Guide for Cisco ASR 9000 Series Routers* preface contains these sections:

Communications, Services, and Additional Information

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

Cisco Bug Search Tool

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



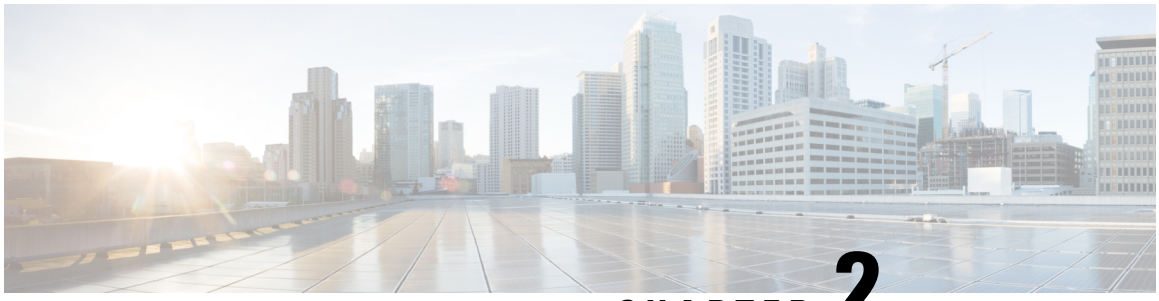
CHAPTER 1

New and Changed System Monitoring Features

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

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

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



CHAPTER 2

Implementing and Monitoring Alarms and Alarm Log Correlation

This module describes the concepts and tasks related to configuring alarm log correlation and monitoring alarm logs and correlated event records. 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 new and revised tasks you need to perform to implement logging correlation and monitor alarms on your network.

Note For more information about system logging on Cisco IOS XR Software and complete descriptions of the alarm management and logging correlation commands listed in this module, see the Related Documents, on page 41 section of this module.

Feature History for Implementing and Monitoring Alarms and Alarm Log Correlation

Release	Modification
Release 3.7.2	This feature was introduced.
Release 3.8.0	SNMP alarm correlation feature was added.

Prerequisites for Implementing and Monitoring Alarms and 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 Alarms and Alarm Log Correlation

Alarm Logging and Debugging Event Management System

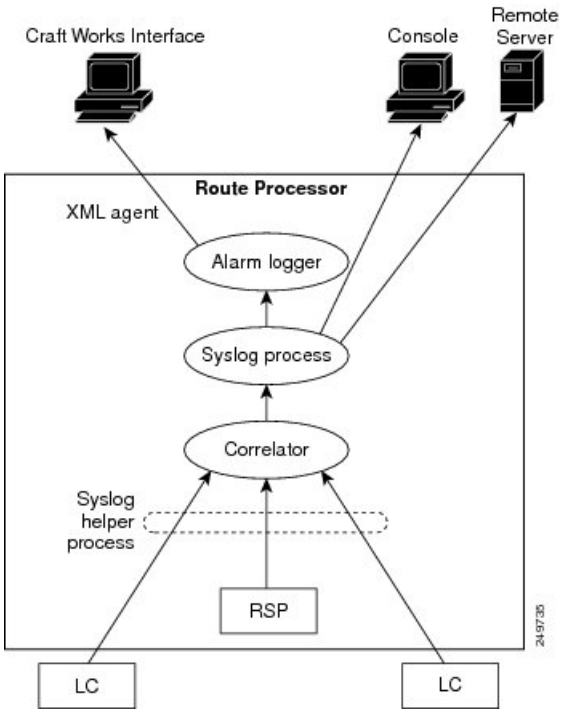
Cisco IOS XR Software Alarm Logging and Debugging Event Management System (ALDEMS) is used to monitor and store alarm messages that are forwarded by system servers and applications. In addition, ALDEMS correlates alarm messages forwarded due to a single root cause.

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 .

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 5 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.

Note For more information about logging correlation, see the Logging Correlation, on page 6 section.

System Logging Process

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

Alarm Logger

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

Note Alarms are prioritized in the logging events buffer. When it is necessary to overwrite an alarm record, the logging events buffer overwrites messages in the following order: nonbistate alarms first, then bistate alarms in the CLEAR state, and, finally, bistate alarms in the SET state. For more information about bistate alarms, see the Bistate Alarms, on page 8 section.

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.

Logging Correlation

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 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 ALDEMS caused by the accumulation of unnecessary messages. Each correlation rule hinges 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. Configure the appropriate set of messages in a logging correlation rule configuration to achieve correlation with a narrow or wide scope (depending on your objective).

Types of Correlation

There are two types of correlation that are configured in rules to isolate root-cause messages:

Nonstateful Correlation—This 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—This 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.

Application of Rules and Rule Sets

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.

In the case of a rule-set application, the behavior is the same; however, the apply configuration takes place for all rules that are part of the given rule set.

The **show logging correlator rule** command is used to display apply settings for a given rule, including those settings that have been configured with the **logging correlator apply ruleset** command.

Root Message and Correlated Messages

When a correlation rule is configured and applied, the correlator starts searching for a message match as specified in the rule. After a match is found, the correlator starts a timer corresponding to the timeout interval that is also specified in the rule. A message search for a match continues until the timer expires. Correlation occurs after the root-cause message is received.

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. See the Correlation Rules, on page 6 section to learn how the root-cause message is forwarded and stored.

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 1: Alarm Severity Levels for Event Logging

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

Bistate Alarms

Bistate alarms are generated by state changes associated with system hardware, such as a change of interface state from active to inactive, the online insertion and removal (OIR) of a 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 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 alarm state, which may be in the set state or the clear state.

Capacity Threshold Setting for Alarms

The capacity threshold setting determines when the alarm system begins reporting threshold crossing alarms. The capacity threshold for generating warning alarms is generally set at 80 percent of buffer capacity, but individual configurations may require different settings.

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 B
- 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.

Reissue Nonbistate Flag

The reissue nonbistate flag controls whether nonbistate alarms (events) are forwarded from the correlator log if their parent bistate root-cause alarm clears. Active bistate non-root-causes are always forwarded in this situation, because the condition is still present.

The reissue-nonbistate flag allows you to control whether non-bistate alarms are forwarded.

Internal Rules

Internal rules are defined on Cisco IOS XR Software and are used by protocols and processes within Cisco IOS XR Software. These rules are not customer configurable, but you may view them by using the **show logging correlator rule** command. All internal rule names are prefixed with [INTERNAL].

SNMP Alarm Correlation

In large-scale systems, such as Cisco IOS XR multi-chassis system , 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. So, this feature addresses these issues.

The objective of this SNMP alarm correlation feature is to:

- Extract the generic pieces of correlation functionality from the existing syslog correlator

- Create DLLs and APIs suitable for reusing the functionality in other components
- Integrate the SNMP agent with the DLLs to enable SNMP trap correlation

How to Implement and Monitor Alarm Management and Logging Correlation

Configuring Logging Correlation Rules

This task explains how to configure logging correlation rules.

The purpose of configuring logging correlation rules is to define the root cause and non-root-cause alarm messages (with message category, group, and code combinations) for logging correlation. The originating root-cause alarm message is forwarded to the syslog process, and all subsequent (non-root-cause) alarm messages are sent to the logging correlation buffer.

The fields inside a message that can be used for configuring correlation rules are as follows:

- Message category (for example, PKT_INFRA, MGBL, OS)
- Message group (for example, LINK, LINEPROTO, or OIR)
- Message code (for example, UPDOWN or GO_ACTIVE).

The logging correlator mechanism, running on the active route processor, begins queueing messages matching the ones specified in the correlation rules for the time specified in the timeout interval of the correlation rule.

The timeout interval begins when the correlator captures any alarm message specified for a given rule.

SUMMARY STEPS

1. **configure**
2. **logging correlator rule** *correlation-rule* { **type** { **stateful** | **nonstateful** }}
3. **timeout** [*milliseconds*]
4. Use the **commit** or **end** command.
5. **show logging correlator rule** {**all** | *correlation-rule1 ... correlation-rule14* } [**context** *context1 ... context 6*] [**location** *node-id1...node-id6*] [**rulesource** { **internal** | **user** }] [**ruletype** { **nonstateful** | **stateful** }] [**summary** | **detail**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**logging correlator rule** *correlation-rule* { **type** { **stateful** | **nonstateful** }}	Configures a logging correlation rule.

	Command or Action	Purpose
	Example: `RP/0/RSP0/CPU0:router(config)# logging correlator rule rule_stateful`	• Stateful correlations can change specifically if the root-cause alarm is bistate. • Nonstate correlations cannot change. All non-root-cause alarms remain in the correlation buffers.
Step 3	**timeout** [*milliseconds*] **Example:** `RP/0/RSP0/CPU0:router(config-corr-rule-st)# timeout 60000`	Specifies the collection period duration time for the logging correlator rule message. • Timeout begins when the first alarm message identified by the correlation rule is logged.
Step 4	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 5	**show logging correlator rule** {**all** | *correlation-rule1 ... correlation-rule14* } [**context** *context1 ... context 6*] [**location** *node-id1...node-id6*] [**rulesource** { **internal** | **user** }] [**ruletype** { **nonstateful** | **stateful** }] [**summary** | **detail**] **Example:** `RP/0/RSP0/CPU0:router# show logging correlator rule all`	(Optional) Displays defined correlation rules. • The output describes the configuration of each rule name, including the message category, group, and code information.

Configuring Logging Correlation Rule Sets

This task explains how to configure logging correlation rule sets.

SUMMARY STEPS

1. **configure**
2. **logging correlator ruleset** *ruleset*
3. **rulename** *rulename*
4. Use the **commit** or **end** command.
5. **show logging correlator ruleset** { **all** | *correlation-ruleset1...correlation-ruleset14* } [**detail** | **summary**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**logging correlator ruleset** *ruleset* **Example:** `RP/0/RSP0/CPU0:router(config)# logging correlator ruleset ruleset1`	Configures a logging correlation rule set.
Step 3	**rulename** *rulename* **Example:** `RP/0/RSP0/CPU0:router(config-corr-ruleset)# rulename stateful_rule`	Configures a rule name.
Step 4	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 5	**show logging correlator ruleset** { **all** | *correlation-ruleset1...correlation-ruleset14* } [**detail** | **summary**] **Example:** `RP/0/RSP0/CPU0:router# show logging correlator ruleset all`	(Optional) Displays defined correlation rule sets.

Configuring Root-cause and Non-root-cause Alarms

To correlate a root cause to one or more non-root-cause alarms and configure them to a rule, use the **rootcause** and **nonrootcause** commands specified for the correlation rule.

SUMMARY STEPS

1. **configure**
2. **logging correlator rule** *correlation-rule* { **type** { **stateful** | **nonstateful** }}

3. **rootcause** { *msg-category group-name msg-code* }
4. **nonrootcause**
5. **alarm** *msg-category group-name msg-code*
6. Use the **commit** or **end** command.
7. **show logging correlator rule** { **all** | *correlation-rule1...correlation-rule14* } [**context** *context1...context 6*] [**location** *node-id1...node-id6*] [**rulesource** { **internal** | **user** }] [**ruletype** { **nonstateful** | **stateful** }] [**summary** | **detail**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**logging correlator rule** *correlation-rule* { **type** { **stateful** | **nonstateful** }} **Example:** `RP/0/RSP0/CPU0:router(config)# logging correlator rule rule_stateful`	Configures a logging correlation rule and enters submodes for stateful and nonstateful rule types. • Stateful correlations can change specifically if the root-cause alarm is bistate. • Nonstate correlations cannot change. All non-root-cause alarms remain in the correlation buffers.
Step 3	**rootcause** { *msg-category group-name msg-code* } **Example:** `RP/0/RSP0/CPU0:router(config-corr-rule-st)# rootcause CAT_BI_1 GROUP_BI_1 CODE_BI_1`	Configures a root-cause alarm message. • This example specifies a root-cause alarm under stateful configuration mode
Step 4	**nonrootcause** **Example:** `RP/0/RSP0/CPU0:router(config-corr-rule-st)# nonrootcause`	Enters the non-root-cause configuration mode
Step 5	**alarm** *msg-category group-name msg-code* **Example:** `RP/0/RSP0/CPU0:router(config-corr-rule-st-nonrc)# alarm CAT_BI_2 GROUP_BI_2 CODE_BI_2`	Specifies a non-root-cause alarm message. • This command can be issued with the **nonrootcause** command, such as **nonrootcause alarm** *msg-category group-name msg-code*
Step 6	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:

	Command or Action	Purpose
		• **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 7	**show logging correlator rule** { **all** | *correlation-rule1...correlation-rule14* } [**context** *context1...context 6*] [**location** *node-id1...node-id6*] [**rulesource** { **internal** | **user** }] [**ruletype** { **nonstateful** | **stateful** }] [**summary** | **detail**] **Example:** `RP/0/RSP0/CPU0:router# show logging correlator rule all`	(Optional) Displays the correlator rules that are defined.

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 hinges 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 nonbistate alarms. This task explains how to implement these flags.

See the Reparent Flag, on page 10 and Reissue Nonbistate Flag, on page 10 sections for detailed information about these flags.

SUMMARY STEPS

1. **configure**
2. **logging correlator rule** *correlation-rule* { **type** { **stateful** | **nonstateful** }}
3. **reissue-nonbistate**
4. **reparent**
5. Use the **commit** or **end** command.
6. **show logging correlator rule** { **all** | *correlation-rule1...correlation-rule14* } [**context** *context1...context 6*] [**location** *node-id1...node-id6*] [**rulesource** { **internal** | **user** }] [**ruletype** { **nonstateful** | **stateful** }] [**summary** | **detail**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:**	Enters global configuration mode.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router# configure`	
Step 2	**logging correlator rule** *correlation-rule* { **type** { **stateful** | **nonstateful** }} **Example:** `RP/0/RSP0/CPU0:router(config)# logging correlator rule rule_stateful type nonstateful`	Configures a logging correlation rule. • Stateful correlations can change specifically if the root-cause alarm is bistate. • Nonstateful correlations cannot change. All non-root-cause alarms remain in the correlation buffers.
Step 3	**reissue-nonbistate** **Example:** `RP/0/RSP0/CPU0:router(config-corr-rule-st)# reissue-nonbistate`	Issues nonbistate alarm messages (events) from the correlator log after its root-cause alarm clears.
Step 4	**reparent** **Example:** `RP/0/RSP0/CPU0:router(config-corr-rule-st)# reparent`	Specifies the behavior of non-root-cause alarms after a root-cause parent clears.
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.
Step 6	**show logging correlator rule** { **all** | *correlation-rule1...correlation-rule14* } [**context** *context1...context 6*] [**location** *node-id1...node-id6*] [**rulesource** { **internal** | **user** }] [**ruletype** { **nonstateful** | **stateful** }] [**summary** | **detail**] **Example:** `RP/0/RSP0/CPU0:router# show logging correlator rule all`	(Optional) Displays the correlator rules that are defined.

What to do next

To activate a defined correlation rule and rule set, you must apply them by using the **logging correlator apply rule** and **logging correlator apply ruleset** commands.

Applying Logging Correlation Rules

This task explains how to apply logging correlation rules.

Applying a correlation rule activates it and gives a scope. A single correlation rule can be applied to multiple scopes on the router; that is, a rule can be applied to the entire router, to several locations, or to several contexts.

Note When a rule is applied or if a rule set that contains this rule is applied, then the rule definition cannot be modified through the configuration until the rule or rule set is once again unapplied.

Note It is possible to configure apply settings at the same time for both a rule and rule sets that contain the rule. In this case, the apply settings for the rule are the union of all these apply configurations.

SUMMARY STEPS

1. **configure**
2. **logging correlator apply rule** *correlation-rule*
3. Do one of the following:
   - **all-of-router**
   - **location** *node-id*
   - **context** *name*
4. Use the **commit** or **end** command.
5. **show logging correlator rule** { **all** | *correlation-rule1...correlation-rule14* } [**context** *context1...context 6*] [**location** *node-id1...node-id6*] [**rulesource** { **internal** | **user** }] [**ruletype** { **nonstateful** | **stateful** }] [**summary** | **detail**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**logging correlator apply rule** *correlation-rule* **Example:** `RP/0/RSP0/CPU0:router(config)# logging correlator apply-rule rule1`	Applies and activates a correlation rule and enters correlation apply rule configuration mode.
Step 3	Do one of the following: • **all-of-router**	• Applies a logging correlation rule to all nodes on the router. • Applies a logging correlation rule to a specific node on the router.

	Command or Action	Purpose

	• **location** *node-id*
	• **context** *name*
	Example:
	`RP/0/RSP0/CPU0:router(config-corr-apply-rule)# all-of-router`
	or
	`RP/0/RSP0/CPU0:router(config-corr-apply-rule)# location 0/2/CPU0`
	or
	`RP/0/RSP0/CPU0:router(config-corr-apply-rule)# logging correlator apply-rule rule2 context POS_0_0_0_0`
		• The location of the node is specified in the format *rack/slot/module*.
		• Applies a logging correlation rule to a specific context.

Step 4	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 5	**show logging correlator rule** { **all** | *correlation-rule1...correlation-rule14* } [**context** *context1...context 6*] [**location** *node-id1...node-id6*] [**rulesource** { **internal** | **user** }] [**ruletype** { **nonstateful** | **stateful** }] [**summary** | **detail**] **Example:** `RP/0/RSP0/CPU0:router# show logging correlator rule all`	(Optional) Displays the correlator rules that are defined.

Applying Logging Correlation Rule Sets

This task explains how to apply logging correlation rule sets.

Applying a correlation rule set activates it and gives a scope. When applied, a single rule-set configuration immediately effects the rules that are part of that given rule set.

Note Rule definitions that were previously applied (singly or as part of another rule set) cannot be modified until that rule or rule set is unapplied. Use the **no** form of the command to negate usage and then try to reapply rule set.

SUMMARY STEPS

1. **configure**
2. **logging correlator apply ruleset** *correlation-rule*
3. Do one of the following:
   - **all-of-router**
   - **location** *node-id*
   - **context** *name*
4. Use the **commit** or **end** command.
5. **show logging correlator ruleset** { **all** | *correlation-ruleset1 ... correlation-ruleset14* } [**detail** | **summary**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**logging correlator apply ruleset** *correlation-rule* **Example:** `RP/0/RSP0/CPU0:router(config)# logging correlator apply ruleset ruleset2`	Applies and activates a rule set and enters correlation apply rule set configuration mode.

Step 3	Do one of the following:
	• **all-of-router**
	• **location** *node-id*
	• **context** *name*
	Example:
	`RP/0/RSP0/CPU0:router(config-corr-ruleset)# all-of-router`
	or
	`RP/0/RSP0/CPU0:router(config-corr-ruleset)# location 0/2/CPU0`
	or
		• Applies a logging correlation rule set to all nodes on the router.
		• Applies a logging correlation rule set to a specific node on the router.
		  • The location of the node is specified in the format *rack/slot/module* .
		• Applies a logging correlation rule set to a specific context.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-corr-ruleset)# context`	
Step 4	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 5	**show logging correlator ruleset** { **all** | *correlation-ruleset1 ... correlation-ruleset14* } [**detail** | **summary**] **Example:** `RP/0/RSP0/CPU0:router# show logging correlator ruleset all`	(Optional) Displays the correlator rules that are defined.

Modifying Logging Events Buffer Settings

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.

This task involves the following steps:

- Modifying logging events buffer size
- Setting threshold for generating alarms
- Setting the alarm filter (severity)

Caution Modifications to alarm settings that lower the severity level for reporting alarms and threshold for generating capacity-warning alarms may slow system performance.

Caution Modifying the logging events buffer size clears the buffer of all event records except for the bistate alarms in the set state.

SUMMARY STEPS

1. **show logging events info**

2. **configure**
3. **logging events buffer-size** *bytes*
4. **logging events threshold** *percent*
5. **logging events level** *severity*
6. Use the **commit** or **end** command.
7. **show logging events info**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**show logging events info** **Example:** `RP/0/RSP0/CPU0:router# show logging events info`	(Optional) Displays the size of the logging events buffer (in bytes), the percentage of the buffer that is occupied by alarm-event records, capacity threshold for reporting alarms, total number of records in the buffer, and severity filter, if any.
Step 2	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 3	**logging events buffer-size** *bytes* **Example:** `RP/0/RSP0/CPU0:router(config)# logging events buffer-size 50000`	Specifies the size of the alarm record buffer. • In this example, the buffer size is set to 50000 bytes.
Step 4	**logging events threshold** *percent* **Example:** `RP/0/RSP0/CPU0:router(config)# logging events threshold 85`	Specifies the percentage of the logging events buffer that must be filled before the alarm logger generates a threshold-crossing alarm. • In this example, the alarm logger generates athreshold-crossing alarm notification when the event buffer reaches 85 percent of capacity.
Step 5	**logging events level** *severity* **Example:** `RP/0/RSP0/CPU0:router(config)# logging events level warnings`	Sets the severity level that determines which logging events are displayed. (See Table 1: Alarm Severity Levels for Event Logging , on page 8 under the Alarm Severity Level and Filtering, on page 7 section for a list of the severity levels.) • Keyword options are as follows: **emergencies**, **alerts**, **critical**, **errors**, **warnings**, **notifications**, and **informational**. • In this example, messages with a warning (Level 4) severity or greater are written to the alarm log. Messages of a lesser severity (notifications and informational messages) are not recorded.
Step 6	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 7	**show logging events info** **Example:** `RP/0/RSP0/CPU0:router# show logging events info`	(Optional) Displays the size of the logging events buffer (in bytes), percentage of the buffer that is occupied by alarm-event records, capacity threshold for reporting alarms, total number of records in the buffer, and severity filter, if any. • This command is used to verify that all settings have been modified and that the changes have been accepted by the system.

Modifying Logging Correlator Buffer Settings

This task explains how to modify the logging correlator buffer settings.

The size of the logging correlator buffer can be adjusted to accommodate the anticipated volume of incoming correlated messages. Records can be removed from the buffer by correlation ID, or the buffer can be cleared of all records.

SUMMARY STEPS

1. **configure**
2. **logging correlator buffer-size** *bytes*
3. **exit**
4. **show logging correlator info**
5. **clear logging correlator delete** *correlation-id*
6. **clear logging correlator delete all-in-buffer**
7. **show logging correlator buffer** { **all-in-buffer** [**ruletype** [**nonstateful** | **stateful**]] | [**rulesource** [**internal** | **user**]] | **rule-name** *correlation-rule1...correlation-rule14* | **correlationID** *correlation-id1..correlation-id14* }

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.

	Command or Action	Purpose
Step 2	**logging correlator buffer-size** *bytes* **Example:** `RP/0/RSP0/CPU0:router(config)# logging correlator buffer-size 100000`	Specifies the size of the logging correlator buffer. • In this example, the size of the logging correlator buffer is set to 100,000 bytes.
Step 3	**exit** **Example:** `RP/0/RSP0/CPU0:router(config)# exit`	Exits global configuration mode and returns the router to EXEC mode.
Step 4	**show logging correlator info** **Example:** `RP/0/RSP0/CPU0:router# show logging correlator info`	(Optional) Displays information about the size of the logging correlator buffer and percentage of the buffer occupied by correlated messages
Step 5	**clear logging correlator delete** *correlation-id* **Example:** `RP/0/RSP0/CPU0:router# clear logging correlator delete 48 49 50`	(Optional) Removes a particular correlated event record or records from the logging correlator buffer. • A range of correlation IDs can also be specified for removal (up to 32 correlation IDs, separated by a space).
Step 6	**clear logging correlator delete all-in-buffer** **Example:** `RP/0/RSP0/CPU0:router# clear logging correlator delete all-in-buffer`	(Optional) Clears all correlated event messages from the logging correlator buffer.
Step 7	**show logging correlator buffer** { **all-in-buffer** [**ruletype** [**nonstateful** | **stateful**]] | [**rulesource** [**internal** | **user**]] | **rule-name** *correlation-rule1...correlation-rule14* | **correlationID** *correlation-id1..correlation-id14* } **Example:** `RP/0/RSP0/CPU0:router# show logging correlator buffer all-in-buffer`	(Optional) Displays the contents of the correlated event record. • Use this step to verify that records for particular correlation IDs have been removed from the correlated event log.

Displaying Alarms by Severity and Severity Range

This task explains how to display alarms by severity and severity range.

Alarms can be displayed according to severity level or a range of severity levels. Severity levels and their respective system conditions are listed in Table 1: Alarm Severity Levels for Event Logging , on page 8 under the Alarm Severity Level and Filtering, on page 7 section.

Note The commands can be entered in any order.

SUMMARY STEPS

1. **show logging events buffer severity-lo-limit** *severity*
2. **show logging events buffer severity-hi-limit** *severity*
3. **show logging events buffer severity-hi-limit** *severity* **severity-lo-limit** *severity*
4. **show logging events buffer severity-hi-limit** *severity* **severity-lo-limit** *severity* **timestamp-lo-limit** *hh* **:** *mm* **:** *ss* [*month*] [*day*] [*year*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**show logging events buffer severity-lo-limit** *severity* **Example:** `RP/0/RSP0/CPU0:router# show logging events buffer severity-lo-limit notifications`	(Optional) Displays logging events with a severity at or below the numeric value of the specified severity level. • In this example, alarms with a severity of notifications (severity of 5) or lower are displayed. Informational (severity of 6) messages are omitted. **Note** Use the **severity-lo-limit** keyword and the *severity* argument to specify the severity level *description*, not the numeric value assigned to that severity level.
Step 2	**show logging events buffer severity-hi-limit** *severity* **Example:** `RP/0/RSP0/CPU0:router# show logging events buffer severity-hi-limit critical`	(Optional) Displays logging events with a severity at or above the numeric value specified severity level. • In this example, alarms with a severity of critical (severity of 2) or greater are displayed. Alerts (severity of 1) and emergencies (severity of 0) are omitted. **Note** Use the **severity-hi-limit** keyword and the *severity* argument to specify the severity level *description*, not the numeric value assigned to that severity level.
Step 3	**show logging events buffer severity-hi-limit** *severity* **severity-lo-limit** *severity* **Example:** `RP/0/RSP0/CPU0:router# show logging events buffer severity-hi-limit alerts severity-lo-limit critical`	(Optional) Displays logging events within a severity range. • In this example, alarms with a severity of critical (severity of 2) and alerts (severity of 1) are displayed. All other event severities are omitted.
Step 4	**show logging events buffer severity-hi-limit** *severity* **severity-lo-limit** *severity* **timestamp-lo-limit** *hh* **:** *mm* **:** *ss* [*month*] [*day*] [*year*]	(Optional) Displays logging events occurring after the specified time stamp and within a severity range. The *month*,

	Command or Action	Purpose
	Example: `RP/0/RSP0/CPU0:router# show logging events buffer severity-lo-limit warnings severity-hi-limit critical timestamp-lo-limit 22:00:00 may 07 04`	*day*, and *year* arguments default to the current month, date, and year, if not specified. • In this example, alarms with a severity of warnings (severity of 4), errors (severity of 3), and critical (severity of 2) that occur after 22:00:00 on May 7, 2004 are displayed. All other messages occurring before the time stamp are omitted.

Displaying Alarms According to a Time Stamp Range

Alarms can be displayed according to a time stamp range. Specifying a specific beginning and endpoint can be useful in isolating alarms occurring during a particular known system event.

This task explains how to display alarms according to a time stamp range.

Note The commands can be entered in any order.

SUMMARY STEPS

1. **show logging events buffer timestamp-lo-limit** *hh* **:** *mm* **:** *ss* [*month*] [*day*] [*year*]
2. **show logging events buffer timestamp-hi-limit** *hh* **:** *mm* **:** *ss* [*month*] [*day*] [*year*]
3. **show logging events buffer timestamp-hi-limit** *hh* **:** *mm* **:** *ss* [*month*] [*day*] [*year*] **timestamp-lo-limit** *hh* **:** *mm* **:** *ss* [*month*] [*day*] [*year*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**show logging events buffer timestamp-lo-limit** *hh* **:** *mm* **:** *ss* [*month*] [*day*] [*year*] **Example:** `RP/0/RSP0/CPU0:router# show logging events buffer timestamp-lo-limit 21:28:00 april 18 04`	(Optional) Displays logging events with a time stamp after the specified time and date. • The *month*, *day*, and *year* arguments default to the current month, date, and year if not specified. • The sample output displays events logged after 21:28:00 on April 18, 2004.
Step 2	**show logging events buffer timestamp-hi-limit** *hh* **:** *mm* **:** *ss* [*month*] [*day*] [*year*] **Example:** `RP/0/RSP0/CPU0:router# show logging events buffer timestamp-hi-limit 21:28:03 april 18 04`	(Optional) Displays logging events with a time stamp before the specified time and date. • The *month*, *day*, and *year* arguments default to the current month, date, and year if not specified. • The sample output displays events logged before 21:28:03 on April 18, 2004.

	Command or Action	Purpose
Step 3	**show logging events buffer timestamp-hi-limit** *hh* **:** *mm* **:** *ss* [*month*] [*day*] [*year*] **timestamp-lo-limit** *hh* **:** *mm* **:** *ss* [*month*] [*day*] [*year*] **Example:** `RP/0/RSP0/CPU0:router# show logging events buffer timestamp-hi-limit 21:28:00 april 18 04 timestamp-lo-limit 21:16:00 april 18 03`	(Optional) Displays logging events with a time stamp after and before the specified time and date. • The *month*, *day*, and *year* arguments default to the current month, day, and year if not specified. • The sample output displays events logged after 21:16:00 on April 18, 2003 and before 21:28:00 on April 18, 2004.

Displaying Alarms According to Message Group and Message Code

This task explains how to display alarms in the logging events buffer according to message code and message group.

Displaying alarms by message group and message code can be useful in isolating related events.

Note The commands can be entered in any order.

SUMMARY STEPS

1. **show logging events buffer group** *message-group*
2. **show logging events buffer message** *message-code*
3. **show logging events buffer group** *message-group* **message** *message-code*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**show logging events buffer group** *message-group* **Example:** `RP/0/RSP0/CPU0:router# show logging events buffer group SONET`	(Optional) Displays logging events matching the specified message group. • In this example, all events that contain the message group SONET are displayed.
Step 2	**show logging events buffer message** *message-code* **Example:** `RP/0/RSP0/CPU0:router# show logging events buffer message ALARM`	(Optional) Displays logging events matching the specified message code. • In this example, all events that contain the message code ALARM are displayed.
Step 3	**show logging events buffer group** *message-group* **message** *message-code* **Example:** `RP/0/RSP0/CPU0:router# show logging events buffer group SONET message ALARM`	(Optional) Displays logging events matching the specified message group and message code. • In this example, all events that contain the message group SONET and message code ALARM are displayed.

Displaying Alarms According to a First and Last Range

This task explains how to display alarms according to a range of the first and last alarms in the logging events buffer.

Alarms can be displayed according to a range, beginning with the first or last alarm in the logging events buffer.

Note The commands can be entered in any order.

SUMMARY STEPS

1. **show logging events buffer first** *event-count*
2. **show logging events buffer last** *event-count*
3. **show logging events buffer first** *event-count* **last** *event-count*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**show logging events buffer first** *event-count* **Example:** `RP/0/RSP0/CPU0:router# show logging events buffer first 15`	(Optional) Displays logging events beginning with the first event in the logging events buffer. • For the *event-count* argument, enter the number of events to be displayed. • In this example, the first 15 events in the logging events buffer are displayed.
Step 2	**show logging events buffer last** *event-count* **Example:** `RP/0/RSP0/CPU0:router# show logging events buffer last 20`	(Optional) Displays logging events beginning with the last event in the logging events buffer. • For the *event-count* argument, enter the number of events to be displayed. • In this example, the last 20 events in the logging events buffer are displayed.
Step 3	**show logging events buffer first** *event-count* **last** *event-count* **Example:** `RP/0/RSP0/CPU0:router# show logging events buffer first 20 last 20`	(Optional) Displays the first and last events in the logging events buffer. • For the *event-count* argument, enter the number of events to be displayed. • In this example, both the first 20 and last 20 events in the logging events buffer are displayed.

Displaying Alarms by Location

This task explains how to display alarms by location.

Note The commands can be entered in any order.

SUMMARY STEPS

1. **show logging events buffer location** *node-id*
2. **show logging events buffer location** *node-id* **event-hi-limit** *event-id* **event-lo-limit** *event-id*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**show logging events buffer location** *node-id* **Example:** `RP/0/RSP0/CPU0:router# show logging events buffer 0/2/CPU0`	(Optional) Isolates the occurrence of the range of event IDs to a particular node. • The location of the node is specified in the format *rack/slot/module*.
Step 2	**show logging events buffer location** *node-id* **event-hi-limit** *event-id* **event-lo-limit** *event-id* **Example:** `RP/0/RSP0/CPU0:router# show logging events buffer location 0/2/CPU0 event-hi-limit 100 event-lo-limit 1`	(Optional) Isolates the occurrence of the range of event IDs to a particular node and narrows the range by specifying a high and low limit of event IDs to be displayed. • The location of the node is specified in the format *rack/slot/module*.

Displaying Alarms by Event Record ID

This task explains how to display alarms by event record ID.

Note The commands can be entered in any order.

SUMMARY STEPS

1. **show logging events buffer all-in-buffer**
2. **show logging events buffer event-hi-limit** *event-id* **event-lo-limit** *event-id*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**show logging events buffer all-in-buffer** **Example:** `RP/0/RSP0/CPU0:router# show logging events buffer all-in-buffer`	(Optional) Displays all messages in the logging events buffer. **Caution** Depending on the alarm severity settings, use of this command can create a large amount of output.

	Command or Action	Purpose
Step 2	**show logging events buffer event-hi-limit** *event-id* **event-lo-limit** *event-id* **Example:** `RP/0/RSP0/CPU0:router# show logging events buffer event-hi-limit 100 event-lo-limit 1`	(Optional) Narrows the range by specifying a high and low limit of event IDs to be displayed.

Displaying the Logging Correlation Buffer Size, Messages, and Rules

This task explains how to display the logging correlation buffer size, messages in the logging correlation buffer, and correlation rules.

Note The commands can be entered in any order.

SUMMARY STEPS

1. **show logging correlator info**
2. **show logging correlator buffer all-in-buffer**
3. **show logging correlator buffer correlationID** *correlation-id*
4. **show logging correlator buffer rule-name** *correlation-rule*
5. **show logging correlator rule all**
6. **show logging correlator rule** *correlation-rule*
7. **show logging correlator ruleset all**
8. **show logging correlator ruleset** *ruleset-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**show logging correlator info** **Example:** `RP/0/RSP0/CPU0:router# show logging correlator info`	(Optional) Displays the size of the logging correlation buffer (in bytes) and the percentage occupied by correlated messages.
Step 2	**show logging correlator buffer all-in-buffer** **Example:** `RP/0/RSP0/CPU0:router# show logging correlator buffer all-in-buffer`	(Optional) Displays all messages in the logging correlation buffer.
Step 3	**show logging correlator buffer correlationID** *correlation-id* **Example:**	(Optional) Displays specific messages matching a particular correlation ID in the correlation buffer.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router# show logging correlator buffer correlationID 37`	
Step 4	**show logging correlator buffer rule-name** *correlation-rule* **Example:** `RP/0/RSP0/CPU0:router# show logging correlator buffer rule-name rule7`	(Optional) Displays specific messages matching a particular rule in the correlation buffer.
Step 5	**show logging correlator rule all** **Example:** `RP/0/RSP0/CPU0:router# show logging correlator rule all`	(Optional) Displays all defined correlation rules.
Step 6	**show logging correlator rule** *correlation-rule* **Example:** `RP/0/RSP0/CPU0:router# show logging correlator rule rule7`	(Optional) Displays the specified correlation rule.
Step 7	**show logging correlator ruleset all** **Example:** `RP/0/RSP0/CPU0:router# show logging correlator ruleset all`	(Optional) Displays all defined correlation rule sets.
Step 8	**show logging correlator ruleset** *ruleset-name* **Example:** `RP/0/RSP0/CPU0:router# show logging correlator ruleset ruleset_static`	(Optional) Displays the specified correlation rule set.

Clearing Alarm Event Records and Resetting Bistate Alarms

This task explains how to clear alarm event records and bistate alarms.

Unnecessary and obsolete messages can be cleared to reduce the size of the event logging buffer and make it more searchable, and thus more navigable.

The filtering capabilities available for clearing events in the logging events buffer (with the **clear logging events delete** command) are also available for displaying events in the logging events buffer (with the **show logging events buffer** command).

Note The commands can be entered in any order.

SUMMARY STEPS

1. **show logging events buffer all-in-buffer**
2. **clear logging events delete timestamp-lo-limit** *hh* **:** *mm* **:** *ss* [*month*] [*day*] [*year*]
3. **clear logging events delete event-hi-limit** *severity* **event-lo-limit** *severity*
4. **clear logging events delete location** *node-id*
5. **clear logging events delete first** *event-count*
6. **clear logging events delete last** *event-count*
7. **clear logging events delete message** *message-code*
8. **clear logging events delete group** *message-group*
9. **clear logging events reset all-in-buffer**
10. **show logging events buffer all-in-buffer**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**show logging events buffer all-in-buffer** **Example:** `RP/0/RSP0/CPU0:router# show logging events buffer all-in-buffer`	It retains the messages before the specified time and displayed the messages after the timestamp. The timestamp-lo-limit specifies the lower time limit. Similarly timestamp-hi-limit specifies the higher time limit of a time window. All events within this time window will be displayed. The default value of the timestamp-lo-limit is the timestamp of the earliest event in the buffer. The timestamp-hi-limit is the timestamp of the latest event in the buffer.
Step 2	**clear logging events delete timestamp-lo-limit** *hh* **:** *mm* **:** *ss* [*month*] [*day*] [*year*] **Example:** `RP/0/RSP0/CPU0:router# clear logging events delete timestamp-lo-limit 20:00:00 april 01 2004`	It retains the messages before the specified time and deletes the messages after the timestamp. The timestamp-lo-limit specifies the lower time limit. Similarly timestamp-hi-limit specifies the higher time limit of a time window. All events within this time window will be deleted. The default value of the timestamp-lo-limit is the timestamp of the earliest event in the buffer. The timestamp-hi-limit is the timestamp of the latest event in the buffer.
Step 3	**clear logging events delete event-hi-limit** *severity* **event-lo-limit** *severity* **Example:** `RP/0/RSP0/CPU0:router# clear logging events delete event-hi-limit warnings event-lo-limit informational`	(Optional) Deletes logging events within a range of severity levels for logging alarm messages. • In this example, all events with a severity level of warnings, notifications, and informational are deleted.
Step 4	**clear logging events delete location** *node-id* **Example:** `RP/0/RSP0/CPU0:router# clear logging events delete location 0/2/CPU0`	(Optional) Deletes logging events from the logging events that have occurred on a particular node. • The location of the node is specified in the format *rack/slot/module*.

	Command or Action	Purpose
Step 5	**clear logging events delete first** *event-count* **Example:** `RP/0/RSP0/CPU0:router# clear logging events delete first 10`	(Optional) Deletes logging events beginning with the first event in the logging events buffer. • In this example, the first 10 events in the logging events buffer are cleared.
Step 6	**clear logging events delete last** *event-count* **Example:** `RP/0/RSP0/CPU0:router# clear logging events delete last 20`	(Optional) Deletes logging events beginning with the last event in the logging events buffer. • In this example, the last 20 events in the logging events buffer are cleared.
Step 7	**clear logging events delete message** *message-code* **Example:** `RP/0/RSP0/CPU0:router# clear logging events delete message sys`	(Optional) Deletes logging events that contain the specified message code. • In this example, all events that contain the message code SYS are deleted from the logging events buffer.
Step 8	**clear logging events delete group** *message-group* **Example:** `RP/0/RSP0/CPU0:router# clear logging events delete group config_i`	(Optional) Deletes logging events that contain the specified message group. • In this example, all events that contain the message group CONFIG_I are deleted from the logging events buffer.
Step 9	**clear logging events reset all-in-buffer** **Example:** `RP/0/RSP0/CPU0:router# clear logging events reset all-in-buffer`	(Optional) Clears all bistate alarms in the SET state from the logging events buffer.
Step 10	**show logging events buffer all-in-buffer** **Example:** `RP/0/RSP0/CPU0:router# show logging events buffer all-in-buffer`	(Optional) Displays all messages in the logging events buffer.

Defining SNMP Correlation Buffer Size

This task explains how to define correlation buffer size for SNMP traps.

SUMMARY STEPS

1. **configure**
2. **snmp-server correlator buffer-size** *bytes*
3. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**snmp-server correlator buffer-size** *bytes* **Example:** `RP/0/RSP0/CPU0:router(config)# snmp-server correlator buffer-size 600`	Defines the buffer size that can store SNMP correlation traps. The default size is 64KB. You can clear the correlation buffers manually or the buffer wraps automatically, wherein the oldest correlations are purged to accommodate the newer correlations.
Step 3	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.

Defining SNMP Rulesets

This task defines 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.

SUMMARY STEPS

1. **configure**
2. **snmp-server correlator ruleset** *name* **rulename** *name*
3. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.

	Command or Action	Purpose
Step 2	**snmp-server correlator ruleset** *name* **rulename** *name* **Example:** `RP/0/RSP0/CPU0:router(config)# snmp-server correlator ruleset rule1 rulename rule2 host ipv4 address 1.2.3.4 host ipv4 address 2.3.4.5 port 182`	Specifies a ruleset that allows you to group two or more rules into a group and apply that group to a set of hosts.
Step 3	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 SNMP Correlation Rules

This task explains how to configure SNMP correlation rules.

The purpose of configuring SNMP trap correlation rules is to define the correlation rules or non-correlation rules and apply them to specific trap destinations.

SUMMARY STEPS

1. **configure**
2. **snmp-server correlator rule** *rule_name* { **nonrootcause trap** *trap_oid* **varbind** *vbind_OID* { **index** | **value** } **regex** *line* | **rootcause trap** *trap_oid* **varbind** *vbind_OID* { **index** | **value** } **regex** *line* | **timeout** }
3. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.

	Command or Action	Purpose
Step 2	**snmp-server correlator rule** *rule_name* { **nonrootcause trap** *trap_oid* **varbind** *vbind_OID* { **index** | **value** } **regex** *line* | **rootcause trap** *trap_oid* **varbind** *vbind_OID* { **index | value** } **regex** *line* | **timeout** } **Example:** `RP/0/RSP0/CPU0:router(config)# snmp-server correlator rule test rootcause A varbind A1 value regex RA1 varbind A2 index regex RA2 timeout 5000 nonrootcause trap B varbind B1 index regex RB1 varbind B2 value regex RB2 trap C varbind C1 value regex RC1 varbind C2 value regex RC2`	Configures a SNMP correlation rule. You can specify the numeric rootcause trap OID or non-rootcause trap matching definitions. • Specifies a numeric non-rootcause trap OID and, optionally, one or more numeric varbinds specific to the non-rootcause trap that must ALL also be matched to have found a valid non-rootcause for this rule. The POSIX regexp specifies a regular expression that the value that the vbind index or value must match. • Specifies a numeric rootcause trap OID and, optionally, one or more numeric varbinds specific to the rootcause trap that must ALL also be matched to have found a valid rootcause for this rule. The POSIX regexp specifies a regular expression that the vbind index or value must match. **Note** You can specify the timeout for detection of a correlation after receipt of first rootcause or non-rootcause in this specified rule. The range is from 1 to 600000 milliseconds. **Note** All OID values for traps and varbinds are verified and rejected, if they do not match valid OIDs supported by IOS XR.
Step 3	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.

Applying SNMP Correlation Rules

The purpose of this task is to apply the SNMP trap correlation rules to specific trap destinations.

SUMMARY STEPS

1. **configure**
2. **snmp-server correlator apply rule** *rule-name* [**all-hosts** \| **host ipv4 address** *address* [*port*]

3. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**snmp-server correlator apply rule** *rule-name* [**all-hosts** | **host ipv4 address** *address* [*port*] **Example:** `RP/0/RSP0/CPU0:router# snmp-server correlator apply rule ifupdown host ipv4 address 1.2.3.4 host ipv4 address 2.3.4.5 port 182`	Applies the SNMP trap correlation rules to specific trap destinations. You have an option of applying the rule to traps destined for all trap hosts, or to a specific subset by specifying individual IP addresses and optional ports.
Step 3	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.

Applying SNMP Correlation Ruleset

The purpose of this task is to apply the set of two SNMP trap correlation rules or more rules as a group to specific trap destinations.

SUMMARY STEPS

1. **configure**
2. **snmp-server correlator apply ruleset** *ruleset-name* [**all-hosts** \| **host ipv4 address** *address* [*port*]
3. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:**	Enters global configuration mode.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router# configure`	
Step 2	**snmp-server correlator apply ruleset** *ruleset-name* [**all-hosts** | **host ipv4 address** *address* [*port*] **Example:** `RP/0/RSP0/CPU0:router# snmp-server correlator apply ruleset ruleset_1 host ipv4 address 1.2.3.4 host ipv4 address 2.3.4.5 port 182`	Applies the SNMP trap correlation ruleset to specific trap destinations. You have an option of applying the set of two or more SNMP trap correlation rules to traps destined for all trap hosts, or to a specific subset by specifying individual IP addresses and optional ports.
Step 3	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.

Asynchronous Syslog Communication

The asynchronous syslog communication feature enables proper ordering of messages testing on each node (LC, RP), non dropping of messages generated from multiple clients on each node (LC, RP) and checking performance, scalability and latency by sending log messages at incremental rates.

This feature enables the following:

- Proper ordering of messages testing on MC min 4+1.
- Non dropping of messages generated from multiple clients on MC min 4+1.
- Syslogd_helper message handling capacity - flood lots of syslog messages using test client (logger), verify if no syslog message is lost (specified rate as per new design).
- 1200/1500 msgs/sec from every node - restart restart/crash syslogd_helper on LCs and RP/correlatord and syslogd on RP.
- Configure the routing protocol ospf. Configure 5k neighbors using sub interfaces. Perform interface flapping to generate log messages and check syslogd_helper performance.
- Enable debug for few heavy processes - sysdb/gsp

Configuration Examples for Alarm Management and Logging Correlation

This section provides these configuration examples:

Increasing the Severity Level for Alarm Filtering to Display Fewer Events and Modifying the Alarm Buffer Size and Capacity Threshold: Example

This configuration example shows how to set the capacity threshold to 90 percent, to reduce the size of the logging events buffer to 10,000 bytes from the default, and to increase the severity level to errors:

```
!
logging events threshold 90
logging events buffer-size 10000
logging events level errors
!
```

Increasing the severity level to errors reduces the number of alarms that are displayed in the logging events buffer, because only alarms with a severity of errors or higher are displayed. Increasing the threshold capacity to 90 percent reduces the time interval between the threshold crossing and wraparound events; the logging events buffer thus does not generate a threshold-crossing alarm until it reaches 90 percent capacity. Reducing the size of the logging events buffer to 10,000 bytes decreases the number of alarms that are displayed in the logging events buffer and reduces the memory requirements for the component.

Configuring a Nonstateful Correlation Rule to Permanently Suppress Node Status Messages: Example

This example shows how to configure a nonstateful correlation rule to permanently suppress node status messages:

```
logging correlator rule node_status type nonstateful
timeout 4000
  rootcause PLATFORM INVMGR NODE_STATE_CHANGE
  nonrootcause
    alarm PLATFORM SYSLDR LC_ENABLED
    alarm PLATFORM ALPHA_DISPLAY CHANGE
  !
!
logging correlator apply rule node_status

  all-of-router
!
```

In this example, three similar messages are identified as forwarded to the syslog process simultaneously after a card boots:

PLATFORM-INVMGR-6-NODE_STATE_CHANGE : Node: 0/1/CPU0, state: IOS XR RUN

PLATFORM-SYSLDR-5-LC_ENABLED : LC in slot 1 is now running IOX

PLATFORM-ALPHA_DISPLAY-6-CHANGE : Alpha display on node 0/1/CPU0 changed to IOX RUN in state default

These messages are similar. To see only one message appear in the logs, one of the messages is designated as the root cause message (the one that appears in the logs), and the other messages are considered non-root-cause messages.

The root-cause message is typically the one that arrives earliest, but that is not a requirement.

```
logging correlator rule node_status type nonstateful
  timeout 4000
  rootcause PLATFORM INVMGR NODE_STATE_CHANGE
  nonrootcause
    alarm PLATFORM SYSLDR LC_ENABLED
    alarm PLATFORM ALPHA_DISPLAY CHANGE
  !
!
```

In this example, the correlation rule named node_status is configured to correlate the PLATFORM INVMGR NODE_STATE_CHANGE alarm (the root-cause message) with the PLATFORM SYSLDR LC_ENABLED and PLATFORM ALPHA_DISPLAY CHANGE alarms. The updown correlation rule is applied to the entire router.

```
logging correlator apply rule node_status
  all-of-router
!
```

After a card boots and sends these messages:

PLATFORM-INVMGR-6-NODE_STATE_CHANGE : Node: 0/1/CPU0, state: IOS XR RUN

PLATFORM-SYSLDR-5-LC_ENABLED : LC in slot 1 is now running IOX

PLATFORM-ALPHA_DISPLAY-6-CHANGE : Alpha display on node 0/1/CPU0 changed to IOX RUN in state default

the correlator forwards the PLATFORM-INVMGR-6-NODE_STATE_CHANGE message to the syslog process, while the remaining two messages are held in the logging correlator buffer.

In this example, the show sample output from the **show logging events buffer all-in-buffer** command displays the alarms stored in the logging events buffer after the 4-second time period expires for the node_status correlation rule:

```
RP/0/RSP0/CPU0:router# show logging events buffer all-in-buffer

#ID :C_id:Source :Time :%CATEGORY-GROUP-SEVERITY-MESSAGECODE: Text

#76 :12  :RP/0/0/CPU0:Aug  2 22:32:43 : invmgr[194]:

%PLATFORM-INVMGR-6-NODE_STATE_CHANGE : Node: 0/1/CPU0, state: IOS XR RUN
```

The **show logging correlator buffer** correlation ID command generates the following output after the one minute interval expires. The output displays the alarms assigned correlation ID 12 in the logging correlator buffer.

```
RP/0/RSP0/CPU0:router# show logging correlator buffer correlationID 46

#C_id.id:Rule Name:Source :Time : Text

#12.1 :nodestatus:RP/0/0/CPU0:Aug  2 22:32:43 : invmgr[194]:
```

```
%PLATFORM-INVMGR-6-NODE_STATE_CHANGE : Node: 0/1/CPU0, state: IOS XR RUN
#12.2 :nodestatus:RP/0/0/CPU0:Aug  2 22:32:43 : sysldr[336]: %PLATFORM-SYSLDR-5-LC_ENABLED
 : LC in slot 1 is now running IOX
#12.3 :nodestatus:RP/0/0/CPU0:Aug  2 22:32:44 : alphadisplay[102]:
%PLATFORM-ALPHA_DISPLAY-6-CHANGE : Alpha display on node 0/1/CPU0 changed to IOX RUN  in
state default
Because this rule was defined as nonstateful, these messages are held in the buffer
indefinitely.
```

Configuring a Stateful Correlation Rule for LINK UPDOWN and SONET ALARM Alarms: Example

This example shows how to configure a correlation rule for the LINK UPDOWN and SONET ALARM messages:

```
!
logging correlator rule updown type stateful
  timeout 10000
  rootcause PKT_INFRA LINK UPDOWN
  nonrootcause
    alarm L2 SONET ALARM
  !
!
logging correlator apply rule updown
  all-of-router
!
```

In this example, suppose that two routers are connected. When the correlator receives a root-cause message, the correlator sends it directly to the syslog process. Subsequent PKT_INFRA-LINK- UPDOWN or L2-SONET-ALARM messages matching the rule are considered leaf messages and are stored in the logging correlator buffer. If, for any reason, a leaf message (such as the L2-SONET-ALARM alarm in this example) is received first, the correlator does not send it to the logging events buffer immediately; the correlator, instead, waits until the timeout interval expires. After the timeout, if the root message is never received, all messages in the logging correlator buffer received during the timeout interval are forwarded to the syslog process.

In this example, the correlation rule named updown is configured to correlate the PKT_INFRA-LINK-UPDOWN alarm (the root message) and L2-SONET-ALARM alarms (leaf messages associated with PKT_INFRA-LINK-UPDOWN alarms).

```
logging correlator rule updown type stateful
  timeout 10000
  rootcause PKT_INFRA LINK UPDOWN
  nonrootcause
    alarm L2 SONET ALARM
In this example, the updown correlation rule is applied to the entire router:
logging correlator apply rule updown
  all-of-router
```

This example shows sample output from the **show logging events buffer all-in-buffer** command. The output displays the alarms stored in the logging events buffer after the one minute time period expires for the updown correlation rule configured:

```
RP/0/RSP0/CPU0:router# show logging events buffer all-in-buffer

#ID :C_id:Source :Time :%CATEGORY-GROUP-SEVERITY-MESSAGECODE: Text

#144 :46 :LC/0/7/CPU0:Jan 30 16:35:39 2004:ifmgr[130]: %PKT_INFRA-LINK-3-UPDOWN :
```

```
Interface POS0/7/0/0, changed state to Down
```

Note Only the first LINK UPDOWN root message is forwarded to the syslog process during the timeout interval.

The following example shows output from the **show logging correlator buffer correlationID** command generated after the one-minute interval expires. The output displays the alarms assigned correlation ID 46 in the logging correlator buffer. In the example, the PKT_INFRA-LINK-UPDOWN root-cause message and L2-SONET-ALARM leaf messages generated during the timeout interval assigned correlation ID 46 are displayed:

```
RP/0/RSP0/CPU0:router# show logging correlator buffer correlationID 46

#C_id.id:Rule Name:Source :Time : Text

#46.1 :updown :LC/0/7/CPU0:Jan 30 16:35:39 2004:ifmgr[130]: %PKT_INFRA-LINK-3-UPDOWN :

Interface POS0/7/0/0, changed state to Down

#46.2 :updown :LC/0/7/CPU0:Jan 30 16:35:41 2004:DI_Partner[50]: %L2-SONET-4-ALARM :

SONET0_7_0_0: SLOS
```

Note The subsequent PKT_INFRA-LINK-UPDOWN and L2-SONET-ALARM leaf messages generated during the timeout interval remain in the logging correlator buffer because they are leaf messages.

This example shows output from the **show logging correlator buffer correlationID** command. The output displays the alarms assigned to correlation IDs 46 and 47, the correlation IDs associated with the PKT_INFRA-LINK-UPDOWN and L2-SONET-ALARM root-cause messages:

```
RP/0/RSP0/CPU0:router# show logging correlator buffer correlationID 46

NO records matching query found
```

Additional References

The following sections provide references related to implementing and monitoring alarm logs and logging correlation on the Cisco ASR 9000 Series Router.

Related Documents

Related Topic	Document Title
Alarm and logging correlation commands	*Alarm Management and Logging Correlation Commands* module in the *System Monitoring Command Reference for Cisco ASR 9000 Series Routers*
Logging services commands	*Logging Services Commands* module in the *System Monitoring Command Reference for Cisco ASR 9000 Series Routers*

Related Topic	Document Title
Onboard Failure Logging (OBFL) configuration tasks	*Implementing Logging Services* module in the *System Monitoring Command Reference for Cisco ASR 9000 Series Routers*
Onboard Failure Logging (OBFL) commands	*Onboard Failure Logging Commands* module in the *System Monitoring Command Reference for Cisco ASR 9000 Series Routers*
Cisco IOS XR software XML API material	*Cisco IOS XR XML API Guide*
Cisco IOS XR software getting started material	*Cisco ASR 9000 Series Aggregation Services Router Getting Started Guide*
Information about user groups and task IDs	*Configuring AAA Services* module in the *System Security Configuration Guide for Cisco ASR 9000 Series Routers*

Standards

Standards	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	—

MIBs

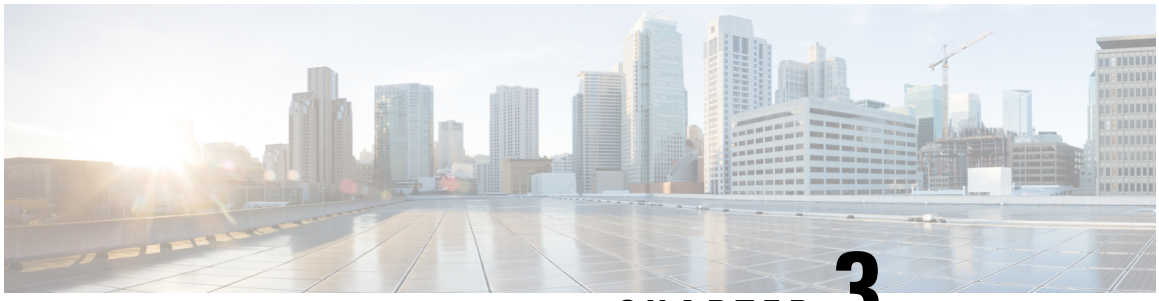
MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs	Title
No new or modified RFCs are supported by this feature, and support for existing RFCs has not been modified by this feature.	—

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	*http://www.cisco.com/cisco/web/support/index.html*



CHAPTER 3

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 new and revised tasks you need to configure and manage EEM policies on your the Cisco ASR 9000 Series Router and write and customize the EEM policies using Tool Command Language (Tcl) scripts to handle Cisco IOS XR Software faults and events.

Note For complete descriptions of the event management commands listed in this module, see the Related Documents, on page 95 section of this module.

Feature History for Configuring and Managing Embedded Event Manager Policies

Release	Modification
Release 4.0.0	This feature was introduced.

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 Management (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 can also be used to prevent or contain faults and to assist in fault recovery.

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 also maintains reliability metrics for each process in the system.

System Event Detection

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. It also relies on a timer event detector to detect that a certain time and date has occurred.

Policy-Based Event Response

When the EEM has detected an event, it can initiate actions in response. These actions are contained in routines called *policy handlers*. 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.

Reliability Metrics

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:
  - 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.
  - If a policy handler does not exist, the EEM does nothing.
- Notifies the processes that have *subscribed* for event notification.

Note A difference exists between scripts with policy actions and scripts that subscribe to receive events. Scripts with policy actions are expected to implement a policy. They are bound by a rule to prevent recursion. Scripts that subscribe to notifications are not bound by such a rule.

- 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 Management Policies

When the EEM has detected an event, it can initiate corrective actions. Actions are prescribed in routines called *policies*. Policies are defined by Tcl scripts (EEM scripts) written by the user through a Tcl API. (See the Embedded Event Manager Scripts and the Scripting Interface (Tcl), on page 45.) 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. You can disable a registered policy at any time.

Embedded Event Manager Scripts and the Scripting Interface (Tcl)

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

The EEM uses these two types of EEM scripts:

- *Regular* EEM scripts identified to the EEM through the **eem script** CLI command. Regular EEM scripts are standalone scripts that incorporate the definition of the event they will handle.

- *EEM callback* scripts identified to the EEM when a process or EEM script registers to handle an event. EEM callback scripts are essentially named functions that are identified to the EEM through the C Language API.

This example shows the usage for the CLI in scripts:

```
sjc-cde-010:/tftpboot/cnwei/fm> cat test_cli_eem.tcl
::cisco::eem::event_register_syslog occurs 1 pattern $_syslog_pattern maxrun 90

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

set errorInfo ""

# 1. query the information of latest triggered fm event
array set arr_einfo [event_reqinfo]

if {$_cerrno != 0} {
    set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
        $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}

set msg $arr_einfo(msg)
set config_cmds ""

# 2. execute the user-defined config commands
if [catch {cli_open} result] {
    error $result $errorInfo
} else {
    array set cli1 $result
}

if [catch {cli_exec $cli1(fd) "config"} result] {
    error $result $errorInfo
}

if {[info exists _config_cmd1]} {
    if [catch {cli_exec $cli1(fd) $_config_cmd1} result] {
        error $result $errorInfo
    }

    append config_cmds $_config_cmd1
}

if {[info exists _config_cmd2]} {
    if [catch {cli_exec $cli1(fd) $_config_cmd2} result] {
        error $result $errorInfo
    }
    append config_cmds "\n"
    append config_cmds $_config_cmd2
}

if [catch {cli_exec $cli1(fd) "end"} result] {
    error $result $errorInfo
}
if [catch {cli_close $cli1(fd) $cli1(tty_id)} result] {
    error $result $errorInfo
}

action_syslog priority info msg "Ran config command $_config_cmd1 $_config_cmd2
```

Script Language

The scripting language is Tool Command Language (Tcl) as implemented within the Cisco IOS XR Software. All Embedded Event Manager scripts are written in Tcl. This full Tcl implementation has been extended by Cisco, and an **eem** command has been added to provide the interface between Tcl scripts and the EEM.

Tcl is a string-based command language that is interpreted at run time. The version of Tcl supported is Tcl version 8.3.4, plus added script support. Scripts are defined using an ASCII editor on another device, not on the networking device. The script is then copied to the networking device and registered with EEM. Tcl scripts are supported by EEM. As an enforced rule, Embedded Event Manager policies are short-lived, run-time routines that must be interpreted and executed in less than 20 seconds of elapsed time. If more than 20 seconds of elapsed time are required, the maxrun parameter may be specified in the event_register statement to specify any desired value.

EEM policies use the full range of the Tcl language's capabilities. However, Cisco provides enhancements to the Tcl language in the form of Tcl command extensions that facilitate the writing of EEM policies. The main categories of Tcl command extensions identify the detected event, the subsequent action, utility information, counter values, and system information.

EEM allows you to write and implement your own policies using Tcl. Writing an EEM script involves:

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

Regular Embedded Event Manager Scripts

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

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

EEM scripts may use any of the EEM script services listed in Embedded Event Manager Policy Tcl Command Extension Categories, on page 48.

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

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

Embedded Event Manager Callback Scripts

EEM callback scripts are entered as a result of an EEM event being raised for a previously registered EEM event that specifies the name of this script in the eem_handler_spec.

When an EEM callback script exits, it is responsible for setting a return code that is used to tell the EEM whether or not to run the default action for this EEM event (if any). If multiple event handlers are scheduled for a given event, the return code from the previous handler is passed into the next handler, which can leave the value as is or update it.

Note EEM callback scripts are free to use any of the EEM script services listed in Table 2: Embedded Event Manager Tcl Command Extension Categories, on page 48, except for the **eem event register** keyword, which is not allowed in an EEM callback script.

Embedded Event Manager Policy Tcl Command Extension Categories

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

Note The Tcl command extensions available in each of these categories for use in all EEM policies are described in later sections in this document.

Table 2: Embedded Event Manager Tcl Command Extension Categories

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

Cisco File Naming Convention for Embedded Event Manager

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

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

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

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

Table 3: 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 4: 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.

Embedded Event Manager Built-In Action	Description
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 EXEC mode.
Publish an application-specific event	Publishes an application-specific event by using the **event_publish appl** Tcl command extension.
Reload the Cisco IOS software	Causes a router to be reloaded by using the EEM **action_reload** command.
Request system information	Represents the sys_reqinfo_xxx family of system-specific information commands by a policy to gather system information.
Send a short e-mail	Sends the e-mail out using Simple Mail Transfer Protocol (SMTP).
Set or modify a counter	Modifies a counter value.

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

Application-specific Embedded Event Management

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

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

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

Event Detection and Recovery

Events are detected by routines called *event detectors*. Event detectors are separate programs that provide an interface between other Cisco IOS XR Software components and the EEM. They process information that can be used to publish events, if necessary.

These event detectors are supported:

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.

General Flow of EEM 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. The relationship is between the EEM server, the core event publishers (event detectors), and the event subscribers (policies). Event publishers screen events and publish them when there is a match on an event specification that is provided by the event subscriber. Event detectors notify the EEM server when an event of interest occurs.

When an event or fault is detected, Embedded Event Manager determines from the event publishers—an example would be the OIR events publisher —if a registration for the encountered fault or event has occurred. EEM matches the event registration information with the event data itself. A policy registers for the detected event with the Tcl command extension event_register_xxx. The event information Tcl command extension event_reqinfo is used in the policy to query the Embedded Event Manager for information about the detected event.

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. POSIX wildcard filename pattern support using *, ?, or [...] is provided for load module pathnames. 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 5: 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. POSIX regular expression pattern support is provided for the message text field.

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

Table 6: Syslog Event Detector Event Monitoring Requests

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

None Event Detector

The None Event Detector publishes an event when the Cisco IOS 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 7: 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.
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 entire processor complex is maintained by the EEM. The data is periodically written to checkpoint.

Hardware Card Reliability Metric Data

Reliability metric data is kept for each hardware card in a processor complex. 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 QNX—Neutrino 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.

Environment Variables

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

Spaces may be used in the *var-value* argument of the **event manager environment** command. The command interprets everything after the *var-name* argument to the end of the line to be part of the *var-value* argument.

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

SUMMARY STEPS

1. **show event manager environment**
2. **configure**
3. **event manager environment** *var-name var-value*
4. Repeat Step 3 for every environment value to be reset.
5. Use the **commit** or **end** command.

6. **show event manager environment**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**show event manager environment** **Example:** `RP/0/RSP0/CPU0:router# show event manager environment`	Displays the names and values of all EEM environment variables.
Step 2	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 3	**event manager environment** *var-name var-value* **Example:** `RP/0/RSP0/CPU0:router(config)# event manager environment _cron_entry 0-59/2 0-23/1 * * 0-7`	Resets environment variables to new values. • The *var-name* argument is the name assigned to the EEM environment configuration variable. • The *var-value* argument is the series of characters, including embedded spaces, to be placed in the environment variable *var-name*. • By convention, the names of all environment variables defined by Cisco begin with an underscore character to set them apart; for example, _show_cmd. • Spaces may be used in the *var-value* argument. The command interprets everything after the *var-name* argument to the end of the line to be part of the *var-value* argument.
Step 4	Repeat Step 3 for every environment value to be reset.	—
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.

	Command or Action	Purpose
Step 6	**show event manager environment** **Example:** `RP/0/RSP0/CPU0:router# show event manager environment`	Displays the reset names and values of all EEM environment variables; allows you to verify the environment variable names and values set in Step 3.

What to do next

After setting up EEM environment variables, find out what policies are available to be registered and then register those policies, as described in the Registering Embedded Event Manager Policies, on page 58.

Registering Embedded Event Manager Policies

Register an EEM policy to run a policy when an event is triggered.

Embedded Event Manager Policies

Registering an EEM policy is performed with the **event manager policy** command in global configuration mode. 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.

Username

To register an EEM policy, you must specify the username that is used to run the script. This name can be different from the user who is currently logged in, but the registering user must have permissions that are a superset of the username that will run the script. Otherwise, the script is not registered and the command is rejected. In addition, the username that will run the script must have access privileges to the commands run by the EEM policy being registered.

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.

Persist-time

An optional **persist-time** keyword for the username can also be defined. The **persist-time** keyword defines the number of seconds the username authentication is valid. When a script is first registered, the configured username for the script is authenticated. After the script is registered, the username is authenticated again each time a script is run. If the AAA server is down, the username authentication can be read from memory. The **persist-time** keyword determines the number of seconds this username authentication is held in memory.

- If the AAA server is down and the **persist-time** keyword has not expired, then the username is authenticated from memory and the script runs.

- If the AAA server is down, and the **persist-time** keyword has expired, then user authentication will fail and the script will not run.

The following values can be used for the **persist-time** keyword.

- The default **persist-time** is 3600 seconds (1 hour). Enter the **event manager policy** command without the **persist-time** keyword to set the **persist-time** to 1 hour.
- Enter 0 to stop the username authentication from being cached. If the AAA server is down, the username will not authenticate and the script will not run.
- Enter **infinite** to stop the username from being marked as invalid. The username authentication held in the cache will not expire. If the AAA server is down, the username will be authenticated from the cache.

System or user keywords

If you enter the **event manager policy** command without specifying either the **system** or **user** keyword, the EEM first tries to locate the specified policy file in the system policy directory. If the EEM finds the file in the system policy directory, it registers the policy as a system policy. If the EEM does not find the specified policy file in the system policy directory, it looks in the user policy directory. If the EEM locates the specified file in the user policy directory, it registers the policy file as a user policy. If the EEM finds policy files with the same name in both the system policy directory and the user policy directory, the policy file in the system policy directory takes precedence and is registered as a system policy.

Once policies have been registered, their registration can be verified through the **show event manager policy registered** command. The output displays registered policy information in two parts. The first line in each policy description lists the index number assigned to the policy, the policy type (system or user), the type of event registered, the time when the policy was registered, and the name of the policy file. The remaining lines of each policy description display information about the registered event and how the event is to be handled, and come directly from the Tcl command arguments that make up the policy file.

SUMMARY STEPS

1. **show event manager policy available** [**system** | **user**]
2. **configure**
3. **event manager policy** *policy-name* **username** *username* [**persist-time** { *seconds* | **infinite** }] | **type** { **system** | **user** }
4. Repeat Step 3 for every EEM policy to be registered.
5. Use the **commit** or **end** command.
6. **show event manager policy registered**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**show event manager policy available** [**system** | **user**] **Example:** `RP/0/RSP0/CPU0:router# show event manager policy available`	Displays all EEM policies that are available to be registered. • Entering the optional **system** keyword displays all available system policies. • Entering the optional **user** keyword displays all available user policies.

	Command or Action	Purpose
Step 2	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.

Step 3	**event manager policy** *policy-name* **username** *username* [**persist-time** { *seconds* | **infinite** }] | **type** { **system** | **user** }
	Example:
	`RP/0/RSP0/CPU0:router(config)# event manager policy cron.tcl username tom type user`
		Registers an EEM policy with the EEM.
		• An EEM script is available to be scheduled by the EEM until the **no** form of this command is entered.
		• Enter the required **username** keyword and argument, where *username* is the username that runs the script.
		• Enter the optional **persist-time** keyword to determine how long the username authentication is held in memory:
		  • Enter the number of *seconds* for the **persist-time** keyword.
		  • Enter the **infinite** keyword to make the authentication permanent (the authentication will not expire).
		• Entering the optional **type system** keywords registers a system policy defined by Cisco.
		• Entering the optional **type user** keywords registers a user-defined policy.
		Note AAA authorization (such as aaa authorization eventmanager) must be configured before EEM policies can be registered. See the *Configuring AAA Services* module of *System Security Configuration Guide for Cisco ASR 9000 Series Routers* for more information about AAA authorization configuration.

Step 4	Repeat Step 3 for every EEM policy to be registered.	—
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.

	Command or Action	Purpose
Step 6	**show event manager policy registered** **Example:** `RP/0/RSP0/CPU0:router# show event manager policy registered`	Displays all EEM policies that are already registered, allowing verification of Step 3.

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.

Before you begin

A policy must be available that is written in the Tcl scripting language. Sample policies are provided in the Sample EEM Policies, on page 67. Sample policies are stored in the system policy directory.

SUMMARY STEPS

1. **show event manager environment** [**all** | *environment-name*]
2. **configure**
3. **event manager environment** *var-name* [*var-value*]
4. Repeat Step 3, on page 62 to configure all the environment variables required by the policy to be registered in Step 5, on page 62.
5. **event manager policy** *policy-name* **username** *username* [**persist-time** [*seconds* | **infinite**] | **type** [**system** | **user**]]
6. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**show event manager environment** [**all**	*environment-name*] **Example:** `RP/0/RSP0/CPU0:router# show event manager environment all`	(Optional) Displays the name and value of EEM environment variables. • The **all** keyword displays all the EEM environment variables. • The *environment-name* argument displays information about the specified environment variable.

	Command or Action	Purpose
Step 2	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 3	**event manager environment** *var-name* [*var-value*] **Example:** `RP/0/RSP0/CPU0:router(config)# event manager environment _cron_entry 0-59/2 0-23/1 * * 0-7`	Resets environment variables to new values. • The *var-name* argument is the name assigned to the EEM environment configuration variable. • The *var-value* argument is the series of characters, including embedded spaces, to be placed in the environment variable *var-name* . • By convention, the names of all environment variables defined by Cisco begin with an underscore character to set them apart; for example, _show_cmd. • Spaces may be used in the *var-value* argument. The command interprets everything after the *var-name* argument to the end of the line to be part of the *var-value* argument.
Step 4	Repeat Step 3, on page 62 to configure all the environment variables required by the policy to be registered in Step 5, on page 62.	—
Step 5	**event manager policy** *policy-name* **username** *username* [**persist-time** [*seconds* | **infinite**] | **type** [**system** | **user**]] **Example:** `RP/0/RSP0/CPU0:router(config)# event manager policy tm_cli_cmd.tcl username user_a type system`	Registers the EEM policy to be run when the specified event defined within the policy occurs. • Use the **system** keyword to register a system policy defined by Cisco. • Use the **user** keyword to register a user-defined system policy. • Use the **persist-time** keyword to specify the length of time the username authentication is valid. In this example, the sample EEM policy named tm_cli_cmd.tcl is registered as a system policy.
Step 6	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.

	Command or Action	Purpose
		• **Cancel** —Remains in the configuration session, without committing the configuration changes.

Displaying EEM Registered Policies

Perform this optional task to display EEM registered policies.

SUMMARY STEPS

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**show event manager policy registered** [**event-type** *type*] [**system**	**user**] [**time-ordered**	**name-ordered**] **Example:** `RP/0/RSP0/CPU0:router# show event manager policy registered system`	Displays information about currently registered policies. • The **event-type** keyword displays the registered policies for a specific event type. • The **time-ordered** keyword displays information about currently registered policies sorted by time. • The **name-ordered** keyword displays the policies in alphabetical order by the policy name.

Unregistering EEM Policies

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

SUMMARY STEPS

1. **show event manager policy registered** [**event-type** *type*] [**system** | **user**] [**time-ordered** | **name-ordered**]
2. **configure**
3. **no event manager policy** *policy-name*
4. Use the **commit** or **end** command.
5. Repeat Step 1, on page 63to ensure that the policy has been removed.

DETAILED STEPS

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

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router# show event manager policy registered system`	• The **time-ordered** keyword displays information about currently registered policies sorted by time. • The **name-ordered** keyword displays the policies in alphabetical order by the policy name.
Step 2	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 3	**no event manager policy** *policy-name* **Example:** `RP/0/RSP0/CPU0:router(config)# no event manager policy tm_cli_cmd.tcl`	Removes the EEM policy from the configuration, causing the policy to be unregistered.
Step 4	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 5	Repeat Step 1, on page 63to ensure that the policy has been removed.	—

Suspending EEM Policy Execution

Perform this task to immediately suspend the execution of all EEM policies. Suspending policies, instead of unregistering them, might be necessary for reasons of temporary performance or security.

SUMMARY STEPS

1. **show event manager policy registered** [**event-type** *type*] [**system** | **user**] [**time-ordered** | **name-ordered**]
2. **configure**
3. **event manager scheduler suspend**
4. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**show event manager policy registered** [**event-type** *type*] [**system** | **user**] [**time-ordered** | **name-ordered**] **Example:** `RP/0/RSP0/CPU0:router# show event manager policy registered system`	Displays information about currently registered policies. • The **event-type** keyword displays the registered policies for a specific event type. • The **time-ordered** keyword displays information about currently registered policies sorted by time. • The **name-ordered** keyword displays the policies in alphabetical order by the policy name.
Step 2	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 3	**event manager scheduler suspend** **Example:** `RP/0/RSP0/CPU0:router(config)# event manager scheduler suspend`	Immediately suspends the execution of all EEM policies.
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.

Managing EEM Policies

Perform this task to specify a directory to use for storing user library files or user-defined EEM policies.

Note This task applies only to EEM policies that are written using Tcl scripts.

SUMMARY STEPS

1. **show event manager directory user** [**library** | **policy**]
2. **configure**
3. **event manager directory user** {**library** *path* | **policy** *path*}

4. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**show event manager directory user** [**library** | **policy**] **Example:** `RP/0/RSP0/CPU0:router# show event manager directory user library`	Displays the directory to use for storing EEM user library or policy files. • The optional **library** keyword displays the directory to use for user library files. • The optional **policy** keyword displays the directory to use for user-defined EEM policies.
Step 2	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 3	**event manager directory user** {**library** *path* | **policy** *path*} **Example:** `RP/0/RSP0/CPU0:router(config)# event manager directory user library disk0:/usr/lib/tcl`	Specifies a directory to use for storing user library files or user-defined EEM policies. • Use the *path* argument to specify the absolute pathname to the user directory.
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.

Displaying Software Modularity Process Reliability Metrics Using EEM

Perform this optional task to display reliability metrics for Cisco IOS XR Software processes.

SUMMARY STEPS

1. **show event manager metric process** {**all** | *job-id* | *process-name*} **location** {**all** | *node-id*}

DETAILED STEPS

	Command or Action	Purpose
Step 1	**show event manager metric process** {**all** | *job-id* | *process-name*} **location** {**all** | *node-id*} **Example:** `RP/0/RSP0/CPU0:router# show event manager environment`	Displays the reliability metric data for processes. The system keeps a record of when processes start and end, and this data is used as the basis for reliability analysis.

Sample EEM Policies

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

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

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

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

This table describes the sample EEM policies.

Table 8: Sample EEM Policy Descriptions

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

Name of Policy	Description
tm_crash_hist.tcl	This policy runs at midnight each day and e-mails a process crash history report to a specified e-mail address.

For more details about the sample policies available and how to run them, see the EEM Event Detector Demo: Example , on page 85.

SUMMARY STEPS

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	**show event manager policy available** [**system**	**user**] **Example:** `RP/0/RSP0/CPU0:router# show event manager policy available`	Displays EEM policies that are available to be registered.
Step 2	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 3	**event manager directory user** {**library** *path*	**policy** *path*} **Example:** `RP/0/RSP0/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 4	**event manager policy** *policy-name* **username** *username* [**persist-time** [*seconds*	**infinite**]	**type** [**system**	**user**]] **Example:** `RP/0/RSP0/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 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.

	Command or Action	Purpose
		• **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

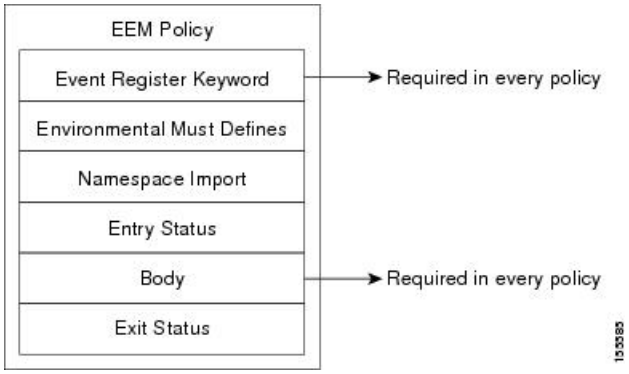
Programming EEM Policies with Tcl

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. All other sections shown in the Tcl Policy Structure and Requirements, on page 69 are optional.

Tcl Policy Structure and Requirements

All EEM policies share the same structure, shown in Figure 2: Tcl Policy Structure and Requirements , on page 69. There are two parts of an EEM policy that are required: the event_register Tcl command extension and the body. The remaining parts of the policy are optional: environmental must defines, namespace import, entry status, and exit status.

Figure 2: Tcl Policy Structure and Requirements



The start of every policy must describe and register the event to detect using an **event_register** Tcl command extension. This part of the policy schedules the running of the policy. For a list of the available EEM **event_register** Tcl command extensions, see the Embedded Event Manager Event Registration Tcl Command Extensions, on page 96. 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. For a list of the available EEM event information Tcl command extensions, see the Embedded Event Manager Event Information Tcl Command Extension, on page 120.
- The action Tcl command extensions, such as **action_syslog**, that are used to specify actions specific to EEM. For a list of the available EEM action Tcl command extensions, see the Embedded Event Manager Action Tcl Command Extensions, on page 139.
- The system information Tcl command extensions, such as **sys_reqinfo_routername**, that are used to obtain general system information. For a list of the available EEM system information Tcl command extensions, see the Embedded Event Manager System Information Tcl Command Extensions, on page 156.
- Use of the SMTP library (to send e-mail notifications) or the CLI library (to run CLI commands) from a policy. For a list of the available SMTP library Tcl command extensions, see the SMTP Library Command Extensions, on page 166. For a list of the available CLI library Tcl command extensions, see the CLI Library Command Extensions, on page 168.
- The **context_save** and **con text_retrieve** Tcl command extensions that are used to save Tcl variables for use by other policies.

The following example Tcl code shows the code to query an event and to log a message as part of the body section:

```
# Query the event info and log a message.
array set arr_einfo [event_reqinfo]
if {$_cerrno != 0} {
  set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
    $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
  error $result
}
global timer_type timer_time_sec
set timer_type $arr_einfo(timer_type)
  set timer_time_sec $arr_einfo(timer_time_sec)
# Log a message.
set msg [format "timer event: timer type %s, time expired %s" \
$timer_type [clock format $timer_time_sec]]
action_syslog priority info msg $msg
if {$_cerrno != 0} {
  set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
    $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
```

```
  error $result
}
```

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 _cerrno. Whenever _cerrno is set, the other Tcl global variables are derived from _cerrno and are set along with it (_cerr_sub_num, _cerr_sub_err, _cerr_posix_err, and _cerr_str).

For example, the **action_syslog** command in the following example sets these global variables as a side effect of the command execution:

```
action_syslog priority warning msg "A sample message generated by action_syslog"
if {$_cerrno != 0} {
  set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
    $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
  error $result
}
```

_cerrno: 32-Bit Error Return Values

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

```
XYSSSSSSSSSSSSSEEEEEEEEPPPPPPPPP
```

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
```

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

Table 9: _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 10: Error Class Encodings, on page 72 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.
PPPPPPPP	The pass-through POSIX error code (9 bits = 512 values). This represents the last of the 32-bit sequence, and the string corresponding to this error code is contained in $_cerr_posix_err.

Error Class Encodings for XY

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

Table 10: 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.

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/RSP0/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 • 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

	Command or Action	Purpose
		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 "_". **Note** You can display the Embedded Event Manager environment variables set on your system by using the **show event manager environment** command in EXEC mode. 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

	Command or Action	Purpose
		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).
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

	Command or Action	Purpose
		is set, four other Tcl global variables are derived from _cerrno and are set along with it (_cerr_sub_num, _cerr_sub_err, _cerr_posix_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 3: Two-Character Abbreviation Specification, on page 49) 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 48. Copy the file to the flash file system on the router—typically disk0:.
Step 11	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 12	**event manager directory user** {**library** *path* | **policy** *path*} **Example:** `RP/0/RSP0/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/RSP0/CPU0:router(config)# event manager policy test.tcl username user_a type user`	Registers the EEM policy to be run when the specified event defined within the policy occurs.
Step 14	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session.

	Command or Action	Purpose
		• **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
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, on page 77and the tclIndex file from Step 3, on page 78to 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
		`proc test1 {} {`
		`  puts "In procedure test1"`
		`}`
		`proc test2 {} {`

	Command or Action	Purpose

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

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

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

Step 4	Copy the Tcl library files from Step 1, on page 77and the tclIndex file from Step 3, on page 78to 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, on page 78.

	Command or Action	Purpose

		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/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 7	**event manager directory user library** *path* **Example:** `RP/0/RSP0/CPU0:router(config)# event manager directory user library disk2:/eem_library`	Specifies the EEM user library directory; this is the directory to which the files in Step 4, on page 78 were copied.
Step 8	**event manager directory user policy** *path* **Example:** `RP/0/RSP0/CPU0:router(config)# event manager directory user policy disk2:/eem_policies`	Specifies the EEM user policy directory; this is the directory to which the file in Step 5, on page 78was copied.
Step 9	**event manager policy** *policy-name* **username** *username* [**persist-time** [*seconds* | **infinite**] | **type** [**system** | **user**]] **Example:** `RP/0/RSP0/CPU0:router(config)# event manager policy libtest.tcl username user_a`	Registers a user-defined EEM policy.
Step 10	**event manager run** *policy* [*argument*] **Example:** `RP/0/RSP0/CPU0:router(config)# event manager run libtest.tcl`	Manually runs an EEM policy.

	Command or Action	Purpose
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.`
		`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.	—

	Command or Action	Purpose

Step 5	Copy a user-defined EEM policy file written in Tcl to the directory used for storing user-defined EEM policies on the target router.	The directory can be the same directory used in Step 4, on page 81.
		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/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 7	**event manager directory user library** *path* **Example:** `RP/0/RSP0/CPU0:router(config)# event manager directory user library disk2:/eem_library`	Specifies the EEM user library directory; this is the directory to which the files in Step 4, on page 81were copied.
Step 8	**event manager directory user policy** *path* **Example:** `RP/0/RSP0/CPU0:router(config)# event manager directory user policy disk2:/eem_policies`	Specifies the EEM user policy directory; this is the directory to which the file in Step 5, on page 82was copied.
Step 9	**event manager policy** *policy-name* **username** *username* [**persist-time** [*seconds* | **infinite**] | **type** [**system** | **user**]] **Example:** `RP/0/RSP0/CPU0:router(config)# event manager policy packagetest.tcl username user_a`	Registers a user-defined EEM policy.
Step 10	**event manager run** *policy* [*argument*] **Example:** `RP/0/RSP0/CPU0:router(config)# event manager run packagetest.tcl`	Manually runs an EEM policy.

	Command or Action	Purpose
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.

Configuration Examples for Event Management Policies

Environmental Variables Configuration: Example

This configuration sets the environment variable cron_entry:

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router#(config)# event manager environment _cron_entry 0-59/2 0-23/1 * * 0-7
```

User-Defined Embedded Event Manager Policy Registration: Example

This configuration registers a user-defined event management policy:

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# event manager policy cron.tcl username tom user
```

Display Available Policies: Example

This is the sample output from the **show event manager policy available** command displaying available policies:

```
RP/0/RSP0/CPU0:router# show event manager policy available

No.  Type    Time Created                 Name
1    system  Mon Mar 15 21:32:14 2004     periodic_diag_cmds.tcl
2    system  Mon Mar 15 21:32:14 2004     periodic_proc_avail.tcl
3    system  Mon Mar 15 21:32:16 2004     periodic_sh_log.tcl
4    system  Mon Mar 15 21:32:16 2004     tm_cli_cmd.tcl
5    system  Mon Mar 15 21:32:16 2004     tm_crash_hist.tcl
```

Display Embedded Event Manager Process: Example

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. This is the sample output from the **show event manager metric process** command displaying reliability metric data:

```
RP/0/RSP0/CPU0:router# show event manager metric process all location 0/1/CPU0

====================================
job id: 78, node name: 0/1/CPU0
process name: wd-critical-mon, instance: 1
--------------------------------
last event type: process start
recent start time: Mon Sep 10 21:36:49 2007
recent normal end time: n/a
recent abnormal end time: n/a
number of times started: 1
number of times ended normally: 0
number of times ended abnormally: 0
most recent 10 process start times:
--------------------------
Mon Sep 10 21:36:49 2007
--------------------------

most recent 10 process end times and types:

cumulative process available time: 59 hours 33 minutes 42 seconds 638 milliseconds
cumulative process unavailable time: 0 hours 0 minutes 0 seconds 0 milliseconds
process availability:  1.000000000
number of abnormal ends within the past 60 minutes (since reload): 0
number of abnormal ends within the past 24 hours (since reload): 0
number of abnormal ends within the past 30 days (since reload): 0
====================================
job id: 56, node name: 0/1/CPU0
process name: dllmgr, instance: 1
--------------------------------
last event type: process start
recent start time: Mon Sep 10 21:36:49 2007
recent normal end time: n/a
recent abnormal end time: n/a
number of times started: 1
number of times ended normally: 0
number of times ended abnormally: 0
most recent 10 process start times:
--------------------------
Mon Sep 10 21:36:49 2007
--------------------------

most recent 10 process end times and types:

cumulative process available time: 59 hours 33 minutes 42 seconds 633 milliseconds
cumulative process unavailable time: 0 hours 0 minutes 0 seconds 0 milliseconds
process availability:  1.000000000
number of abnormal ends within the past 60 minutes (since reload): 0
number of abnormal ends within the past 24 hours (since reload): 0
number of abnormal ends within the past 30 days (since reload): 0
====================================
```

Configuration Examples for Writing Embedded Event Manager Policies Using Tcl

EEM Event Detector Demo: Example

This example uses the sample policies to demonstrate how to use Embedded Event Manager policies. Proceed through the following sections to see how to use the sample policies:

EEM Sample Policy Descriptions

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

Event Manager Environment Variables for the Sample Policies

Event manager environment variables are Tcl global variables that are defined external to the EEM policy before the policy is registered and run. The sample policies require three of the e-mail environment variables to be set; only _email_cc is optional. Other required and optional variable settings are outlined in the following tables.

This table describes a list of the e-mail variables.

Table 11: E-mail-Specific Environmental Variables Used by the Sample Policies

Environment Variable	Description	Example
_domainname	The default domain name.	example.com
_email_server	Simple Mail Transfer Protocol (SMTP) mail server used to send e-mail.	mailserver.example.com
_email_to	Address to which e-mail is sent.	engineering@example.com
_email_from	Address from which e-mail is sent.	devtest@example.com
_email_cc	Address to which the e-mail must be copied.	manager@example.com

This table describes the EEM environment variables that must be set before the sl_intf_down.tcl sample policy is run.

Table 12: Environment Variables Used in the sl_intf_down.tcl Policy

Environment Variable	Description	Example
_config_cmd1	First configuration command that is run.	**interface gigabitEthernet1/0/5/0**

Environment Variable	Description	Example
_config_cmd2	Second configuration command that is run. This variable is optional and need not be specified.	**no shutdown**
_syslog_pattern	Regular expression pattern match string that is used to compare syslog messages to determine when the policy runs.	.*UPDOWN.*FastEthernet0/0.*

This table describes the EEM environment variables that must be set before the tm_cli_cmd.tcl sample policy is run.

Table 13: Environment Variables Used in the tm_cli_cmd.tcl Policy

Environment Variable	Description	Example
_cron_entry	CRON specification that determines when the policy will run.	0-59/1 0-23/1 * * 0-7
_show_cmd	CLI command to be executed when the policy is run.	**show version**

This table describes the EEM environment variables that must be set before the tm_crash_reporter.tcl sample policy is run.

Table 14: Environment Variables Used in the tm_crash_reporter.tcl Policy

Environment Variable	Description	Example
_crash_reporter_debug	Value that identifies whether debug information for tm_crash_reporter.tcl will be enabled. This variable is optional and need not be specified.	1
_crash_reporter_url	URL location to which the crash report is sent.	http://www.example.com/fm/interface_tm.cgi

This table describes the EEM environment variables that must be set before the tm_fsys_usage.tcl sample policy is run.

Table 15: Environment Variables Used in the tm_fsys_usage.tcl Policy

Environment Variable	Description	Example
_tm_fsys_usage_cron	CRON specification that is used in the event_register Tcl command extension. If unspecified, the tm_fsys_usage.tcl policy is triggered once per minute. This variable is optional and need not be specified.	0-59/1 0-23/1 * * 0-7
_tm_fsys_usage_debug	When this variable is set to a value of 1, disk usage information is displayed for all entries in the system. This variable is optional and need not be specified.	1

Environment Variable	Description	Example
_tm_fsys_usage_freebytes	Free byte threshold for systems or specific prefixes. If free space falls below a given value, a warning is displayed. This variable is optional and need not be specified.	disk2:98000000
_tm_fsys_usage_percent	Disk usage percentage thresholds for systems or specific prefixes. If the disk usage percentage exceeds a given percentage, a warning is displayed. If unspecified, the default disk usage percentage is 80 percent for all systems. This variable is optional and need not be specified.	nvram:25 disk2:5

Registration of Some EEM Policies

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

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

Basic Configuration Details for All Sample Policies

To allow e-mail to be sent from the Embedded Event Manager (EEM), the **hostname** and **domain-name** commands must be configured. The EEM environment variables must also be set. After a Cisco IOS XR Software image has been booted, use the following initial configuration, substituting appropriate values for your network. The environment variables for the tm_fsys_usage sample policy (see Table 15: Environment Variables Used in the tm_fsys_usage.tcl Policy, on page 86) are all optional and are not listed here:

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

Using the Sample Policies

This section contains these configuration scenarios to demonstrate how to use the four sample Tcl policies:

Running the sl_intf_down.tcl Sample Policy

This sample policy demonstrates the ability to modify the configuration when a syslog message with a specific pattern is logged. The policy gathers detailed information about the event and uses the CLI library to run the configuration commands specified in the EEM environment variables _config_cmd1 and, optionally, _config_cmd2. An e-mail message is sent with the results of the CLI command.

The following sample configuration demonstrates how to use this policy. Starting in EXEC mode, use the **show event manager policy registered** command to verify that no policies are currently registered. The next command is the **show event manager policy available** command, which displays policies that are available to be installed. After you enter the **configure** command to reach global configuration mode, you can register the sl_intf_down.tcl policy with EEM using the **event manager policy** command. Exit from global configuration mode and enter the **show event manager policy registered** command again, to verify that the policy has been registered.

The policy runs when an interface goes down. Enter the **show event manager environment** command to display the current environment variable values. Unplug the cable (or configure a shutdown) for the interface specified in the _syslog_pattern EEM environment variable. The interface goes down, prompting the syslog daemon to log a syslog message about the interface being down, and the syslog event detector is called.

The syslog event detector reviews the outstanding event specifications and finds a match for interface status change. The EEM server is notified, and the server runs the policy that is registered to handle this event—sl_intf_down.tcl.

```
enable
show event manager policy registered
show event manager policy available
configure terminal
 event manager policy sl_intf_down.tcl
 end
show event manager policy registered
show event manager environment
```

Running the tm_cli_cmd.tcl Sample Policy

This sample policy demonstrates the ability to periodically run a CLI command and to e-mail the results. The CRON specification "0-59/2 0-23/1 * * 0-7" causes this policy to be run on the second minute of each hour. The policy gathers detailed information about the event and uses the CLI library to execute the configuration commands specified in the EEM environment variable _show_cmd. An e-mail message is sent with the results of the CLI command.

The following sample configuration demonstrates how to use this policy. Starting in EXEC mode, enter the **show event manager policy registered** command to verify that no policies are currently registered. The next command is the **show event manager policy available** command, which displays the policies that are available to be installed. After you enter the **configure** command to reach global configuration mode, you can register the tm_cli_cmd.tcl policy with EEM using the **event manager policy** command. Exit from global configuration mode and enter the **show event manager policy registered** command to verify that the policy has been registered.

The timer event detector triggers an event for this case periodically, according to the CRON string set in the EEM environment variable _cron_entry. The EEM server is notified, and the server runs the policy that is registered to handle this event—tm_cli_cmd.tcl.

```
enable
show event manager policy registered
show event manager policy available
configure terminal
```

```
 event manager policy tm_cli_cmd.tcl
 end
show event manager policy registered
```

Running the tm_crash_reporter.tcl Sample Policy

This sample policy demonstrates the ability to send an HTTP-formatted crash report to a URL location. If the policy registration is saved in the startup configuration file, the policy is triggered 5 seconds after bootup. When triggered, the script attempts to find the reload reason. If the reload reason was due to a crash, the policy searches for the related crashinfo file and sends this information to a URL location specified by the user in the environment variable _crash_reporter_url. A CGI script, interface_tm.cgi, has been created to receive the URL from the tm_crash_reporter.tcl policy and save the crash information in a local database on the target URL machine.

A Perl CGI script, interface_tm.cgi, has been created and is designed to run on a machine that contains an HTTP server and is accessible by the router that runs the tm_crash_reporter.tcl policy. The interface_tm.cgi script parses the data passed into it from tm_crash_reporter.tcl and appends the crash information to a text file, creating a history of all crashes in the system. Additionally, detailed information on each crash is stored in three files in a crash database directory that is specified by the user. Another Perl CGI script, crash_report_display.cgi, has been created to display the information stored in the database created by the interface_tm.cgi script. The crash_report_display.cgi script should be placed on the same machine that contains interface_tm.cgi. The machine should be running a web browser such as Internet Explorer or Netscape. When the crash_report_display.cgi script is run, it displays the crash information in a readable format.

The following sample configuration demonstrates how to use this policy. Starting in EXEC mode, enter the **show event manager policy registered** command to verify that no policies are currently registered. Next, enter the **show event manager policy available** command to display which policies are available to be installed. After you enter the **configure** command to reach global configuration mode, you can register the tm_crash_reporter.tcl policy with EEM using the **event manager policy** command. Exit from global configuration mode and enter the **show event manager policy registered** command to verify that the policy has been registered.

```
enable
show event manager policy registered
show event manager policy available
configure terminal
 event manager policy tm_crash_reporter.tcl
 end
show event manager policy registered
```

Running the tm_fsys_usage.tcl Sample Policy

This sample policy demonstrates the ability to periodically monitor disk space usage and report through syslog when configurable thresholds have been crossed.

The following sample configuration demonstrates how to use this policy. Starting in user EXEC mode, enter the **show event manager policy registered** command to verify that no policies are currently registered. Next, enter the **show event manager policy available** command to display which policies are available to be installed. After you enter the **configure** command to reach global configuration mode, you can register the tm_fsys_usage.tcl policy with EEM using the **event manager policy** command. Exit from global configuration mode and enter the **show event manager policy registered** command again to verify that the policy has been registered. If you had configured any of the optional environment variables that are used in the tm_fsys_usage.tcl policy, the **show event manager environment** command displays the configured variables.

```
enable
show event manager policy registered
```

```
show event manager policy available
configure terminal
 event manager policy tm_fsys_usage.tcl
 end
show event manager policy registered
show event manager environment
```

Programming Policies with Tcl: Sample Scripts Example

This section contains two of the sample policies that are included as EEM system policies. For more details about these policies, see the EEM Event Detector Demo: Example , on page 85.

tm_cli_cmd.tcl Sample Policy

The following sample policy runs a configurable CRON entry. The policy executes a configurable Cisco IOS XR SoftwareCLI command and e-mails the results. An optional log file can be defined to which the output is appended with a time stamp.

```
::cisco::eem::event_register_timer cron name crontimer2 cron_entry $_cron_entry maxrun 240
#-------------------------------------------------------------------
# EEM policy that will periodically execute a cli command and email the
# results to a user.
#
# July 2005, Cisco EEM team
#
# Copyright (c) 2005 by cisco Systems, Inc.
# All rights reserved.
#-------------------------------------------------------------------
### The following EEM environment variables are used:
###
### _cron_entry (mandatory)            - A CRON specification that determines
###                                      when the policy will run. See the
###                                      IOS XR Embedded Event Manager
###                                      documentation for more information
###                                      on how to specify a cron entry.
### Example: _cron_entry                 0-59/1 0-23/1 * * 0-7
###
### _log_file (mandatory without _email_....)
###                                    - A filename to append the output to.
###                                      If this variable is defined, the
###                                      output is appended to the specified
###                                      file with a timestamp added.
### Example: _log_file                   disk0:/my_file.log
###
### _email_server (mandatory without _log_file)
###                                    - A Simple Mail Transfer Protocol (SMTP)
###                                      mail server used to send e-mail.
### Example: _email_server               mailserver.example.com
###
### _email_from (mandatory without _log_file)
###                                    - The address from which e-mail is sent.
### Example: _email_from                 devtest@example.com
###
### _email_to (mandatory without _log_file)
###                                    - The address to which e-mail is sent.
### Example: _email_to                   engineering@example.com
###
### _email_cc (optional)               - The address to which the e-mail must
###                                      be copied.
### Example: _email_cc                   manager@example.com
```

```
###
### _show_cmd (mandatory)              - The CLI command to be executed when
###                                      the policy is run.
### Example: _show_cmd                   show version
###
# check if all required environment variables exist
# If any required environment variable does not exist, print out an error msg and quit
if {![info exists _log_file]} {
    if {![info exists _email_server]} {
set result \
"Policy cannot be run: variable _log_file or _email_server has not been set"
error $result $errorInfo
    }
    if {![info exists _email_from]} {
set result \
"Policy cannot be run: variable _log_file or _email_from has not been set"
error $result $errorInfo
    }
    if {![info exists _email_to]} {
set result \
"Policy cannot be run: variable _log_file ore _email_to has not been set"
error $result $errorInfo
    }
    if {![info exists _email_cc]} {
#_email_cc is an option, must set to empty string if not set.
set _email_cc ""
    }
}
if {![info exists _show_cmd]} {
    set result \
        "Policy cannot be run: variable _show_cmd has not been set"
    error $result $errorInfo
}
namespace import ::cisco::eem::*
namespace import ::cisco::lib::*
# query the event info and log a message
array set arr_einfo [event_reqinfo]
if {$_cerrno != 0} {
    set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
        $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}
global timer_type timer_time_sec
set timer_type $arr_einfo(timer_type)
set timer_time_sec $arr_einfo(timer_time_sec)
# log a message
set msg [format "timer event: timer type %s, time expired %s" \
        $timer_type [clock format $timer_time_sec]]
action_syslog priority info msg $msg
if {$_cerrno != 0} {
    set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
$_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}
# 1. execute the command
if [catch {cli_open} result] {
    error $result $errorInfo
} else {
    array set cli1 $result
}

# save exact execution time for command
set time_now [clock seconds]
# execute command
```

```
if [catch {cli_exec $cli1(fd) $_show_cmd} result] {
    error $result $errorInfo
} else {
    set cmd_output $result
    # format output: remove trailing router prompt
    regexp {\n*(.*\n)([^\n]*)$} $result dummy cmd_output
}
if [catch {cli_close $cli1(fd) $cli1(tty_id)} result] {
    error $result $errorInfo
}
# 2. log the success of the CLI command
set msg [format "Command \"%s\" executed successfully" $_show_cmd]
action_syslog priority info msg $msg
if {$_cerrno != 0} {
    set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
        $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}
# 3. if _log_file is defined, then attach it to the file
if {[info exists _log_file]} {
    # attach output to file
    if [catch {open $_log_file a+} result] {
        error $result
    }
    set fileD $result
    # save timestamp of command execution
    #       (Format = 00:53:44 PDT Mon May 02 2005)
    set time_now [clock format $time_now -format "%T %Z %a %b %d %Y"]
    puts $fileD "%%% Timestamp = $time_now"
    puts $fileD $cmd_output
    close $fileD
}
# 4. if _email_server is defined send the email out
if {[info exists _email_server]} {
    set routername [info hostname]
    if {[string match "" $routername]} {
error "Host name is not configured"
    }
    if [catch {smtp_subst [file join $tcl_library email_template_cmd.tm]} \
    result] {
error $result $errorInfo
    }
    if [catch {smtp_send_email $result} result] {
error $result $errorInfo
    }
}
```

sl_intf_down.tcl Sample Policy

The following sample policy runs when a configurable syslog message is logged. The policy executes a configurable CLI command and e-mails the results.

```
::cisco::eem::event_register_syslog occurs 1 pattern $_syslog_pattern maxrun 90
#------------------------------------------------------------------
# EEM policy to monitor for a specified syslog message.
# Designed to be used for syslog interface-down messages.
# When event is triggered, the given config commands will be run.
#
# July 2005, Cisco EEM team
#
# Copyright (c) 2005 by cisco Systems, Inc.
# All rights reserved.
#------------------------------------------------------------------
```

```
### The following EEM environment variables are used:
###
### _syslog_pattern (mandatory)        - A regular expression pattern match string
###                                      that is used to compare syslog messages
###                                      to determine when policy runs
### Example: _syslog_pattern             .*UPDOWN.*FastEthernet0/0.*
###
### _email_server (mandatory)          - A Simple Mail Transfer Protocol (SMTP)
###                                      mail server used to send e-mail.
### Example: _email_server               mailserver.example.com
###
### _email_from (mandatory)            - The address from which e-mail is sent.
### Example: _email_from                 devtest@example.com
###
### _email_to (mandatory)              - The address to which e-mail is sent.
### Example: _email_to                   engineering@example.com
###
### _email_cc (optional)               - The address to which the e-mail must
###                                      be copied.
### Example: _email_cc                   manager@example.com
###
### _config_cmd1 (optional)            - The first configuration command that
###                                      is executed.
### Example: _config_cmd1                interface Ethernet1/0
###
### _config_cmd2 (optional)            - The second configuration command that
###                                      is executed.
### Example: _config_cmd2                no shutdown
###
# check if all the env variables we need exist
# If any of them doesn't 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 $errorInfo
}
if {![info exists _email_cc]} {
     #_email_cc is an option, must set to empty string if not set.
     set _email_cc ""
}
namespace import ::cisco::eem::*
namespace import ::cisco::lib::*
# 1. query the information of latest triggered eem event
array set arr_einfo [event_reqinfo]
if {$_cerrno != 0} {
    set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
      $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}
set msg $arr_einfo(msg)
set config_cmds ""
# 2. execute the user-defined config commands
if [catch {cli_open} result] {
    error $result $errorInfo
```

```
} else {
    array set cli1 $result
}

if [catch {cli_exec $cli1(fd) "config t"} result] {
    error $result $errorInfo
}
if {[info exists _config_cmd1]} {
    if [catch {cli_exec $cli1(fd) $_config_cmd1} result] {
        error $result $errorInfo
    }
    append config_cmds $_config_cmd1
}
if {[info exists _config_cmd2]} {
    if [catch {cli_exec $cli1(fd) $_config_cmd2} result] {
        error $result $errorInfo
    }
    append config_cmds "\n"
    append config_cmds $_config_cmd2
}
if [catch {cli_exec $cli1(fd) "end"} result] {
    error $result $errorInfo
}
if [catch {cli_close $cli1(fd) $cli1(tty_id)} result] {
    error $result $errorInfo
}
after 60000
# 3. send the notification email
set routername [info hostname]
if {[string match "" $routername]} {
    error "Host name is not configured"
}
if [catch {smtp_subst [file join $tcl_library email_template_cfg.tm]} result] {
    error $result $errorInfo
}
if [catch {smtp_send_email $result} result] {
    error $result $errorInfo
}
```

The following e-mail template file is used with the preceding EEM sample policy:

```
email_template_cfg.tm
Mailservername: $_email_server
From: $_email_from
To: $_email_to
Cc: $_email_cc
Subject: From router $routername: Periodic $_show_cmd Output
$cmd_output
```

Tracing Tcl set Command Operations: Example

Tcl is a flexible language. One of the flexible aspects of Tcl is that you can override commands. In this example, the Tcl **set** command is renamed as _set, and a new version of the **set** command is created that displays a message containing the text "setting" and appends the scalar variable that is being set. This example can be used to trace all instances of scalar variables being set.

```
rename set _set
proc set {var args} {
   puts [list setting $var $args]
   uplevel _set $var $args
```

```
};
When this is placed in a policy, a message is displayed anytime a scalar variable is set,
for example:

02:17:58: sl_intf_down.tcl[0]: setting test_var 1
```

Additional References

The following sections provide references related to configuring and managing Embedded Event Manager policies.

Related Documents

Related Topic	Document Title
Embedded Event Manager commands	*Embedded Event Manager Commands* module in the *System Monitoring Command Reference for Cisco ASR 9000 Series Routers*
Route processor failover commands	Hardware Redundancy and Node Administration Commands module in the *Interface and Hardware Component Command Reference for Cisco ASR 9000 Series Routers*
Cisco IOS XR XML API material	*Cisco IOS XR XML API Guide*
Cisco IOS XR getting started material	*Cisco ASR 9000 Series Aggregation Services Router Getting Started Guide*
Information about user groups and task IDs	*Configuring AAA Services* module in the *System Security Configuration Guide for Cisco ASR 9000 Series Routers*

Standards

Standards	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	—

MIBs

MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs	Title
No new or modified RFCs are supported by this feature, and support for existing RFCs has not been modified by this feature.	—

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	*http://www.cisco.com/cisco/web/support/index.html*

Embedded Event Manager Policy Tcl Command Extension Reference

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

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

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

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

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

```
[type ?]
```

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

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

Embedded Event Manager Event Registration Tcl Command Extensions

The following EEM event registration Tcl command extensions are supported:

event_register_appl

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

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

Syntax

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

Arguments

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

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

Result String

None

Set _cerrno

No

event_register_cli

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

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

Syntax

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

Arguments

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

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

Result String

None

Set _cerrno

No

event_register_config

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

Syntax

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

Arguments

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

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

Result String

None

Set _cerrno

No

event_register_counter

Registers for a counter event as both a publisher and a subscriber. Use this Tcl command extension to run a policy on the basis of a named counter crossing a threshold. This event counter, as a subscriber, identifies the name of the counter to which it wants to subscribe and depends on another policy or another process to actually manipulate the counter. For example, let policyB act as a counter policy, whereas policyA (although it does not need to be a counter policy) uses register_counter, counter_modify, or unregister_counter Tcl command extensions to manipulate the counter defined in policyB.

Syntax

```
event_register_counter name ? entry_op gt|ge|eq|ne|lt|le entry_val ?
exit_op gt|ge|eq|ne|lt|le exit_val ? [queue_priority low|normal|high]
[maxrun ?] [nice 0|1]
```

Arguments

name	(Mandatory) Name of the counter.
entry_op	(Mandatory) Entry comparison operator used to compare the current counter value with the entry value; if true, an event is raised and event monitoring is disabled until exit criteria are met.
entry_val	(Mandatory) Value with which the current counter value should be compared, to decide if the counter event should be raised.
exit_op	(Mandatory) Exit comparison operator used to compare the current counter value with the exit value; if true, event monitoring for this event is reenabled.
exit_val	(Mandatory) Value with which the current counter value should be compared to decide if the exit criteria are met.
queue_priority	(Optional) Priority level at which the script will be queued; normal priority is greater than low priority but less than high priority. The priority here is not execution priority, but queuing priority. If this argument is not specified, the default priority is normal.
maxrun	(Optional) Maximum run time of the script (specified in SSSSSSSSSS[.MMM] format, where SSSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the default 20-second run-time limit is used.
nice	(Optional) Policy run-time priority setting. When the *nice* argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.

Result String

None

Set _cerrno

No

event_register_hardware

Registers for an environmental monitoring hardware device that is specified by the hardware event and condition.

Syntax

```
event_register_hardware env_device ? env_cond ?
[priority normal|low|high] [maxrun_sec ?] [maxrun_nsec ?] [nice 0|1]
```

Arguments

env_device	(Mandatory) Environmental device that is used to monitor.The integer number must be inclusively between 1 and 2147483647. This is a bit mask that monitors multiple types of environmental devices. The following supported devices and their corresponding bitmasks are listed: • 0x0001 chassis • 0x0002 backplane • 0x0004 slot • 0x0008 card • 0x0010 port • 0x0020 fan • 0x0040 group of power supplies • 0x0080 power supply • 0x0100 sensor They can be bit wise OR'ed to monitor multiple devices.
env_cond	(Mandatory) Environmental condition that is used to monitor. This is a bit mask that monitors multiple kinds of environmental conditions. The following supported environmental conditions and their corresponding bitmasks are listed: • 0x0001 low warning • 0x0002 high warning • 0x0004 warning • 0x0010 low critical • 0x0020 high critical • 0x0040 critical • 0x0100 pre-shutdown • 0x0200 shutdown

priority	(Optional) Priority level that the script is queued. If not specified, the default uses the normal priority.
maxrun_sec, maxrun_nsec	(Optional) Maximum runtime of the script that is specified in seconds and nanoseconds. The integer number must be inclusively between 0 and 2147483647. If not specified, use the default 20-second run-time limit.
nice	(Optional) Maximum runtime of the script that is specified in seconds and nanoseconds. The integer number must be inclusively between 0 and 2147483647. If not specified, use the default 20-second run-time limit.

Result String

None

Set _cerrno

No

event_register_none

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

Syntax

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

Arguments

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

Result String

None

Set _cerrno

No

event_register_oir

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

Syntax

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

Arguments

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

Result String

None

Set _cerrno

No

event_register_process

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

Syntax

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

Arguments

abort	(Mandatory) Abnormal process termination. Process may terminate because of exiting with a nonzero exit status, receiving a kernel-generated signal, or receiving a SIGTERM or SIGKILL signal that is not sent because of user request.
term	(Mandatory) Normal process termination.

start	(Mandatory) Process start.
job_id	(Optional) Number assigned to the EEM policy that published the process event. Number is set to 798, because all other numbers are reserved for Cisco use.
instance	(Optional) Process instance ID. If specified, this argument must be an integer between 1 and 4294967295, inclusive.
path	(Optional) Process pathname (regular expression string).
node	(Optional) The node name is a string that consists of the word "node" followed by two fields separated by a slash (/), using the following format: node<slot-number>/<cpu-number> The slot-number is the hardware slot number. The cpu-number is the hardware CPU number. For example, the SP CPU in a Supervisor card on a Cisco Catalyst 6500 series switch located in slot 0 would be specified as node0/0. The RP CPU in a Supervisor card on a Cisco Catalyst 6500 series switch located in slot 0 would be addressed as node0/1. If the *node* argument is not specified, the default node specification is always the regular expression pattern match of * representing all applicable nodes.
queue_priority	(Optional) Priority level at which the script will be queued; normal priority is greater than low priority but less than high priority. The priority here is not execution priority, but queuing priority. If this argument is not specified, the default priority is normal.
maxrun	(Optional) Maximum run time of the script (specified in SSSSSSSSSS[.MMM] format, where SSSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the default 20-second run-time limit is used.
nice	(Optional) Policy run-time priority setting. When the *nice* argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.
tag	Tag is acceptable but ignored. Cisco IOS EEM scripts with the tag option can run in an Cisco IOS XR software environment without any error. Since Cisco IOS XR software does not support multiple events, the tag has no effect.

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

Result String

None

Set _cerrno

No

event_register_snmp

Registers for a Simple Network Management Protocol (SNMP) statistics event. Use this Tcl command extension to run a policy when a given counter specified by an SNMP object ID (oid) crosses a defined threshold. When a snmp policy is registered, a poll timer is specified. Event matching occurs when the poll timer for the

registered event expires. The **snmp-server manager** CLI command must be enabled for the SNMP notifications to work using Tcl policies.

Syntax

```
event_register_snmp [tag ?] oid ? get_type exact|next
entry_op gt|ge|eq|ne|lt|le entry_val ?
entry_type value|increment|rate
[exit_comb or|and]
[exit_op gt|ge|eq|ne|lt|le] [exit_val ?]
[exit_type value|increment|rate]
[exit_time ?] poll_interval ? [average_factor ?]
[queue_priority low|normal|high|last]
[maxrun ?] [nice 0|1]
```

Aruguments

tag	(Optional) String identifying a tag that can be used with the trigger Tcl command extension to support multiple event statements within a Tcl script.
entry_op	(Mandatory) Entry comparison operator used to compare the current OID data value with the entry value; if true, an event will be raised and event monitoring will be disabled until exit criteria are met.
get_type	(Mandatory) Type of SNMP get operation that needs to be applied to the OID specified. If the get_type argument is "exact," the value of the specified OID is retrieved; if the get_type argument is "next," the value of the lexicographical successor to the specified OID is retrieved.
entry_val	(Mandatory) Value with which the current oid data value should be compared to decide if the SNMP event should be raised.
entry-type	Specifies a type of operation to be applied to the object ID specified by the entry-val argument. Value is defined as the actual value of the entry-val argument. Increment uses the entry-val field as an incremental difference and the entry-val is compared with the difference between the current counter value and the value when the event was last triggered (or the first polled sample if this is a new event). A negative value checks the incremental difference for a counter that is decreasing. Rate is defined as the average rate of change over a period of time. The time period is the average-factor value multiplied by the poll-interval value. At each poll interval the difference between the current sample and the previous sample is taken and recorded as an absolute value. An average of the previous average-factor value samples is taken to be the rate of change.
exit_comb	(Optional) Exit combination operator used to indicate the combination of exit condition tests required to decide if the exit criteria are met so that the event monitoring can be reenabled. If it is "and," both exit value and exit time tests must be passed to meet the exit criteria. If it is "or," either exit value or exit time tests can be passed to meet the exit criteria When exit_comb is "and," exit_op, and exit_val (exit_time) must exist. When exit_comb is "or," (exit_op and exit_val) or (exit_time) must exist.
exit_op	(Optional) Exit comparison operator used to compare the current oid data value with the exit value; if true, event monitoring for this event will be reenabled.

exit_val	(Optional) Value with which the current oid data value should be compared to decide if the exit criteria are met.
exit-type	(Optional) Specifies a type of operation to be applied to the object ID specified by the exit-val argument. If not specified, the value is assumed. Value is defined as the actual value of the exit-val argument. Increment uses the exit-val field as an incremental difference and the exit-val is compared with the difference between the current counter value and the value when the event was last triggered (or the first polled sample if this is a new event). A negative value checks the incremental difference for a counter that is decreasing. Rate is defined as the average rate of change over a period of time. The time period is the average-factor value multiplied by the poll-interval value. At each poll interval the difference between the current sample and the previous sample is taken and recorded as an absolute value. An average of the previous average-factor value samples is taken to be the rate of change.
exit_time	(Optional) Number of POSIX timer units after an event is raised when event monitoring will be enabled again. Specified in SSSSSSSSSS[.MMM] format where SSSSSSSSSS must be an integer number representing seconds between 0 and 4294967295, inclusive. MMM represents milliseconds and must be an integer number between 0 and 999.
poll_interval	(Mandatory) Interval between consecutive polls in POSIX timer units. Currently the interval is forced to be at least 1 second (specified in SSSSSSSSSS[.MMM] format, where SSSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999).
average-factor	(Optional) Number in the range from 1 to 64 used to calculate the period used for rate-based calculations. The average-factor value is multiplied by the poll-interval value to derive the period in milliseconds. The minimum average factor value is 1.

Result string

None

Set _cerrno

No

event_register_snmp_notification

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

Syntax

```
event_register_snmp_notification [tag ?] oid ? oid_val ?
op {gt|ge|eq|ne|lt|le}
[src_ip_address ?]
[dest_ip_address ?]
[queue_priority {normal|low|high|last}]
```

```
[maxrun ?]
[nice {0|1}]
[default ?]
[direction {incoming|outgoing}]
[msg_op {drop|send}]
```

Argument

tag	(Optional) String identifying a tag that can be used with the trigger Tcl command extension to support multiple event statements within a Tcl script.
oid	(Mandatory) OID number of the data element in SNMP dot notation (for example, 1.3.6.1.2.1.2.1.0). If the specified OID ends with a dot (.), then all OIDs that start with the OID number before the dot are matched. It supports all OID supported by SNMP in XR.
oid_val	(Mandatory) OID value with which the current OID data value should be compared to decide if the SNMP event should be raised.
op	(Mandatory) Comparison operator used to compare the current OID data value with the SNMP Protocol Data Unit (PDU) OID data value; if this is true, an event is raised.
src_ip_address	(Optional) Source IP address where the SNMP notification trap originates. The default is all; it is set to receive SNMP notification traps from all IP addresses. This option will not be supported in XR as src_ip_address is only for incoming trap which is not supported in EEM XR.
dest_ip_address	(Optional) Destination IP address where the SNMP notification trap is sent. The default is all; it is set to receive SNMP traps from all destination IP addresses.
default	(Optional) Specifies the time period in seconds during which the snmp notification event detector waits for the policy to exit. Thetime periodis specified in sssssssss[.mmm] format, where sssssssss must be an integer representing seconds between 0 and 4294967295 and mmm must be an integer representing milliseconds between 0 and 999
direction	(Optional) The direction of the incoming or outgoing SNMP trap or inform PDU to filter. The default value is outgoing. For XR direction incoming will not be supported and policy registration will fail if user provides direction as incoming.
msg_op	(Optional) The action to be taken on the SNMP PDU (drop it or send it) once the event is triggered. The default value is send. For XR msg_op drop will not be supported and policy registration will fail if user provides msg_op as drop.

Result String

None

Set _cerrno

No

event_register_stat

Registers for a statistics event. Use this Tcl command extension to run a policy when a given statistical counter crosses a defined threshold.

The following three fields are listed to uniquely identify the statistics counter that the EEM keyword monitors:

- Data element name corresponds to the argument name. For example, the ifstats-generic name is defined as interface generic statistics.
- The first modifier of the data element corresponds to the *modifier_1* argument. For example, Ethernet1_0 is defined as the first modifier for ifstats-generic, which qualifies the interface generic statistics to be specific for the Ethernet interface.
- The second modifier of the data element corresponds to the *modifier_2* argument. For example, input-ptks is defined as the second modifier for ifstats-generic, which further qualifies the interface statistics for the specific Ethernet interface is the number of packets received.

Syntax

```
event_register_stat name ? [modifier_1 ?] [modifier_2 ?]

entry_op gt|ge|eq|ne|lt|le entry_val ? [exit_comb or|and]

[exit_op gt|ge|eq|ne|lt|le] [exit_val ?] [exit_time_sec ?] [exit_time_nsec ?]
[poll_interval_sec ?] [poll_interval_nsec ?] [priority normal|low|high]
[maxrun_sec ?] [maxrun_nsec ?] [nice 0|1] [tag ?]
```

Arguments

name	(Mandatory) Statistics data element name.
modifier_1	Mandatory for interface statistics but optional for others. For interface statistics, this variable is the interface name. To get the interface name, use the **show interface brief** command. This command lists all the currently configured interface names designated by a slash (/), for example, Ethernet 1/0. When you want this interface to be configured for the *modifier_1* argument, change the slash to an underscore.
modifier_2	Mandatory for interface statistics but optional for others. For interface statistics, this variable is the interface statistic name. To get the interface statistic name, use the **show event manager statistics -table** command with the **all** keyword to list all the classes of statistics. Then, use the **show event manager statistics -table** command with the *name* argument to get the specific statistics name for modifier_2.
entry_op	(Mandatory) Entry comparison operator that is used to compare the current statistics value with the entry value. If true, an event is raised and event monitoring is disabled until the exit criteria is met.
entry_val	(Mandatory) Value in which the current statistical counter value that is compared to decide if the statistical event can be raised.

exit_comb	(Mandatory) Exit combination operator that indicates the combination of exit condition tests that are required to decide if the exit criteria is met so that event monitoring is reenabled. If so, both exit value and exit time tests must be passed to meet the exit criteria.Or either exit value or exit time tests are passed to meet the exit criteria. *exit_comb* and *exit_op*, *exit_val* arguments (*exit_time_sec* argument or *exit_time_nsec* argument) must exist. *exit_comb* argument or (*exit_op* and *exit_val* arguments) or (*exit_time_sec* argument or *exit_time_nsec* argument) must exist.
exit_op	Exit comparison operator that is used to compare the current statistics value with the exit value. If true, event monitoring for this event is reenabled.
exit_val	Value in which the current statistical counter value is compared to decide if the exit criteria is met.
exit_time_sec exit_time_nse	Number of POSIX timer units after the event is raised when event monitoring is enabled again. The integer number must be between 0 and 2147483647, inclusive.
poll_interval_sec poll_interval_nsec	Either the *poll_interval_sec* or *poll_interval_nsec* arguments must be specified. The interval must be between the consecutive polls in POSIX time units. Currently, it is forced to be at least one second. The integer number must be between 0 and 2147483647, inclusive.
priority	(Optional) Priority level that is queued for the script. If not specified, the default is using the normal priority.
maxrun_sec, maxrun_nsec	(Optional) Maximum run time of the script that is specified in seconds and nanoseconds. If not specified, 20-second run-time limit is used as the default. The integer number must be between 0 and 2147483647, inclusive.
nice	(Optional) When the *nice* argument is set to the value of 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.
tag	Tag is acceptable but ignored.Cisco IOS EEM scripts with the tag option can run in an Cisco IOS XR software environment without any error. Since Cisco IOS XR software does not support multiple events, the tag has no effect.

Note Exit criteria can be time-based, value-based, or both. Event monitoring is not reenabled until the exit criteria is met.

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

Reslt String

None

Set _cerrno

No

event_register_syslog

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

Syntax

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

Arguments

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

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

Table 16: Severity Level Mapping For Syslog Events

Severity Keyword	Syslog Priority	Description
severity_fatal	LOG_EMERG (0)	System is unusable.

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

Result String

None

Set _cerrno

No

event_register_timer

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

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

Syntax

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

Arguments

watchdog	(Mandatory) Watchdog timer.
countdown	(Mandatory) Countdown timer.
absolute	(Mandatory) Absolute timer.
cron	(Mandatory) CRON timer.

name	(Optional) Name of the timer.
cron_entry	(Optional) Entry must be specified if the CRON timer type is specified. Must not be specified if any other timer type is specified. A cron_entry is a partial UNIX crontab entry (the first five fields) as used with the UNIX CRON daemon. A cron_entry specification consists of a text string with five fields. The fields are separated by spaces. The fields represent the time and date when CRON timer events will be triggered. The fields are described in Table 17: Time and Date When CRON Events Will Be Triggered , on page 113 . Ranges of numbers are allowed. Ranges are two numbers separated with a hyphen. The specified range is inclusive. For example, 8-11 for an hour entry specifies execution at hours 8, 9, 10, and 11. A field may be an asterisk (*), which always stands for "first-last." Lists are allowed. A list is a set of numbers (or ranges) separated by commas. Examples: "1,2,5,9" and "0-4,8-12". Step values can be used in conjunction with ranges. Following a range with "/<number>" specifies skips of the number's value through the range. For example, "0-23/2" is used in the hour field to specify an event that is triggered every other hour. Steps are also permitted after an asterisk, so if you want to say "every two hours", use "*/2". Names can also be used for the month and the day of week fields. Use the first three letters of the particular day or month (case does not matter). Ranges or lists of names are not allowed. The day on which a timer event is triggered can be specified by two fields: day of month and day of week. If both fields are restricted (that is, are not *), an event will be triggered when either field matches the current time. For example, "30 4 1,15 * 5" would cause an event to be triggered at 4:30 a.m. on the 1st and 15th of each month, plus every Friday. Instead of the first five fields, one of seven special strings may appear. These seven special strings are described in Table 18: Special Strings for cron_entry, on page 113. Example 1: "0 0 1,15 * 1" would trigger an event at midnight on the 1st and 15th of each month, as well as on every Monday. To specify days by only one field, the other field should be set to *; "0 0 * * 1" would trigger an event at midnight only on Mondays. Example 2: "15 16 1 * *" would trigger an event at 4:15 p.m. on the first day of each month. Example 3: "0 12 * * 1-5" would trigger an event at noon on Monday through Friday of each week. Example 4: "@weekly" would trigger an event at midnight once a week on Sunday.
time	(Optional) Time must be specified if a timer type other than CRON is specified. Must not be specified if the CRON timer type is specified. For watchdog and countdown timers, the number of seconds and milliseconds until the timer expires; for the absolute timer, the calendar time of the expiration time. Time is specified in SSSSSSSSSS[.MMM] format, where SSSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999. An absolute expiration date is the number of seconds and milliseconds since January 1, 1970. If the date specified has already passed, the timer expires immediately.

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

Table 17: Time and Date When CRON Events Will Be Triggered

Field	Allowed Values
minute	0-59
hour	0-23
day of month	1-31
month	1-12 (or names, see Table 18: Special Strings for cron_entry, on page 113)
day of week	0-7 (0 or 7 is Sun, or names; see Table 18: Special Strings for cron_entry, on page 113)

Table 18: Special Strings for cron_entry

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

Result String

None

Set _cerrno

No

See Also

event_register_timer_subscriber, on page 114

event_register_timer_subscriber

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

Syntax

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

Arguments

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

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

Result String

None

Set _cerrno

No

See Also

event_register_timer, on page 111

event_register_track

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

Syntax

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

Arguments

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

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

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

Result String

None

Set _cerrno

No

event_register_wdsysmon

Registers for a Watchdog system monitor event. Use this Tcl command extension to register for a composite event which is a combination of several subevents or conditions. For example, you can use the **event_register_wdsysmon** command to register for the combination of conditions wherein the CPU usage of a certain process is over 80 percent, and the memory used by the process is greater than 50 percent of its initial allocation. This Tcl command extension is supported only in Software Modularity images.

Syntax

```
event_register_wdsysmon [timewin ?]
[sub12_op and|or|andnot]
[sub23_op and|or|andnot]
[sub34_op and|or|andnot]
[sub1 subevent-description]
[sub2 subevent-description]
[sub3 subevent-description]
[sub4 subevent-description] [node ?]
[queue_priority low|normal|high]
[maxrun ?] [nice 0|1]
```

Arguments

timewin	(Optional) Time window within which all of the subevents have to occur in order for an event to be generated and is specified in SSSSSSSSSS[.MMM] format. SSSSSSSSSS format must be an integer representing seconds between 0 and 4294967295, inclusive. MMM format must be an integer representing milliseconds between 0 and 999).
sub12_op	(Optional) Combination operator for comparison between subevent 1 and subevent 2.
sub34_op	(Optional) Combination operator for comparison between subevent 1 and 2, subevent 3, and subevent 4.
sub1	(Optional) Subevent 1 is specified.
subevent-description	(Optional) Syntax for the subevent.
sub2	(Optional) Subevent 2 is specified.
sub3	(Optional) Subevent 3 is specified.

sub4	(Optional) Subevent 4 is specified.
node	(Optional) Node name to be monitored for deadlock conditions is a string that consists of the word 'node', which is followed by two fields separated by a slash (/) using the following format: `node<slot-number>/<cpu-number>` The slot-number is the hardware slot number. The cpu-number is the hardware CPU number. For example, the SP CPU in a Supervisor card on a Cisco Catalyst 6500 Series Switch located in slot 0 is specified as node0/0. The RP CPU in a Supervisor card on a Cisco Catalyst 6500 Series Switch located in slot 0 is addressed as node0/1. If the node argument is not specified, the default node specification is the local node on which the registration is done.
queue_priority	(Optional) Priority level at which the script is queued; normal priority is greater than low priority but less than high priority. The priority here is not execution priority, but queuing priority. If this argument is not specified, the default priority is normal.
maxrun	(Optional) Maximum run time of the script that is specified in SSSSSSSSSS[.MMM] format. SSSSSSSSSS format must be an integer representing seconds between 0 and 4294967295, inclusive. MMM format must be an integer representing milliseconds between 0 and 999. If this argument is not specified, the default 20-second run-time limit is used.
nice	(Optional) Policy run-time priority setting. When the *nice* argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.

Subevents

The syntax of subevent descriptions can be one of seven cases.

For arguments in subevent description, the following constraints apply on the value of number arguments:

- For dispatch_mgr, val must be an integer between 0 and 4294967295, inclusive.
- For cpu_proc and cpu_tot, val must be an integer between 0 and 100, inclusive.
- For mem_proc, mem_tot_avail, and mem_tot_used, if is_percent is FALSE, val must be an integer between 0 and 4294967295, inclusive.

1. deadlock procname ?

Arguments

procname	(Mandatory) Regular expression that specifies the process name that you want to monitor for deadlock conditions. This subevent ignores the time window even if it is given.

1. dispatch_mgr [procname ?] [op gt|ge|eq|ne|lt|le] [val ?] [period ?]

Arguments

procname	(Optional) Regular expression that specifies the process name that you want to monitor for the dispatch_manager status.
op	(Optional) Comparison operator that is used to compare the collected number of events with the specified value. If true, an event is raised.
val	(Optional) Value in which the number of events that have occurred is compared.
period	(Optional) Time period for the number of events that have occurred and is specified in SSSSSSSSSS[.MMM] format. SSSSSSSSSS format must be an integer representing seconds between 0 and 4294967295, inclusive. MMM format must be an integer representing milliseconds between 0 and 999. If this argument is not specified, the most recent sample is used.

1. cpu_proc [procname ?] [op gt|ge|eq|ne|lt|le] [val ?] [period ?]

Arguments

procname	(Optional) Regular expression that specifies the process name that you want to monitor for CPU utilization conditions.
op	(Optional) Comparison operator that is used to compare the collected CPU usage sample percentage with the specified percentage value. If true, an event is raised.
val	(Optional) Percentage value in which the average CPU usage during the sample period is compared.
period	(Optional) Time period for averaging the collection of samples and is specified in SSSSSSSSSS[.MMM] format. SSSSSSSSSS format must be an integer representing seconds between 0 and 4294967295, inclusive. MMM format must be an integer representing milliseconds between 0 and 999. If this argument is not specified, the most recent sample is used.

1. cpu_tot [op gt|ge|eq|ne|lt|le] [val ?] [period ?]

Arguments

op	(Optional) Comparison operator that is used to compare the collected total system CPU usage sample percentage with the specified percentage value. If true, an event is raised.
val	(Optional) Percentage value in which the average CPU usage during the sample period is compared.
period	(Optional) Time period for averaging the collection of samples and is specified in SSSSSSSSSS[.MMM] format. SSSSSSSSSS format must be an integer representing seconds between 0 and 4294967295, inclusive. MMM format must be an integer representing milliseconds between 0 and 999. If this argument is not specified, the most recent sample is used.

1. mem_proc [procname ?] [op gt|ge|eq|ne|lt|le] [val ?] [is_percent TRUE|FALSE] [period ?]

Arguments

procname	(Optional) Regular expression that specifies the process name that you want to monitor for memory usage.
op	(Optional) Comparison operator that is used to compare the collected memory used with the specified value. If true, an event is raised.
val	(Optional) Percentage or an absolute value that is specified in kilobytes. A percentage represents the difference between the oldest sample in the specified time period and the latest sample. If memory usage increased from 150 KB to 300 KB within the time period, the percentage increase is 100. This is the value in which the measured value is compared.
is_percent	(Optional) If set to TRUE, the percentage value is collected and compared. Otherwise, the absolute value is collected and compared.
period	(Optional) If is_percent is set to TRUE, the time period for the percentage is computed. Otherwise, the time period for the collection samples is averaged and is specified in SSSSSSSSSS[.MMM] format. SSSSSSSSSS format must be an integer representing seconds between 0 and 4294967295, inclusive. MMM format must be an integer representing milliseconds between 0 and 999. If this argument is not specified, the most recent sample is used.

1. mem_tot_avail [op gt|ge|eq|ne|lt|le] [val ?] [is_percent TRUE|FALSE] [period ?]

Arguments

op	(Optional) Comparison operator that is used to compare the collected available memory with the specified value. If true, an event is raised.
val	(Optional) Percentage or an absolute value that is specified in kilobytes. A percentage represents the difference between the oldest sample in the specified time period and the latest sample. If available memory usage has decreased from 300 KB to 150 KB within the time period, the percentage decrease is 50. This is the value in which the measured value is compared.
is_percent	(Optional) If set to TRUE, the percentage value is collected and compared. Otherwise, the absolute value is collected and compared.
period	(Optional) If is_percent is set to TRUE, the time period for the percentage is computed. Otherwise, the time period for the collection samples is averaged and is specified in SSSSSSSSSS[.MMM] format. SSSSSSSSSS format must be an integer representing seconds between 0 and 4294967295, inclusive. MMM format must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the most recent sample is used.

1. mem_tot_used [op gt|ge|eq|ne|lt|le] [val ?] [is_percent TRUE|FALSE] [period ?]

Arguments

op	(Optional) Comparison operator that is used to compare the collected used memory with the specified value. If true, an event is raised.

val	(Optional) Percentage or an absolute value that is specified in kilobytes. A percentage represents the difference between the oldest sample in the specified time period and the latest sample. If memory usage has increased from 150 KB to 300 KB within the time period, the percentage increase is 100. This is the value in which the measured value is compared.
is_percent	(Optional) If set to TRUE, the percentage value is collected and compared. Otherwise, the absolute value is collected and compared.
period	(Optional) If is_percent is set to TRUE, the time period for the percentage is computed. Otherwise, the time period for the collection samples is averaged and is specified in SSSSSSSSSS[.MMM] format. SSSSSSSSSS format must be an integer representing seconds between 0 and 4294967295, inclusive. MMM format must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the most recent sample is used. **Note** This argument is mandatory if is_percent is set to TRUE; otherwise, it is optional.

Result String

None

Set _cerrno

No

Note Inside a subevent description, each argument is position as independent.

Embedded Event Manager Event Information Tcl Command Extension

The following EEM Event Information Tcl Command Extensions are supported:

event_reqinfo

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

Syntax

```
event_reqinfo
```

Arguments

None

Result String

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

For EEM_EVENT_APPLICATION

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

Event Type	Description
event_id	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
event_type	Type of event.
event_type_string	ASCII string that represents the name of the event for this event type.
event_pub_secevent_pub_msec	The time, in seconds and milliseconds, when the event was published to the Embedded Event Manager.
sub_system	Number assigned to the EEM policy that published the application event. Number is set to 798 because all other numbers are reserved for Cisco use.
type	Event subtype within the specified component.
data1data2data3data4	Argument data that is passed to the application-specific event when the event is published. The data is character text, an environment variable, or a combination of the two.

For EEM_EVENT_COUNTER

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

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

For EEM_EVENT_NONE

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

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

For EEM_EVENT_OIR

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

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

For EEM_EVENT_PROCESS (Software Modularity Only)

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

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

Event Type	Description
event_pub_secevent_pub_msec	Time, in seconds and milliseconds, when the event was published to the Embedded Event Manager.
sub_system	Number assigned to the EEM policy that published the application-specific event. Number is set to 798 because all other numbers are reserved for Cisco use.
instance	Process instance ID.
process_name	Process name.
path	Process absolute name including path.
exit_status	Process last exit status.
respawn_count	Number of times that the process was restarted.
last_respawn_seclast_respawn_msec	Calendar time when the last restart occurred.
fail_count	Number of restart attempts of the process that failed. This count will be reset to 0 when the process is successfully restarted.
Event Type	**Description**
dump_count	Number of core dumps taken of the process.
node_name	Name of the node that the process is on. The node name is a string that consists of the word "node" followed by two fields separated by a slash character using the following format: node<slot-number>/<cpu-number> The slot-number is the hardware slot number. The cpu-number is the hardware CPU number.

For EEM_EVENT_RF

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

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

Event Type	Description
event	RF progression or status event notification that caused this event to be published.

For EEM_EVENT_SYSLOG_MSG

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

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

For EEM_EVENT_TIMER_ABSOLUTE

EEM_EVENT_TIMER_COUNTDOWN

EEM_EVENT_TIMER_WATCHDOG

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

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

Event Type	Description
timer_type	Type of the timer. Can be one of the following: • watchdog • countdown • absolute
timer_time_sectimer_time_msec	Time when the timer expired.
timer_remain_sectimer_remain_msec	Remaining time before the next expiration.

For EEM_EVENT_TIMER_CRON

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

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

For EEM_EVENT_TRACK

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

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

Event Type	Description
event_pub_secevent_pub_msec	Time, in seconds and milliseconds, when the event was published to the Embedded Event Manager.
track_number	Number of the tracked object that caused the event to be triggered.
track_state	State of the tracked object when the event was triggered; valid states are up or down.

For EEM_EVENT_WDSYSMON

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

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

Where the subevent info string is for a deadlock subevent:

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

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

Where each entry is:

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

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

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

For dispatch_mgr Subevent

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

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

For cpu_proc Subevent

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

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

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

For cpu_tot Subevent

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

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

For mem_proc Subevent

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

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

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

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

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

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

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

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

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

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

- *value* is 0.

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

For mem_tot_avail Subevent

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

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

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

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

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

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

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

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

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

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

For mem_tot_used Subevent

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

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

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

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

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

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

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

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

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

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

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

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

Set _cerrno

Yes

event_reqinfo_multi

Adds a new function to retrieve the event_reqinfo data for every event that contributed to the triggering of the script. The data returned will be a list of result strings indexed by event specification tag. Error processing is the same as in event_reqinfo function.

Syntax

```
event_reqinfo_multi
```

Arguments

None

Result String

The following section shows the result string from the event reqinfo multi call:

```
"<ev-tag> {event_id %u event_type %u event_type_string
{%s} event_pub_sec %ld event_pub_msec %ld timer_type {%s} timer_time_sec
%ld timer_time_msec %ld timer_remain_sec %ld timer_remain_msec %ld}
                 <ev-tag> {event_id %u event_type %u event_type_string
{%s} event_pub_sec %ld event_pub_msec %ld oid {%s} val {%s} delta_val
{%s} exit_event {%s}}"
               Typical usage for a multi-event consisting of both a timer event and an
SNMP event might be:
               array set arr_minfo [event_reqinfo_multi]
               if {$_cerrno != 0} {
                   set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
                       $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
                   error $result
               }
               array set arr_einfo $arr_minfo(<ev-tag-for-timer-event-spec>)
               global timer_type timer_time_sec
```

```
set timer_type $arr_einfo(timer_type)
set timer_time_sec $arr_einfo(timer_time_sec)
```

The output of event_reqinfo_multi is ordered from most recent to least recent event that contributed to the triggering of the policy.

Embedded Event Manager Event Publish Tcl Command Extension

event_publish appl

Publishes an application-specific event.

Syntax

```
event_publish sub_system ? type ? [arg1 ?] [arg2 ?] [arg3 ?] [arg4 ?]
```

Arguments

sub_system	(Mandatory) Number assigned to the EEM policy that published the application-specific event. Number is set to 798 because all other numbers are reserved for Cisco use.
type	(Mandatory) Event subtype within the specified component. The sub_system and type arguments uniquely identify an application event. Must be an integer between 1 and 4294967295, inclusive.
[arg1 ?]-[arg4 ?]	(Optional) Four pieces of application event publisher string data.

Result String

None

Set _cerrno

Yes

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

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

Sample Usage

This example demonstrates how to use the **event_publish appl** Tcl command extension to execute a script *n* times repeatedly to perform some function (for example, to measure the amount of CPU time taken by a given group of Tcl statements). This example uses two Tcl scripts.

Script1 publishes a type 9999 EEM event to cause Script2 to run for the first time. Script1 is registered as a none event and is run using the Cisco IOS XR software CLI **event manager run** command. Script2 is registered as an EEM application event of type 9999, and this script checks to see if the application publish arg1 data (the iteration number) exceeds the EEM environment variable test_iterations value. If the test_iterations value is exceeded, the script writes a message and exits; otherwise the script executes the remaining statements and

reschedules another run. To measure the CPU utilization for Script2, use a value of test_iterations that is a multiple of 10 to calculate the amount of average CPU time used by Script2.

To run the Tcl scripts, enter the following Cisco IOS XR software commands:

```
configure terminal
 event manager environment test_iterations 100
 event manager policy script1.tcl
 event manager policy script2.tcl
 end
event manager run script1.tcl
```

The Tcl script Script2 is executed 100 times. If you execute the script without the extra processing and derive the average CPU utilization, and then add the extra processing and repeat the test, you can subtract the former CPU utilization from the later CPU utilization to determine the average for the extra processing.

Script1 (script1.tcl)

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

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

# Query the event info.
array set arr_einfo [event_reqinfo]
if {$_cerrno != 0} {
    set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
        $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}

action_syslog priority info msg "EEM application_publish test start"
if {$_cerrno != 0} {
    set result [format \
          "component=%s; subsys err=%s; posix err=%s;\n%s" \
          $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}

# Cause the first iteration to run.
event_publish sub_system 798 type 9999 arg1 0
if {$_cerrno != 0} {
    set result [format \
          "component=%s; subsys err=%s; posix err=%s;\n%s" \
          $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}
```

Script2 (script2.tcl)

```
::cisco::eem::event_register_appl sub_system 798 type 9999

# Check if all the required environment variables exist.
# If any required environment variable does not exist, print out an error msg and quit.
if {![info exists test_iterations]} {
    set result \
      "Policy cannot be run: variable test_iterations has not been set"
    error $result $errorInfo
}
```

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

# Query the event info.
array set arr_einfo [event_reqinfo]
if {$_cerrno != 0} {
    set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
        $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}
# Data1 contains the arg1 value used to publish this event.
set iter $arr_einfo(data1)

# Use the arg1 info from the previous run to determine when to end.
if {$iter >= $test_iterations} {
    # Log a message.
    action_syslog priority info msg "EEM application_publish test end"
    if {$_cerrno != 0} {
      set result [format \
            "component=%s; subsys err=%s; posix err=%s;\n%s" \
            $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
      error $result
    }
    exit 0
}
set iter [expr $iter + 1]

# Log a message.
set msg [format "EEM application_publish test iteration %s" $iter]
action_syslog priority info msg $msg
if {$_cerrno != 0} {
    set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
      $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}

# Do whatever processing that you want to measure here.

# Cause the next iteration to run. Note that the iteration is passed to the
# next operation as arg1.
event_publish sub_system 798 type 9999 arg1 $iter
if {$_cerrno != 0} {
    set result [format \
          "component=%s; subsys err=%s; posix err=%s;\n%s" \
          $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}
```

Embedded Event Manager Multiple Event Support Tcl Command Extensions

Attribute

Specifies a complex event used for Multi Event Support.

Syntax

```
attribute tag ? [occurs ?]
```

Arguments

tag	Specifies a tag using the *event-tag* argument that can be used with the **attribute** command to associate an event.
occurs	(Optional) Specifies the number of occurrences before an EEM event is triggered. If not specified, an EEM event is triggered on the first occurrence. The range is from 1 to 4294967295

Result String

None

Example:

attribute tag 1 occurs 1

Correlate

Builds a single complex event and allows Boolean logic to relate events.

Syntax

```
correlate event ? event ?
```

Arguments

event	Specifies the event that can be used with the **trigger** command to support multiple event statements within an script. If the event associated with the *event-tag* argument occurs for the number of times specified by the **trigger** command, the result is true. If not, the result is false.
andnot	(Optional) Specifies that if event 1 occurs the action is executed, and if event 2 and event 3 occur together the action is not executed.
and	(Optional) Specifies that if event 1 occurs the action is executed, and if event 2 and event 3 occur together the action is executed.
or	(Optional) Specifies that if event 1 occurs the action is executed, or else if event 2 and event 3 occur together the action is executed.

Result String

None

Example:

correlate event 1 or event 2 and event 3

Trigger

Specifies the multiple event configuration ability of Embedded Event Manager (EEM) events. A multiple event is one that can involve one or more event occurrences and a time period for the event to occur. The events are raised based on the specified parameters.

Syntax

```
trigger [occurs ?] [period ?] [period-start ?] [delay ?]
```

Arguments

occurs	(Optional) Specifies the number of times the total correlation occurs before an EEM event is raised. When a number is not specified, an EEM event is raised on the first occurrence. The range is from 1 to 4294967295.
period	(Optional) Time interval in seconds and optional milliseconds, during which the one or more occurrences must take place. This is specified in the format sssssssss[.mmm], where sssssssss must be an integer number representing seconds between 0 and 4294967295, inclusive and mmm represents milliseconds and must be an integer number between 0 to 999.
period-start	(Optional) Specifies the start of an event correlation window. If not specified, event monitoring is enabled after the first CRON period occurs.
delay	(Optional) Specifies the number of seconds and optional milliseconds after which an event will be raised if all the conditions are true (specified in the format sssssssss[.mmm], where sssssssss must be an integer number representing seconds between 0 and 4294967295, inclusive and mmm represents milliseconds and must be an integer number between 0 to 999).

Result String

None

Example:

trigger occurs 1 period-start "0 8 * * 1-5" period 720

Embedded Event Manager Action Tcl Command Extensions

action_process

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

Syntax

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

Arguments

start	(Mandatory) Specifies that a process is to be started.
restart	(Mandatory) Specifies that a process is to be restarted.
kill	(Mandatory) Specifies that a process is to be stopped (killed).

job_id	(Optional) System manager assigned job ID for the process. If you specify this argument, it must be an integer between 1 and 4294967295, inclusive.
process_name	(Optional) Process name. Either job_id must be specified or process_name and instance must be specified.
instance	(Optional) Process instance ID. If you specify this argument, it must be an integer between 1 and 4294967295, inclusive.

Result String

None

Set _cerrno

Yes

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

This error means that the action command requested was unknown.

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

This error means that the arguments passed in were invalid.

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

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

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

This error means that the process name was not known.

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

This error means that an object size exceeded its maximum.

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

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

action_program

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

Syntax

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

Arguments

path	(Mandatory) Pathname of a program to run.
argv	(Optional) Argument string of the program.
envp	(Optional) Environment string of the program.
stdin	(Optional) Pathname for stdin.
stdout	(Optional) Pathname for stdout.
stderr	(Optional) Pathname for stderr.

Result String

None

Set _cerrno

Yes

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

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

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

This error means that the action command requested was unknown.

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

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

action_script

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

Syntax

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

Arguments

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

Result String

None

Set _cerrno

Yes

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

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

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

This error means that the action command requested was unknown.

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

This error means that a configuration error has occurred.

action_setver_prior

Revert the process identified by the absolute path to the prior version.

Syntax

```
action_setver_prior [path ?]
```

Arguments

path	(Mandatory) The process executable path.

Result String

None

Set _cerrno

Yes

action_setnode

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

- action_process
- sys_reqinfo_proc
- sys_reqinfo_proc_all

- sys_reqinfo_crash_history
- sys_reqinfo_proc_version

Syntax

```
action_setnode [node ?]
```

Arguments

node	(Mandatory) Name of the node.

Result String

None

Set _cerrno

Yes

action_syslog

Logs a message.

Syntax

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

Arguments

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

Result String

None

Set _cerrno

Yes

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

This error means that the action command requested was unknown.

action_track_read

Reads the state of a tracked object when an Embedded Event Manager (EEM) script is triggered.

Syntax

```
action_track_read ?
```

Arguments

? (represents a string)	(Mandatory) Tracked object name.

Result String

```
name {%s}
state {%s}
```

Set _cerrno

Yes

```
FH_ENOTRACK
```

This error means that the tracked object name was not found.

Embedded Event Manager Utility Tcl Command Extensions

appl_read

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

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

Syntax

```
appl_read name ? length ?
```

Arguments

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

Result String

```
data %s
```

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

Set _cerrno

Yes

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

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

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

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

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

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

appl_reqinfo

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

Syntax

```
appl_reqinfo key ?
```

Arguments

key	(Mandatory) String key of the data.

Result String

```
data %s
```

Where data is the application string data to be retrieved.

Set _cerrno

Yes

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

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

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

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

appl_setinfo

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

Syntax

```
appl_setinfo key ? data ?
```

Arguments

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

Result String

None

Set _cerrno

Yes

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

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

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

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

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

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

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

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

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

This error means that the API message length was invalid.

counter_modify

Modifies a counter value.

Syntax

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

Arguments

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

Result String

```
val_remain %d
```

Where val_remain is the current value of the counter.

Set _cerrno

Yes

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

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

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

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

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

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

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

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

fts_get_stamp

Returns the time period elapsed since the last software boot. Use this Tcl command extension to return the number of nanoseconds since boot in an array “nsec nnnn” where nnnn is the number of nanoseconds.

Syntax

```
fts_get_stamp
```

Arguments

None

Result String

```
nsec %d
```

Where nsec is the number of nanoseconds since boot.

Set _cerrno

No

register_counter

Registers a counter and returns a counter event ID. This Tcl command extension is used by a counter publisher to perform this registration before using the event ID to manipulate the counter.

Syntax

```
register_counter name ?
```

Arguments

name	(Mandatory) The name of the counter to be manipulated.

Result String

```
event_id %d
event_spec_id %d
```

Where event_id is the counter event ID for the specified counter; it can be used to manipulate the counter by the **unregister_counter** or **counter_modify** Tcl command extensions. The event_spec_id argument is the event specification ID for the specified counter.

Set _cerrno

Yes

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

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

```
(_cerr_sub_err =  4)    FH_EINITONCE  (Init() is not yet done, or done twice.)
```

This error means that the request to register the specific event was made before the EEM event detector had completed its initialization.

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

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

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

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

```
(_cerr_sub_err = 10)    FH_ECORRUPT  (internal EEM API context is corrupt)
```

This error means that the internal EEM API context structure is corrupt.

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

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

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

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

```
(_cerr_sub_err = 16)    FH_EBADFMPPTR  (bad ptr to fh_p data structure)
```

This error means that the context pointer that is used with each EEM API call is incorrect.

```
(_cerr_sub_err = 17)    FH_EBADADDRESS  (bad API control block address)
```

This error means that a control block address that was passed in the EEM API was incorrect.

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

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

```
(_cerr_sub_err = 25)    FH_ESUBSEXCEED  (number of subscribers exceeded)
```

This error means that the number of timer or counter subscribers exceeded the maximum.

```
(_cerr_sub_err = 26)    FH_ESUBSIDXINV  (invalid subscriber index)
```

This error means that the subscriber index was invalid.

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

This error means that the event detector was unavailable.

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

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

register_timer

Registers a timer and returns a timer event ID. This Tcl command extension is used by a timer publisher to perform this registration before using the event ID to manipulate the timer if it does not use the **event_register_timer** command extension to register as a publisher and subscriber.

Syntax

```
register_timer watchdog|countdown|absolute|cron name ?
```

Arguments

name	(Mandatory) Name of the timer to be manipulated.

Result String

```
event_id %u
```

Where event_id is the timer event ID for the specified timer (can be used to manipulate the timer by the **timer_arm** or **timer_cancel** command extensions).

Set _cerrno

Yes

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

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

```
(_cerr_sub_err =  4)    FH_EINITONCE  (Init() is not yet done, or done twice.)
```

This error means that the request to register the specific event was made before the EEM event detector had completed its initialization.

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

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

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

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

```
(_cerr_sub_err = 10)    FH_ECORRUPT  (internal EEM API context is corrupt)
```

This error means that the internal EEM API context structure is corrupt.

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

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

```
(_cerr_sub_err = 16)    FH_EBADFMPPTR  (bad ptr to fh_p data structure)
```

This error means that the context pointer that is used with each EEM API call is incorrect.

```
(_cerr_sub_err = 17)    FH_EBADADDRESS  (bad API control block address)
```

This error means that a control block address that was passed in the EEM API was incorrect.

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

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

```
(_cerr_sub_err = 25)    FH_ESUBSEXCEED  (number of subscribers exceeded)
```

This error means that the number of timer or counter subscribers exceeded the maximum.

```
(_cerr_sub_err = 26)    FH_ESUBSIDXINV  (invalid subscriber index)
```

This error means that the subscriber index was invalid.

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

This error means that the event detector was unavailable.

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

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

timer_arm

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

Syntax

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

Arguments

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

Result String

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

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

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

Set _cerrno

Yes

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

This error means that the event was not registered.

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

This error means that the event detector was unavailable.

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

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

timer_cancel

Cancels a timer.

Syntax

```
timer_cancel event_id ?
```

Arguments

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

Result String

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

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

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

Set _cerrno

Yes

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

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

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

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

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

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

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

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

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

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

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

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

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

This error means that the event detector was unavailable.

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

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

unregister_counter

Unregisters a counter. This Tcl command extension is used by a counter publisher to unregister a counter that was previously registered with the **register_counter** Tcl command extension.

Syntax

```
unregister_counter event_id ? event_spec_id ?
```

Arguments

event_id	(Mandatory) Counter event ID returned by the **register_counter** command extension. Must be an integer between 0 and 4294967295, inclusive.
event_spec_id	(Mandatory) Counter event specification ID for the specified counter returned by the **register_counter** command extension. Must be an integer between 0 and 4294967295, inclusive.

Result String

None

Set _cerrno

Yes

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

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

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

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

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

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

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

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

```
(_cerr_sub_err = 26)    FH_ESUBSIDXINV  (invalid subscriber index)
```

This error means that the subscriber index was invalid.

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

This error means that the event detector was unavailable.

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

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

Embedded Event Manager System Information Tcl Command Extensions

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

sys_reqinfo_cpu_all

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

Syntax

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

Arguments

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

Result String

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

Where each process CPU info string is:

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

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

Set _cerrno

Yes

sys_reqinfo_crash_history

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

Syntax

```
sys_reqinfo_crash_history
```

Arguments

None

Result String

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

Where each crash info string is:

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

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

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

Set _cerrno

Yes

sys_reqinfo_mem_all

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

Syntax

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

Arguments

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

Result String

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

Where each process mem info string is:

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

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

Set _cerrno

Yes

sys_reqinfo_proc

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

Syntax

```
sys_reqinfo_proc job_id ?
```

Arguments

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

Result String

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

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

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

Set _cerrno

Yes

sys_reqinfo_proc_all

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

Syntax

```
sys_reqinfo_proc_all
```

Arguments

None

Result String

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

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

Set _cerrno

Yes

sys_reqinfo_proc_version

Queries the version of the given process.

Syntax

```
sys_reqinfo_proc_version [job_id ?]
```

Arguments

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

Result String

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

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

Set _cerrno

Yes

sys_reqinfo_routername

Queries the router name.

Syntax

```
sys_reqinfo_routername
```

Arguments

None

Result String

```
routername %s
```

Where routername is the name of the router.

Set _cerrno

Yes

sys_reqinfo_syslog_freq

Queries the frequency information of all syslog events.

Syntax

```
sys_reqinfo_syslog_freq
```

Arguments

None

Result String

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

Where each event frequency string is:

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

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

pattern	Regular expression used to perform syslog message pattern matching.

Set _cerrno

Yes

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

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

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

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

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

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

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

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

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

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

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

This error means that the event detector was unavailable.

sys_reqinfo_syslog_history

Queries the history of the specified syslog message.

Syntax

```
sys_reqinfo_syslog_history
```

Arguments

None

Result String

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

Where each log hist string is:

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

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

Set _cerrno

Yes

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

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

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

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

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

This error means that the history list was empty.

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

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

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

This error means that the event detector was unavailable.

sys_reqinfo_stat

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

Syntax

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

Arguments

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

mod_2	(Optional) Statistics data element modifier 2.

Result String

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

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

Set _cerrno

Yes

sys_reqinfo_snmp

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

Syntax

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

Arguments

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

Result String

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

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

sys_reqinfo_snmp_trap

This command is used to send a trap.

Syntax

```
sys_reqinfo_snmp_trap enterprise_oid ent-oid generic_trapnum gen-trapnum specific_trapnum
spe-trapnum
 trap_oid oid trap_var varname
```

- Use the ***enterprise_oid*** argument to specify the enterprise oid of the trap.
- Use the ***generic_trapnum*** argument to specify generic trap number of the trap.

- Use the ***specific_trapnum*** argument to specify specific trap number of the trap.
- Use the ***trap_oid*** argument to specify oid of the trap to send.
- Use the ***trap_var*** argument to specify the variable of oid(s) to send.

Example

sys_reqinfo_snmp_trap **enterprise_oid** 1.3.6.1.4.1.9.9.41.2 **generic_trapnum** 6 **specific_trapnum** 1 **trap_oid** 1.3.6.1.4.1.9.9.41.2.0.1 **trap_var** var1

sys_reqinfo_snmp_trapvar

This command is used to setup an array of oid and value given a trap variable. Similar to IOS, the trap variable can contain a list of 10 multiple oids and values.

Syntax

```
sys_reqinfo_snmp_trapvar var varname oid oid int|uint|counter|gauge|octet|string|ipv4  value
```

- Use the ***var*** argument to specify the trap variable name.
- Use the ***oid*** argument to specify the oid of the trap.

Example

sys_reqinfo_snmp_trapvar **var** var1 **oid** 1.3.6.1.4.1.9.9.41.1.2.3.1.3 **int** 4

SMTP Library Command Extensions

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

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

E-Mail Template

The e-mail template file has the following format:

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

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

The following is a sample e-mail template file:

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

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

Exported Tcl Command Extensions

smtp_send_email

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

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

Syntax

```
smtp_send_email text
```

Arguments

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

Result String

None

Set _cerrno

- Wrong 1st line format—Mailservername:list of server names.
- Wrong 2nd line format—From:from-address.
- Wrong 3rd line format—To:list of to-addresses.
- Wrong 4th line format—CC:list of cc-addresses.

- Error connecting to mail server:—$sock closed by remote server (where $sock is the name of the socket opened to the mail server).
- Error connecting to mail server:—$sock reply code is $k instead of the service ready greeting (where $sock is the name of the socket opened to the mail server; $k is the reply code of $sock).
- Error connecting to mail server:—cannot connect to all the candidate mail servers.
- Error disconnecting from mail server:—$sock closed by remote server (where $sock is the name of the socket opened to the mail server).

Sample Scripts

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

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

smtp_subst

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

Syntax

```
smtp_subst e-mail_template
```

Arguments

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

Result String

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

Set _cerrno

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

CLI Library Command Extensions

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

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

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

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

For noninteractive commands, once the command is entered, the command will run to completion. To run different types of commands using an EEM script, different CLI library command sequences should be used, which are documented in the Using the CLI Library to Run a Noninteractive Command, on page 174 and in the Using the CLI Library to Run an Interactive Command, on page 175.

Exported Tcl Command Extensions

cli_close

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

Syntax

```
cli_close fd tty_id
```

Arguments

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

Result String

None

Set _cerrno

Cannot close the channel.

cli_exec

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

Syntax

```
cli_exec fd cmd
```

Arguments

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

Result String

The output of the CLI command executed.

Set _cerrno

Error reading the channel.

cli_get_ttyname

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

Syntax

```
cli_get_ttyname tty_id
```

Arguments

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

Result String

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

Set _cerrno

None

cli_open

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

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

Syntax

```
cli_open
```

Arguments

None

Result String

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

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

Set _cerrno

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

cli_read

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

Syntax

```
cli_read fd
```

Arguments

fd	(Mandatory) CLI channel handler.

Result String

All the contents read.

Set _cerrno

Cannot get router name.

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

cli_read_drain

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

Syntax

```
cli_read_drain fd
```

Arguments

fd	(Mandatory) The CLI channel handler.

Result String

All the contents read.

Set _cerrno

None

cli_read_line

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

Syntax

```
cli_read_line fd
```

Arguments

fd	(Mandatory) CLI channel handler.

Result String

The line read.

Set _cerrno

None

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

cli_read_pattern

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

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

Syntax

```
cli_read_pattern fd ptn
```

Arguments

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

Result String

All the contents read.

Set _cerrno

None

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

cli_write

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

Syntax

```
cli_write fd cmd
```

Arguments

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

Result String

None

Set _cerrno

None

Sample Usage

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

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

Using the CLI Library to Run a Noninteractive Command

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

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

Using the CLI Library to Run an Interactive Command

To run interactive commands, three phases are needed:

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

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

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

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

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

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

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

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

Tcl Context Library Command Extensions

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

Exported Commands

context_retrieve

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

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

Syntax

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

Arguments

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

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

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

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

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

Result String

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

Set _cerrno

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

Sample Usage

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

Example 1: Save

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

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

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

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

```
} else {
      action_syslog msg "context_save succeeded"
}
```

Example 1: Retrieve

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

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

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

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

Example 2: Save

If var is specified, saves the value of var.

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

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

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

Example 2: Retrieve

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

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

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

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

Example 3: Save

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

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

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

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

Example 3: Retrieve

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

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

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

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

Example 4: Save

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

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

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

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

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

Example 4: Retrieve

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

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

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

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

context_save

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

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

Syntax

```
context_save ctxt [pattern]
```

Arguments

ctxt	(Mandatory) Context name.

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

Result String

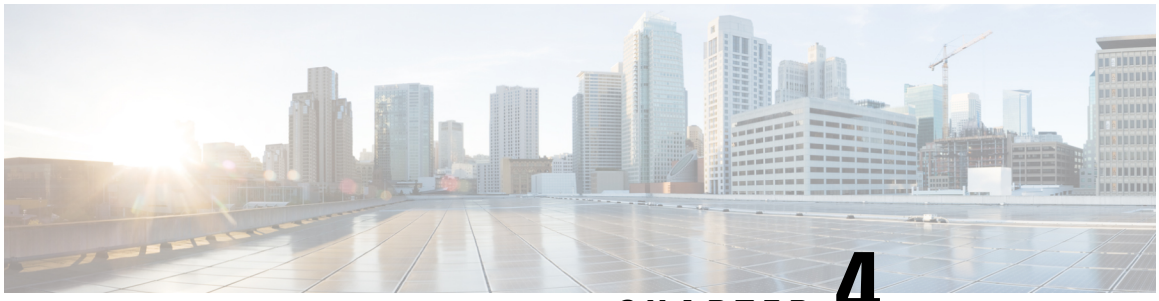
None

Set _cerrno

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

Sample Usage

For examples showing how to use the **context_save** and **context_retrieve** command extension functionality to save and retrieve data, see the Sample Usage, on page 177.



CHAPTER 4

Implementing IP Service Level Agreements

IP Service Level Agreements (IP SLAs) is a portfolio of technology embedded in most devices that run Cisco IOS XR Software, which allows you to analyze IP service levels for IP applications and services, increase productivity, lower operational costs, and reduce the frequency of network outages.

Using IP SLA, service provider customers can measure and provide service level agreements. IP SLA can perform network assessments, verify quality of service (QoS), ease the deployment of new services, and assist administrators with network troubleshooting.

Note For a complete description of the IP SLA commands used in this chapter, refer to the *IP Service Level Agreement Commands on the Cisco ASR 9000 Series Router* module of *System Management Command Reference for Cisco ASR 9000 Series Routers*.

Feature History for Implementing IP Service Level Agreements

Release	Modification
Release 3.7.2	This feature was introduced.
Release 6.0.1	The TWAMP accuracy enhancement was introduced.

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 configurable operations that is supported by Cisco IOS XR Software is 2000.
- The current validated scale numbers for scheduling operations are as follows:
  - The number of UDP echo operations is 1000 operations with default frequency.
  - The number of UDP jitter operations is 1000 operations with default frequency.
  - The number of ICMP echo operations is 1000 operations with default frequency.
  - The number of ICMP echo-path operations is 400 operations with default frequency.
  - The ICMP jitter operations that can be configured with default frequency without packet loss is 75.
  - The MPLS LSP ping operations that can be configured with default frequency without packet loss is 100.
  - The MPLS LSP trace operations that can be configured with default frequency without packet loss is 100.
- 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.
  - For the UDP and ICMP jitter operation, the following guidelines are recommended:
    - frequency > timeout + 2 seconds + num_packets * packet_interval
    - timeout > rtt_threshold
    - num_packet > loss_threshold
  - For all other IP SLAs operations, the following configuration guideline is recommended:
    - frequency > timeout > rtt_threshold

Information About Implementing IP Service Level Agreements

About IP Service Level Agreements Technology

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. This information is collected:

- Response times
- One-way latency, jitter (interpacket delay variance)
- Packet loss
- Network resource availability

IP SLA originated from the technology previously known as Service Assurance Agent (SAA). 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.

Depending on the specific IP SLA operation, statistics of delay, packet loss, jitter, packet sequence, connectivity, and path are monitored by and stored in the router and provided through command-line interface (CLI), Extensive Markup Language (XML), and SNMP MIBs. IP SLA uses the Cisco RTTMON Management Information Base (MIB) to interact between external Network Management System (NMS) applications and the IP SLA operations that are running on Cisco devices.

For a complete description of the object variables that are referenced by IP SLA, see the text of the CISCO-RTTMON-MIB.my file that is available from the Cisco MIB Locator.

Service Level Agreements

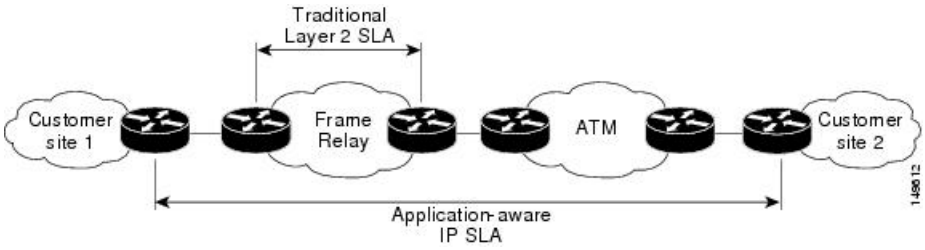
Internet commerce has grown significantly in the past few years as the technology has advanced to provide faster, more reliable access to the Internet. Many companies need online access and conduct most of their business on line and any loss of service can affect the profitability of the company. Internet 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 3: Scope of Traditional Service Level Agreement Versus IP SLA, on page 186 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 3: Scope of Traditional Service Level Agreement Versus IP SLA



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

Table 19: 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 20: List of Benefits for IP SLA

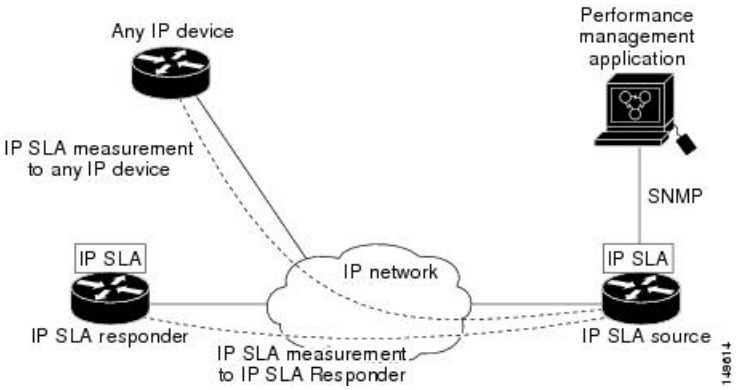
Benefit	Description
IP SLA monitoring	Provides service level agreement monitoring, measurement, and verification.

Benefit	Description
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.

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 4: IP SLA Operations, on page 187 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 4: IP SLA Operations



Operations are divided into two classes, which depend on whether they rely on the IP SLA Responder component to be running at the target device or not. The former is used only with Cisco devices; whereas, the latter is used with any device that has IP connectivity. Operations that are based on Internet Control Message Protocol (ICMP) are examples of the second class; whereas, UDP-based operations are examples of the first.

In responder-based operations, the IP SLA Responder is enabled in the destination device and provides information such as the processing delays of IP SLA packets. The responder-based operation has improved accuracy over the ICMP operation discussed above, and offers the capability of unidirectional measurements. In replies to the IP SLA source device, the responder includes information about processing delays. The IP SLA source device removes the delays in its final performance calculation. Use of the responder is optional for the UDP echo operation, but it is required for the UDP jitter operation. If no IP SLA Responder is used, the target device should support the UDP echo operation.

In ICMP operations, the source IP SLA device sends several ICMP packets to the destination. The destination device, which is any IP device, echoes with replies. The source IP SLA device uses the sent and received time stamps to calculate the response time. The ICMP echo operation resembles the traditional extended ping

utility, and it measures only the response time between the source device and the destination device. ICMP path-echo and path-jitter operations use the traceroute mechanism to identify the whole path. Subsequent ICMP packets are sent to each path node, and the measurements are correlated to provide hop-by-hop round-trip delay and jitter information.

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.

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 21: 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.

Operation	Description
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 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. Additional statistics are also provided, which are not otherwise available through standard ICMP-based 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 4: IP SLA Operations, on page 187 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, but it is optional for UDP echo 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.

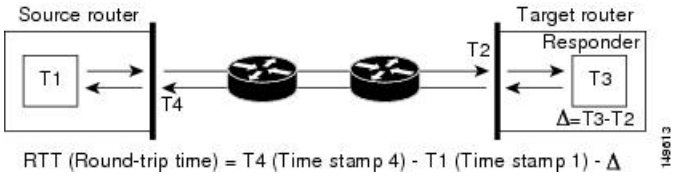
Response Time Computation for IP SLA

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. 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. At times of high network activity, an ICMP ping test often shows a long and inaccurate response time, while an IP SLA-based responder shows an accurate response time.

Figure 4: IP SLA Operations, on page 187 shows how the responder works. 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 5: IP SLA Responder Time Stamping



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 21: Types of Operations for IP SLA, on page 188 describes the different IP SLA operations, including information about whether or not an operation supports the use of nondefault VRF tables.

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.

Note Multiple SLA probes with the same configuration (source and port number) must not be scheduled to run simultaneously.

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.

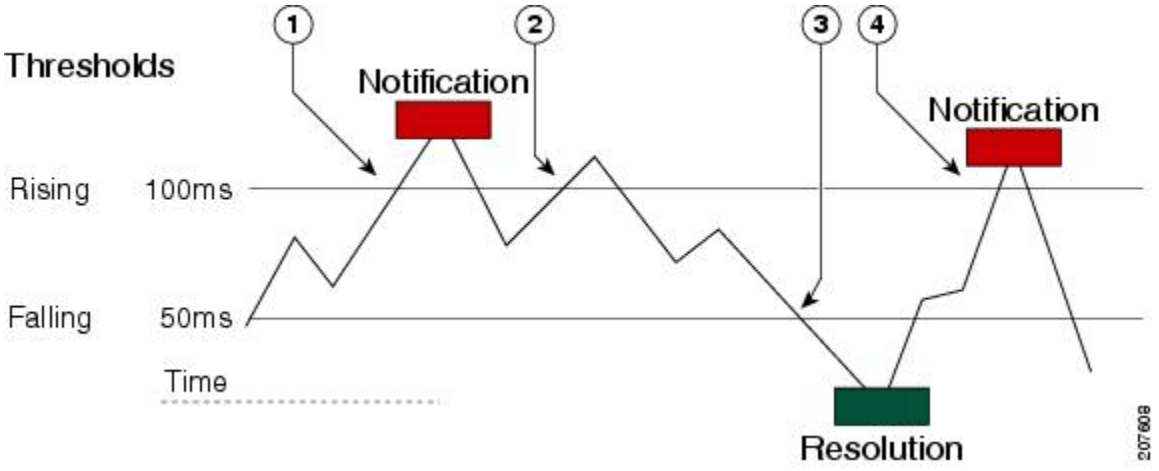
IP SLA Threshold Monitoring and Notifications

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 6: 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.

Advantages of TWAMP

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

The 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 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. The sequence of commands is as follows, but 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.

- 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.

TWAMP Accuracy Enhancement

The TWAMP (Two-Way Active Measurement Protocol) accuracy enhancement provides microsecond granularity in TWAMP measurements. This enhancement allows the collection of ingress and egress time stamps as closely as possible to the wire to achieve more accuracy. The granularity also depends on the synchronization mechanism used. The TWAMP accuracy enhancement uses the NTP RP (Network Time Protocol Route Processor) to LC (Line Card) synchronization.

Hardware Time Stamp

The hardware time stamp feature provides greater accuracy than other time synchronization protocols. This feature is enabled by default and requires no configuration.

The objective of using the LC clock for hardware time stamp is to provide timing signals to the connected servers, so that the system clocks can be synchronized accurately.

Benefits of Hardware Time Stamping

Hardware time stamping achieves microsecond precision, better accuracy (closer to wire) and better performance at scale.

Hardware Time Stamp Disable

Some platforms may rely on a certain configuration or deployment to provide hardware time stamping. In particular, the Cisco ASR9000 series routers need PTP synchronization as a clock source. Such a solution may not be available in all user scenarios. To allow the use of other sources of time stamping (NTP clock source, through a daemon running on RP) a new configuration - **ipsla hw-timestamp disable** is introduced to ignore the time stamp values provided by other platform dependent layers and revert back to platform independent time stamps.

Use the **hw-timestamp disable** command in IP SLA configuration to disable hardware time stamping.

Limitations

The TWAMP accuracy enhancement has the following limitations:

- It may be required to configure PTP or GPS interface to provide a clock source for hardware time stamping

- TWAMP reflector does not support IPv6
- TWAMP does not support SNMP or XML
- TWAMP does not support VRF (VPN Routing and Forwarding)

Recommendations

TWAMP Sender must not use port values already assigned by IANA organization for other applications. See http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml.

To avoid conflict and consequently unexpected behavior, it is highly recommended for TWAMP Sender applications to use port values in the range of the Dynamic Ports (also known as the Private or Ephemeral Ports) from 49152 to 65535 (never assigned by IANA, as per RFC 6335).

One-Way Delay Measurement

The UDP (User Datagram Protocol) jitter operation is capable of measuring one-way delay for packets. The NTP synchronization of the LC hardware clock to the RP clock supports one-way delay measurement. One-way delay measurement is also possible with PTP and GPS synchronization.

Note The round-trip time (RTT) and jitter measurements are validated with both NTP and PTP clock synchronizations.

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 195.

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.

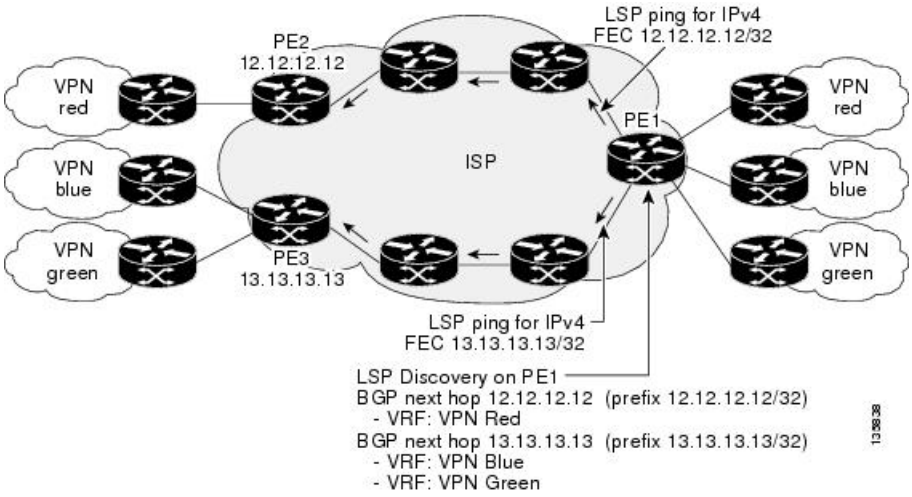
BGP Next-hop Neighbor Discovery

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 7: BGP Next-hop Neighbor Discovery for a Simple VPN, on page 196 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 7: 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 257 and the Configuring an MPLS LSP Monitoring Trace Instance, on page 261.

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.

SUMMARY STEPS

1. **configure**
2. **ipsla responder**
3. **type udp ipv4 address** *ip-address* **port** *port*
4. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.

	Command or Action	Purpose

Step 2	**ipsla responder**
	Example:
	`RP/0/RSP0/CPU0:router(config)# ipsla responder`
	`RP/0/RSP0/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/RSP0/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 199 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 22: UDP Jitter Operation Parameters, on page 199.

Table 22: 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

UDP Jitter Operation Parameter	Default	Configured Using
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 198 section.

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

You can configure and schedule a UDP jitter operation.

SUMMARY STEPS

1. **configure**
2. **ipsla operation** *operation-number*
3. **type udp jitter**
4. **destination address** *ipv4address*
5. **destination port** *port*
6. **packet count** *count*
7. **packet interval** *interval*
8. **frequency** *seconds*
9. **exit**
10. **ipsla schedule operation** *op-num*

11. **life** { **forever** | *seconds*}
12. **ageout** *seconds*
13. **recurring**
14. **start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*]
15. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**ipsla operation** *operation-number* **Example:** `RP/0/RSP0/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/RSP0/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/RSP0/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/RSP0/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/RSP0/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/RSP0/CPU0:router(config-ipsla-udp-jitter)# packet interval 30`	(Optional) Specifies the time between packets. The default interval between packets is 20 milliseconds.

	Command or Action	Purpose
Step 8	**frequency** *seconds* **Example:** `RP/0/RSP0/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/RSP0/CPU0:router(config-ipsla-udp-jitter)# exit`
	`RP/0/RSP0/CPU0:router(config-ipsla-op)# exit`
	`RP/0/RSP0/CPU0:router(config-ipsla)# exit`
	`RP/0/RSP0/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/RSP0/CPU0:router(config)# ipsla schedule operation 432`
	`RP/0/RSP0/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/RSP0/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/RSP0/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/RSP0/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/RSP0/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.

	Command or Action	Purpose
		• (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.

SUMMARY STEPS

1. **configure**
2. **ipsla operation** *operation-number*
3. **type udp jitter**
4. **vrf** *vrf-name*
5. **destination address** *ipv4address*
6. **destination port** *port*
7. **frequency** *seconds*
8. **statistics** [**hourly** | **interval** *seconds*]
9. **buckets** *hours*
10. **distribution count** *slot*
11. **distribution interval** *interval*
12. **exit**
13. **datasize request** *size*
14. **timeout** *milliseconds*
15. **tos** *number*
16. **exit**
17. **ipsla schedule operation** *op-num*
18. **life** {**forever** | *seconds*}
19. **ageout** *seconds*
20. **recurring**
21. **start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*]

22. Use the **commit** or **end** command.
23. **show ipsla statistics** [*operation-number*]
24. **show ipsla statistics aggregated** [*operation-number*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**ipsla operation** *operation-number* **Example:** `RP/0/RSP0/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/RSP0/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/RSP0/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/RSP0/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/RSP0/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/RSP0/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.

	Command or Action	Purpose

Step 8	**statistics** [**hourly** | **interval** *seconds*]
	Example:
	`RP/0/RSP0/CPU0:router(config-ipsla-udp-jitter)# statistics hourly`
	`RP/0/RSP0/CPU0:router(config-ipsla-op-stats)#`
		(Optional) Specifies the statistics collection parameters for UDP jitter operation.

Step 9	**buckets** *hours* **Example:** `RP/0/RSP0/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/RSP0/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/RSP0/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/RSP0/CPU0:router(config-ipsla-op-stats)# exit`	Exits from IP SLA statistics configuration mode.
Step 13	**datasize request** *size* **Example:** `RP/0/RSP0/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/RSP0/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/RSP0/CPU0:router(config-ipsla-udp-jitter)# tos 255`	Specifies the type of service number.

	Command or Action	Purpose

Step 16	**exit**
	Example:
	`RP/0/RSP0/CPU0:router(config-ipsla-udp-jitter)# exit`
	`RP/0/RSP0/CPU0:router(config-ipsla-op)# exit`
	`RP/0/RSP0/CPU0:router(config-ipsla)# exit`
	`RP/0/RSP0/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/RSP0/CPU0:router(config)# ipsla schedule operation 432`
	`RP/0/RSP0/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/RSP0/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/RSP0/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/RSP0/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/RSP0/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.

	Command or Action	Purpose
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/RSP0/CPU0:router # show ipsla statistics 432`	Displays the current statistics.
Step 24	**show ipsla statistics aggregated** [*operation-number*] **Example:** `RP/0/RSP0/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 198 section.

Configuring and Scheduling a UDP Echo Operation on the Source Device

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

SUMMARY STEPS

1. **configure**
2. **ipsla operation** *operation-number*
3. **type udp echo**
4. **destination address** *ipv4address*
5. **destination port** *port*
6. **frequency** *seconds*
7. **exit**
8. **ipsla schedule operation** *op-num*
9. **life** [**forever** | *seconds*]
10. **ageout** *seconds*
11. **recurring**
12. **start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*]
13. Use the **commit** or **end** command.
14. **show ipsla statistics** [*operation-number*]
15. **show ipsla statistics aggregated** [*operation-number*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**ipsla operation** *operation-number* **Example:** `RP/0/RSP0/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/RSP0/CPU0:router(config-ipsla-op)# type udp echo`	Configures the operation as a UDP echo operation, and configures characteristics for the operation.

	Command or Action	Purpose
Step 4	**destination address** *ipv4address* **Example:** `RP/0/RSP0/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/RSP0/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/RSP0/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/RSP0/CPU0:router(config-ipsla-udp-echo)# exit`
	`RP/0/RSP0/CPU0:router(config-ipsla-op)# exit`
	`RP/0/RSP0/CPU0:router(config-ipsla)# exit`
	`RP/0/RSP0/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/RSP0/CPU0:router(config)# ipsla schedule operation 432`
	`RP/0/RSP0/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/RSP0/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/RSP0/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.

	Command or Action	Purpose
Step 11	**recurring** **Example:** `RP/0/RSP0/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/RSP0/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/RSP0/CPU0:router# show ipsla statistics 432`	Displays the current statistics.
Step 15	**show ipsla statistics aggregated** [*operation-number*] **Example:** `RP/0/RSP0/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.

SUMMARY STEPS

1. **configure**
2. **ipsla operation** *operation-number*
3. **type udp echo**
4. **vrf** *vrf-name*
5. **destination address** *ipv4address*
6. **destination port** *port*
7. **frequency** *seconds*
8. **datasize request** *size*
9. **tos** *number*
10. **timeout** *milliseconds*
11. **tag** *text*
12. **exit**
13. **ipsla schedule operation** *op-num*
14. **life** {**forever** | *seconds*}
15. **ageout** *seconds*
16. **recurring**
17. **start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*]
18. Use the **commit** or **end** command.
19. **show ipsla statistics enhanced aggregated** [*operation-number*] **interval** *seconds*
20. **show ipsla statistics** [*operation-number*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**ipsla operation** *operation-number* **Example:** `RP/0/RSP0/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/RSP0/CPU0:router(config-ipsla-op)# type udp echo`	Configures the operation as a UDP echo operation, and configures characteristics for the operation.

	Command or Action	Purpose
Step 4	**vrf** *vrf-name* **Example:** `RP/0/RSP0/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/RSP0/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/RSP0/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/RSP0/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/RSP0/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/RSP0/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/RSP0/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.

	Command or Action	Purpose
Step 11	**tag** *text* **Example:** `RP/0/RSP0/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/RSP0/CPU0:router(config-ipsla-udp-echo)# exit`
	`RP/0/RSP0/CPU0:router(config-ipsla-op)# exit`
	`RP/0/RSP0/CPU0:router(config-ipsla)# exit`
	`RP/0/RSP0/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/RSP0/CPU0:router(config)# ipsla schedule operation 432`
	`RP/0/RSP0/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/RSP0/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/RSP0/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/RSP0/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/RSP0/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.

	Command or Action	Purpose
		• (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/RSP0/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/RSP0/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.

SUMMARY STEPS

1. **configure**

2. **ipsla operation** *operation-number*
3. **type icmp echo**
4. **destination address** *ipv4address*
5. **frequency** *seconds*
6. **exit**
7. **ipsla schedule operation** *op-num*
8. **life** {**forever** | *seconds*}
9. **ageout** *seconds*
10. **recurring**
11. **start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*]
12. Use the **commit** or **end** command.
13. **show ipsla statistics** [*operation-number*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**ipsla operation** *operation-number* **Example:** `RP/0/RSP0/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/RSP0/CPU0:router(config-ipsla-op)# type icmp echo`	Defines an ICMP echo operation type.
Step 4	**destination address** *ipv4address* **Example:** `RP/0/RSP0/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/RSP0/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.

	Command or Action	Purpose

Step 6	**exit**
	Example:
	`RP/0/RSP0/CPU0:router(config-ipsla-icmp-echo)# exit`
	`RP/0/RSP0/CPU0:router(config-ipsla-op)# exit`
	`RP/0/RSP0/CPU0:router(config-ipsla)# exit`
	`RP/0/RSP0/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/RSP0/CPU0:router(config)# ipsla schedule operation 432`
	`RP/0/RSP0/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/RSP0/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/RSP0/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/RSP0/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/RSP0/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.

	Command or Action	Purpose
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/RSP0/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.

SUMMARY STEPS

1. **configure**
2. **ipsla operation** *operation-number*
3. **type icmp echo**
4. **vrf** *vrf-name*
5. **destination address** *ipv4address*
6. **frequency** *seconds*
7. **datasize request** *size*
8. **tos** *number*
9. **timeout** *milliseconds*
10. **tag** *text*
11. exit
12. **ipsla schedule operation** *op-num*
13. **life** {**forever** | *seconds*}
14. **ageout** *seconds*
15. **recurring**
16. **start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*]
17. Use the **commit** or **end** command.
18. **show ipsla statistics** [*operation-number*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**ipsla operation** *operation-number* **Example:** `RP/0/RSP0/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/RSP0/CPU0:router(config-ipsla-op)# type icmp echo`	Defines an ICMP echo operation type.
Step 4	**vrf** *vrf-name* **Example:** `RP/0/RSP0/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/RSP0/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/RSP0/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/RSP0/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:**	Defines a type of service (ToS) byte in the IP header of IP SLA operations.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-ipsla-icmp-echo)# tos 1`	**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/RSP0/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/RSP0/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/RSP0/CPU0:router(config-ipsla-icmp-echo)# exit`
	`RP/0/RSP0/CPU0:router(config-ipsla-op)# exit`
	`RP/0/RSP0/CPU0:router(config-ipsla)# exit`
	`RP/0/RSP0/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/RSP0/CPU0:router(config)# ipsla schedule operation 432`
	`RP/0/RSP0/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/RSP0/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/RSP0/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.

	Command or Action	Purpose
Step 15	**recurring** **Example:** `RP/0/RSP0/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/RSP0/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/RSP0/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.

SUMMARY STEPS

1. **configure**
2. **ipsla operation** *operation-number*
3. **type icmp path-echo**
4. **destination address** *ipv4address*
5. **frequency** *seconds*
6. **exit**
7. **ipsla schedule operation** *op-num*
8. **life** {**forever** | *seconds*}
9. **ageout** *seconds*
10. **recurring**
11. **start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*]
12. Use the **commit** or **end** command.
13. **show ipsla statistics** [*operation-number*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**ipsla operation** *operation-number* **Example:** `RP/0/RSP0/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/RSP0/CPU0:router(config-ipsla-op)# type icmp`	Defines an ICMP path-echo operation type.

	Command or Action	Purpose
	path-echo RP/0/RSP0/CPU0:router(config-ipsla-icmp-path-echo)#	
Step 4	**destination address** *ipv4address* **Example:** RP/0/RSP0/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/RSP0/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/RSP0/CPU0:router(config-ipsla-icmp-path-echo)# exit RP/0/RSP0/CPU0:router(config-ipsla-op)# exit RP/0/RSP0/CPU0:router(config-ipsla)# exit RP/0/RSP0/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/RSP0/CPU0:router(config)# ipsla schedule operation 432 RP/0/RSP0/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/RSP0/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/RSP0/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/RSP0/CPU0:router(config-ipsla-sched)# recurring	(Optional) Specifies that the operation starts automatically at the specified time and for the specified duration every day.

	Command or Action	Purpose
Step 11	**start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*] **Example:** `RP/0/RSP0/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/RSP0/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.

SUMMARY STEPS

1. **configure**
2. **ipsla operation** *operation-number*
3. **type icmp path-echo**
4. **vrf** *vrf-name*
5. **lsr-path** *ip-address*
6. **destination address** *ipv4address*

7. **frequency** *seconds*
8. **datasize request** *size*
9. **tos** *number*
10. **timeout** *milliseconds*
11. **tag** *text*
12. **lsr-path** *ipaddress1* {*ipaddress2* {... {*ipaddress8*}}}
13. **exit**
14. **ipsla schedule operation** *op-num*
15. **life** {**forever** | *seconds*}
16. **ageout** *seconds*
17. **recurring**
18. **start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*]
19. Use the **commit** or **end** command.
20. **show ipsla statistics** [*operation-number*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**ipsla operation** *operation-number* **Example:** `RP/0/RSP0/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/RSP0/CPU0:router(config-ipsla-op)# type icmp path-echo`
	`RP/0/RSP0/CPU0:router(config-ipsla-icmp-path-echo)#`
		Defines an ICMP path-echo operation type.

Step 4	**vrf** *vrf-name* **Example:** `RP/0/RSP0/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/RSP0/CPU0:router(config-ipsla-imcp-path-echo)# lsr-path 20.25.22.1`	Specifies that a loose source routing path is to be used.

	Command or Action	Purpose
Step 6	**destination address** *ipv4address* **Example:** `RP/0/RSP0/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/RSP0/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/RSP0/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/RSP0/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/RSP0/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/RSP0/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/RSP0/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.

	Command or Action	Purpose

Step 13	**exit**
	Example:
	`RP/0/RSP0/CPU0:router(config-ipsla-icmp-path-echo)# exit`
	`RP/0/RSP0/CPU0:router(config-ipsla-op)# exit`
	`RP/0/RSP0/CPU0:router(config-ipsla)# exit`
	`RP/0/RSP0/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/RSP0/CPU0:router(config)# ipsla schedule operation 432`
	`RP/0/RSP0/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/RSP0/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/RSP0/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/RSP0/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/RSP0/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.

	Command or Action	Purpose
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/RSP0/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 23: 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.

SUMMARY STEPS

1. **configure**
2. **ipsla operation** *operation-number*
3. **type icmp path-jitter**
4. **destination address** *ipv4address*
5. **packet count** *count*
6. **packet interval** *interval*
7. **frequency** *seconds*
8. exit
9. **ipsla schedule operation** *op-num*
10. **life** {**forever** | *seconds*}
11. **ageout** *seconds*
12. **recurring**
13. **start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*]
14. Use the **commit** or **end** command.
15. **show ipsla statistics** [*operation-number*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.

	Command or Action	Purpose
Step 2	**ipsla operation** *operation-number* **Example:** `RP/0/RSP0/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/RSP0/CPU0:router(config-ipsla-op)# type icmp path-jitter`	Defines an ICMP path-jitter operation type.
Step 4	**destination address** *ipv4address* **Example:** `RP/0/RSP0/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/RSP0/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/RSP0/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/RSP0/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/RSP0/CPU0:router(config-ipsla-icmp-path-jitter)# exit`
	`RP/0/RSP0/CPU0:router(config-ipsla-op)# exit`
	`RP/0/RSP0/CPU0:router(config-ipsla)# exit`
	`RP/0/RSP0/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:**	Schedules the start time of the operation. You can configure a basic schedule.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config)# ipsla schedule operation 432` `RP/0/RSP0/CPU0:router(config-ipsla-sched)#`	
Step 10	**life** {**forever** | *seconds*} **Example:** `RP/0/RSP0/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/RSP0/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/RSP0/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/RSP0/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.

	Command or Action	Purpose
Step 15	**show ipsla statistics** [*operation-number*] **Example:** `RP/0/RSP0/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.

SUMMARY STEPS

1. **configure**
2. **ipsla operation** *operation-number*
3. **type icmp path-jitter**
4. **vrf** *vrf-name*
5. **lsr-path** *ip-address*
6. **destination address** *ipv4address*
7. **packet count** *count*
8. **packet interval** *interval*
9. **frequency** *seconds*
10. **datasize request** *size*
11. **tos** *number*
12. **timeout** *milliseconds*
13. **tag** *text*
14. **exit**
15. **ipsla schedule operation** *op-num*
16. **life** {**forever** | *seconds*}
17. **ageout** *seconds*
18. **recurring**
19. **start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*]
20. Use the **commit** or **end** command.
21. **show ipsla statistics** [*operation-number*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**ipsla operation** *operation-number* **Example:**	Specifies the operation number. The range is from 1 to 2048.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config)# ipsla operation 432`	
Step 3	**type icmp path-jitter** **Example:** `RP/0/RSP0/CPU0:router(config-ipsla-op)# type icmp path-jitter`	Defines an ICMP path-jitter operation type.
Step 4	**vrf** *vrf-name* **Example:** `RP/0/RSP0/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/RSP0/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/RSP0/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/RSP0/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/RSP0/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/RSP0/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:**	(Optional) Sets the protocol data size in the payload of the request packet for the specified IP SLA operation.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-ipsla-icmp-path-jitter)# datasize request 512`	• 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/RSP0/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/RSP0/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/RSP0/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/RSP0/CPU0:router(config-ipsla-icmp-path-jitter)# exit`
	`RP/0/RSP0/CPU0:router(config-ipsla-op)# exit`
	`RP/0/RSP0/CPU0:router(config-ipsla)# exit`
	`RP/0/RSP0/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/RSP0/CPU0:router(config)# ipsla schedule operation 432`
	`RP/0/RSP0/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/RSP0/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:**	(Optional) Specifies the number of seconds to keep the operation in memory when it is not actively collecting

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-ipsla-sched)# ageout 3600`	information. The default value of 0 seconds means that the operation never times out.
Step 18	**recurring** **Example:** `RP/0/RSP0/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/RSP0/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/RSP0/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)

SUMMARY STEPS

1. **configure**
2. **ipsla operation** *operation-number*
3. **type mpls lsp ping**
4. **output interface** *type interface-path-id*
5. **target** {**ipv4** *destination-address destination-mask* | **traffic-eng tunnel** *tunnel-interface* | **pseudowire** *destination-address circuit-id*}
6. **lsp selector ipv4** *ip-address*
7. **force explicit-null**
8. **reply dscp** *dscp-bits*
9. **reply mode** {**control-channel** | **router-alert**}
10. **exp** *exp-bits*
11. **ttl** *time-to-live*
12. **exit**
13. **ipsla schedule operation** *operation-number*
14. **start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*]
15. Use the **commit** or **end** command.
16. **show ipsla statistics** [*operation-number*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**ipsla operation** *operation-number* **Example:** `RP/0/RSP0/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/RSP0/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/RSP0/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/RSP0/CPU0:router(config-ipsla-mpls-lsp-ping)# target ipv4 10.25.26.10 255.255.255.255`
	or
	`RP/0/RSP0/CPU0:router(config-ipsla-mpls-lsp-ping)# target ipv4 10.25.26.10/32`
	or
	`RP/0/RSP0/CPU0:router(config-ipsla-mpls-lsp-ping)# target traffic-eng tunnel 12`
	or
	`RP/0/RSP0/CPU0:router(config-ipsla-mpls-lsp-trace)# 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:**	(Optional) Specifies the local host IPv4 address used to select the LSP in an MPLS LSP ping operation.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-ipsla-mpls-lsp-ping)# lsp selector ipv4 127.0.0.2`	
Step 7	**force explicit-null** **Example:** `RP/0/RSP0/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/RSP0/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/RSP0/CPU0:router(config-ipsla-mpls-lsp-ping)# reply mode router-alert`
	or
	`RP/0/RSP0/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/RSP0/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/RSP0/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/RSP0/CPU0:router(config-ipsla-mpls-lsp-ping)# exit`
	`RP/0/RSP0/CPU0:router(config-ipsla-op)# exit`
	`RP/0/RSP0/CPU0:router(config-ipsla)# exit`
	`RP/0/RSP0/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:**	Schedules the start time of the operation. You can configure a basic schedule.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config)# ipsla schedule operation 432` `RP/0/RSP0/CPU0:router(config-ipsla-sched)#`	
Step 14	**start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*] **Example:** `RP/0/RSP0/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/RSP0/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)

SUMMARY STEPS

1. **configure**
2. **ipsla operation** *operation-number*
3. **type mpls lsp trace**
4. **output interface** *type interface-path-id*
5. Do one of the following:
   - **target ipv4** *destination-address destination-mask*
   - **target traffic-eng tunnel** *tunnel-interface*
6. **lsp selector ipv4** *ip-address*
7. **force explicit-null**
8. **reply dscp** *dscp-bits*
9. **reply mode router-alert**
10. **exp** *exp-bits*
11. **ttl** *time-to-live*
12. **exit**
13. **ipsla schedule operation** *operation-number*
14. **start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*]
15. Use the **commit** or **end** command.
16. **show ipsla statistics** [*operation-number*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**ipsla operation** *operation-number* **Example:**	Configures an IP SLA operation and specifies the operation number. The range is from 1 to 2048.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config)# ipsla operation 432`	
Step 3	**type mpls lsp trace** **Example:** `RP/0/RSP0/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/RSP0/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/RSP0/CPU0:router(config-ipsla-mpls-lsp-trace)# target ipv4 10.25.26.10 255.255.255.255`
	`RP/0/RSP0/CPU0:router(config-ipsla-mpls-lsp-trace)# target ipv4 10.25.26.10/32`
	or
	`RP/0/RSP0/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/RSP0/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/RSP0/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/RSP0/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.

	Command or Action	Purpose
Step 9	**reply mode router-alert** **Example:** `RP/0/RSP0/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/RSP0/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/RSP0/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/RSP0/CPU0:router(config-ipsla-mpls-lsp-trace)# exit`
	`RP/0/RSP0/CPU0:router(config-ipsla-op)# exit`
	`RP/0/RSP0/CPU0:router(config-ipsla)# exit`
	`RP/0/RSP0/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/RSP0/CPU0:router(config)# ipsla schedule operation 432`
	`RP/0/RSP0/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/RSP0/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.

	Command or Action	Purpose
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/RSP0/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.

SUMMARY STEPS

1. **configure**
2. **ipsla reaction operation** *operation-number*
3. **react** [**connection-loss**]
4. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**ipsla reaction operation** *operation-number* **Example:** `RP/0/RSP0/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/RSP0/CPU0:router(config-ipsla-react)# react connection-loss`
	`RP/0/RSP0/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.

SUMMARY STEPS

1. **configure**
2. **ipsla reaction operation** *operation-number*
3. **react** [**jitter-average** {**dest-to-source** | **source-to-dest**}]
4. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**ipsla reaction operation** *operation-number* **Example:** `RP/0/RSP0/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/RSP0/CPU0:router(config-ipsla-react)# react jitter-average`
	`RP/0/RSP0/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.

SUMMARY STEPS

1. **configure**
2. **ipsla reaction operation** *operation-number*
3. **react** [**packet-loss** [**dest-to-source** | **source-to-dest**]]
4. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**ipsla reaction operation** *operation-number* **Example:** `RP/0/RSP0/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/RSP0/CPU0:router(config-ipsla-react)# react packet-loss dest-to-source`
	`RP/0/RSP0/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.

SUMMARY STEPS

1. **configure**
2. **ipsla reaction operation** *operation-number*
3. **react** [**rtt**]
4. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**ipsla reaction operation** *operation-number* **Example:** `RP/0/RSP0/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/RSP0/CPU0:router(config-ipsla-react)# react rtt`
	`RP/0/RSP0/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.

SUMMARY STEPS

1. **configure**
2. **ipsla reaction operation** *operation-number*
3. **react** [**timeout**]
4. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:**	Enters global configuration mode.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router# configure`	
Step 2	**ipsla reaction operation** *operation-number* **Example:** `RP/0/RSP0/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/RSP0/CPU0:router(config-ipsla-react)# react timeout`
	`RP/0/RSP0/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.

SUMMARY STEPS

1. **configure**
2. **ipsla reaction operation** *operation-number*
3. **react** [**verify-error**]
4. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.

	Command or Action	Purpose
Step 2	**ipsla reaction operation** *operation-number* **Example:** `RP/0/RSP0/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/RSP0/CPU0:router(config-ipsla-react)# react verify-error`
	`RP/0/RSP0/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 24: 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 250.
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 249.
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 251.
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 253.

Generating Events for Each Violation

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

SUMMARY STEPS

1. **configure**
2. **ipsla reaction operation** *operation-number*
3. **react** [**connection-loss** | **jitter-average** {**dest-to-source** | **source-to-dest**} | **packet-loss** [**dest-to-source** | **source-to-dest**] | **rtt** | **timeout** | **verify-error**]
4. **threshold type immediate**
5. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**ipsla reaction operation** *operation-number* **Example:**	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.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config)# ipsla reaction operation 432`	

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/RSP0/CPU0:router(config-ipsla-react)# react timeout`
	`RP/0/RSP0/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/RSP0/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.

SUMMARY STEPS

1. **configure**
2. **ipsla reaction operation** *operation-number*
3. **react** [**connection-loss** | **jitter-average** {**dest-to-source** | **source-to-dest**} | **packet-loss** [**dest-to-source** | **source-to-dest**] | **rtt** | **timeout** | **verify-error**]
4. **threshold type consecutive** *occurrences*
5. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**ipsla reaction operation** *operation-number* **Example:** `RP/0/RSP0/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/RSP0/CPU0:router(config-ipsla-react)# react connection-loss`
	`RP/0/RSP0/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/RSP0/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.

SUMMARY STEPS

1. **configure**
2. **ipsla reaction operation** *operation-number*

3. **react** [**connection-loss** | **jitter-average** {**dest-to-source** | **source-to-dest**} | **packet-loss** [**dest-to-source** | **source-to-dest**] | **rtt** | **timeout** | **verify-error**]
4. **threshold type xofy** *X value Y value*
5. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**ipsla reaction operation** *operation-number* **Example:** `RP/0/RSP0/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/RSP0/CPU0:router(config-ipsla-react)# react rtt`
								`RP/0/RSP0/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/RSP0/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.

SUMMARY STEPS

1. **configure**
2. **ipsla reaction operation** *operation-number*
3. **react** [**connection-loss** | **jitter-average** {**dest-to-source** | **source-to-dest**} | **packet-loss** [**dest-to-source** | **source-to-dest**] | **rtt** | **timeout** | **verify-error**]
4. **threshold type average** *number-of-probes*
5. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**ipsla reaction operation** *operation-number* **Example:** `RP/0/RSP0/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/RSP0/CPU0:router(config-ipsla-react)# react packet-loss dest-to-source`
								`RP/0/RSP0/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/RSP0/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.

	Command or Action	Purpose
		• **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

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.

SUMMARY STEPS

1. **configure**
2. **ipsla reaction operation** *operation-number*
3. **react** [**connection-loss** | **jitter-average** {**dest-to-source** | **source-to-dest**} | **packet-loss** [**dest-to-source** | **source-to-dest**] | **rtt** | **timeout** | **verify-error**]
4. **action** [**logging** | **trigger**]
5. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**ipsla reaction operation** *operation-number* **Example:** `RP/0/RSP0/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:**	Specifies a reaction if there is a connection-loss for the monitored operation.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-ipsla-react)# react connection-loss` `RP/0/RSP0/CPU0:router(config-ipsla-react-cond)#`	
Step 4	**action** [**logging** | **trigger**] **Example:** `RP/0/RSP0/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 server twamp

Perform this task to configure server twamp.

SUMMARY STEPS

1. **configure**
2. **ipsla**
3. **server twamp**
4. **port** *number*
5. **timer inactivity** *value*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**ipsla** **Example:** `RP/0/RSP0/CPU0:router(config)# ipsla`	Enters the IPSLA configuration mode.
Step 3	**server twamp** **Example:** `RP/0/RSP0/CPU0:router(config-ipsla)# server twamp`	Enables the server twamp configuration mode. The port and timer details for the server can be configured.
Step 4	**port** *number* **Example:** `RP/0/RSP0/CPU0:router(config-ipsla)# port 80`	The port details for the server.
Step 5	**timer inactivity** *value* **Example:** `RP/0/RSP0/CPU0:router(config-ipsla)# timer inactivity 100`	The timer details for the server. The server can remain inactive for the set time.

Configuring responder twamp

Perform this task to configure responder twamp.

SUMMARY STEPS

1. **configure**
2. **ipsla**
3. **responder twamp**
4. **timeout** *value*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.

	Command or Action	Purpose
Step 2	**ipsla** **Example:** `RP/0/RSP0/CPU0:router(config)# ipsla`	Enters the IPSLA configuration mode.
Step 3	**responder twamp** **Example:** `RP/0/RSP0/CPU0:router(config-ipsla)# responder twamp`	Enables the responder twamp configuration mode. The timer details for the responder can be configured.
Step 4	**timeout** *value* **Example:** `RP/0/RSP0/CPU0:router(config-ipsla)# timeout 100`	The timer details for the responder. The responder can remain inactive for the set time.

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.

SUMMARY STEPS

1. **configure**
2. **ipsla**
3. **mpls discovery vpn**
4. **interval** *minutes*
5. **exit**
6. **mpls lsp-monitor**
7. **monitor** *monitor-id*
8. **type mpls lsp ping**
9. **vrf** *vrf-name*
10. **scan interval** *scan-interval*
11. **scan delete-factor** *factor-value*
12. **timeout** *milliseconds*

13. **datasize request** *size*
14. **lsp selector ipv4** *ip-address*
15. **force explicit-null**
16. **reply dscp** *dscp-bits*
17. **reply mode router-alert**
18. **ttl** *time-to-live*
19. **tag** *text*
20. **exp** *exp-bits*
21. **statistics hourly** [**buckets** *hours*]
22. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**ipsla** **Example:** `RP/0/RSP0/CPU0:router(config)# ipsla`	Enters IP SLA configuration mode and configures IP service level agreements.
Step 3	**mpls discovery vpn** **Example:** `RP/0/RSP0/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/RSP0/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/RSP0/CPU0:router(config-ipsla-mpls-discovery-vpn)# exit`	Exits MPLS discovery VPN configuration mode.

Step 6	**mpls lsp-monitor**
	Example:
	`RP/0/RSP0/CPU0:router(config-ipsla)# mpls lsp-monitor`
	`RP/0/RSP0/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.

	Command or Action	Purpose

Step 7	**monitor** *monitor-id*
	Example:
	`RP/0/RSP0/CPU0:router(config-ipsla-mplslm)# monitor 1`
	`RP/0/RSP0/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/RSP0/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/RSP0/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/RSP0/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/RSP0/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/RSP0/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:**	(Optional) Specifies the payload size of the MPLS LSP echo request packets. The default value is 100 bytes.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-ipsla-mplslm-lsp-ping)# datasize request 512`	**Note** This command is available in MPLS LSP ping mode only.
Step 14	**lsp selector ipv4** *ip-address* **Example:** `RP/0/RSP0/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/RSP0/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/RSP0/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/RSP0/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/RSP0/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/RSP0/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/RSP0/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.

	Command or Action	Purpose
Step 21	**statistics hourly** [**buckets** *hours*] **Example:** `RP/0/RSP0/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.

SUMMARY STEPS

1. **configure**
2. **ipsla**
3. **mpls discovery vpn**
4. **interval** *minutes*
5. **exit**
6. **mpls lsp-monitor**
7. **monitor** *monitor-id*
8. **type mpls lsp trace**
9. **vrf** *vrf-name*
10. **scan interval** *scan-interval*
11. **scan delete-factor** *factor-value*
12. **timeout** *milliseconds*

13. **lsp selector ipv4** *ip-address*
14. **force explicit-null**
15. **reply dscp** *dscp-bits*
16. **reply mode router-alert**
17. **ttl** *time-to-live*
18. **tag** *text*
19. **exp** *exp-bits*
20. **statistics hourly** [**buckets** *hours*]
21. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**ipsla** **Example:** `RP/0/RSP0/CPU0:router(config)# ipsla`	Enters IP SLA configuration mode and configures IP service level agreements.
Step 3	**mpls discovery vpn** **Example:** `RP/0/RSP0/CPU0:router(config-ipsla)# mpls discovery vpn`	(Optional) Enables MPLS VPN BGP next-hop neighbor discovery.
Step 4	**interval** *minutes* **Example:** `RP/0/RSP0/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/RSP0/CPU0:router(config-ipsla-mpls-discovery-vpn)# exit`	Exits MPLS discovery VPN configuration mode.

Step 6	**mpls lsp-monitor**
	Example:
	`RP/0/RSP0/CPU0:router(config-ipsla)# mpls lsp-monitor`
	`RP/0/RSP0/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.

	Command or Action	Purpose

Step 7	**monitor** *monitor-id*
	Example:
	`RP/0/RSP0/CPU0:router(config-ipsla-mplslm)# monitor 1`
	`RP/0/RSP0/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/RSP0/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/RSP0/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/RSP0/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/RSP0/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/RSP0/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.

	Command or Action	Purpose
Step 13	**lsp selector ipv4** *ip-address* **Example:** `RP/0/RSP0/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/RSP0/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/RSP0/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/RSP0/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/RSP0/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/RSP0/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/RSP0/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/RSP0/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.

	Command or Action	Purpose
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.

SUMMARY STEPS

1. **configure**
2. **ipsla**
3. **mpls lsp-monitor**
4. **reaction monitor** *monitor-id*
5. **react** {**connection-loss** | **timeout**}
6. **action logging**
7. **threshold type** {**consecutive** *occurrences* | **immediate**}
8. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.

	Command or Action	Purpose
Step 2	**ipsla** **Example:** `RP/0/RSP0/CPU0:router(config)# ipsla`	Enters IP SLA configuration mode and configures IP service level agreements.

Step 3	**mpls lsp-monitor**
	Example:
	`RP/0/RSP0/CPU0:router(config-ipsla)# mpls lsp-monitor`
	`RP/0/RSP0/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/RSP0/CPU0:router(config-ipsla-mplslm)# reaction monitor 2`
	`RP/0/RSP0/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/RSP0/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/RSP0/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/RSP0/CPU0:router(config-ipsla-mplslm-react-cond)# threshold type consecutive`	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.

SUMMARY STEPS

1. **configure**
2. **ipsla**
3. **mpls lsp-monitor**
4. **schedule monitor** *monitor-id*
5. **frequency** *seconds*
6. **schedule period** *seconds*
7. **start-time** *hh*:*mm*:*ss* [*day* | *month day*]
8. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**ipsla** **Example:** `RP/0/RSP0/CPU0:router(config)# ipsla`	Enters IP SLA configuration mode and configures IP service level agreements.

Step 3	**mpls lsp-monitor**
	Example:
	`RP/0/RSP0/CPU0:router(config-ipsla)# mpls lsp-monitor`
	`RP/0/RSP0/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/RSP0/CPU0:router(config-ipsla-mplslm)# schedule monitor 2`
	`RP/0/RSP0/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:**	(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 or Action	Purpose
	`RP/0/RSP0/CPU0:router(config-ipsla-mplslm-sched)# frequency 600`	command. You must specify this value before scheduling an MPLS LSP monitor instance start time.
Step 6	**schedule period** *seconds* **Example:** `RP/0/RSP0/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/RSP0/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.

LSP Path Discovery

Perform this task to configure the LSP Path Discovery (LPD) and its required parameters, including echo interval, path, and scan.

SUMMARY STEPS

1. **configure**
2. **ipsla**
3. **mpls lsp-monitor**
4. **monitor** *monitor-id*
5. **type mpls lsp ping**
6. **path discover**
7. **echo interval** *time*
8. **echo maximum lsp selector ipv4** *host address*
9. **echo multipath bitmap-size** *size*

10. **echo retry** *count*
11. **echo timeout** *value*
12. **path retry** *range*
13. **path secondary frequency** {**both** | **connection-loss** | **timeout**} *value*}
14. **scan period** *value*
15. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**ipsla** **Example:** `RP/0/RSP0/CPU0:router(config)# ipsla`	Enters IP SLA configuration mode and configures IP service level agreements.
Step 3	**mpls lsp-monitor** **Example:** `RP/0/RSP0/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/RSP0/CPU0:router(config-ipsla-mplslm)# monitor 2`	Configures an MPLS LSP monitor instance.
Step 5	**type mpls lsp ping** **Example:** `RP/0/RSP0/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/RSP0/CPU0:router(config-ipsla-mplslm-lsp-ping)# path discover`	Enables LSP path discovery.
Step 7	**echo interval** *time* **Example:** `RP/0/RSP0/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.

	Command or Action	Purpose
Step 8	**echo maximum lsp selector ipv4** *host address* **Example:** `RP/0/RSP0/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/RSP0/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/RSP0/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/RSP0/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/RSP0/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/RSP0/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/RSP0/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:

	Command or Action	Purpose
		• **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
```

Additional References

The following sections provide references related to IP Service Level Agreements.

Related Documents

Related Topic	Document Title
IP Service Level Agreement commands	*IP Service Level Agreement Commands* module in the *System Monitoring Command Reference for Cisco ASR 9000 Series Routers*
Information about user groups and task IDs	*Configuring AAA Services* module in the *System Security Configuration Guide for Cisco ASR 9000 Series Routers*

Standards	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	—

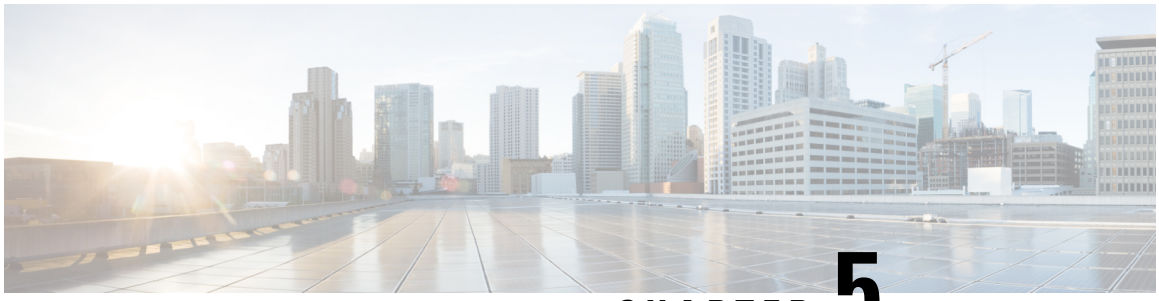
MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs	Title
No new or modified RFCs are supported by this feature, and support for existing RFCs has not been modified by this feature.	—

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	*http://www.cisco.com/cisco/web/support/index.html*



CHAPTER 5

Implementing Logging Services

This module describes the new and revised 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.

Note For more information about logging services on the Cisco IOS XR Software and complete descriptions of the logging commands listed in this module, see the Related Documents, on page 305 section of this module.

Feature History for Implementing Logging Services

Release	Modification
Release 3.7.2	This feature was introduced.
Release 6.1.2	Platform Automated Monitoring (PAM) tool was introduced for all Cisco IOS XR 64-bit platforms.

Prerequisites for Implementing Logging Services

These prerequisites are required to implement logging services 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.

Information About Implementing Logging Services

System Logging Process

By default, routers are configured to send syslog messages to a syslog process. The syslog process controls the distribution of messages to the destination of syslog messages such as the logging buffer, terminal lines, or a syslog server. The syslog process also sends messages to the console terminal by default.

Format of System Logging Messages

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 -group -severity -message -code* : *message-text*

This is a sample syslog message:

```
RP/0/RSP0/CPU0:router:Nov 28 23:56:53.826 : config[65710]: %SYS-5-CONFIG_I : Configured
from console by console
```

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

Table 25: General Syslog Message Format

Field	Description
node-id	Node from which the syslog message originated.
timestamp	Time stamp in the form *month day HH:MM:SS*, indicating when the message was generated. **Note** The time-stamp format can be modified using the **service timestamps** command. See the Modifying the Format of Time Stamps, on page 290 section.
process-name	Process that generated the syslog message.
[*pid*]	Process ID (pid) of the process that generated the syslog message.
%message -group- severity -message -code	Message group name, severity, and message code associated with the syslog message.
message-text	Text string describing the syslog message.

Duplicate Message Suppression

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. The suppression and logging process is the same for logging history and for external syslog servers.

When duplicate message suppression is enabled, two types of events are handled differently:

- New messages

  New messages are always logged immediately.

- Repeated messages

  Repeated messages are subject to suppression. The suppression of repeated messages is interrupted when a new message occurs.

For information about configuring this feature, see the Suppressing Duplicate Syslog Messages, on page 292.

Syslog Message Destinations

Syslog message logging to the console terminal is enabled by default. To disable logging to the console terminal, use the **logging console disable** command in global configuration mode. To reenable logging to the console terminal, use the **logging console** command in global configuration mode.

Syslog messages can be sent to destinations other than the console, such as the logging buffer, syslog servers, and terminal lines other than the console (such as vtys).

This table lists the commands used to specify syslog destinations.

Table 26: Commands Used to Set Syslog Destinations

Command	Description
logging buffered	Specifies the logging buffer as a destination for syslog messages.
logging {*hostname* | *ip-address*}	Specifies a syslog server host as a destination for syslog messages. IPv4 and IPv6 are supported.
logging monitor	Specifies terminal lines other than the console as destinations for syslog messages.

The **logging buffered** command copies logging messages to the logging buffer. The buffer is circular, so newer messages overwrite older messages after the buffer is full. To display the syslog messages that are logged in the logging buffer, use the **show logging** command. The first message displayed is the oldest message in the buffer. To clear the current contents of the logging buffer, use the **clear logging** command. To disable logging to the logging buffer, use the **no logging buffered** command in global configuration mode.

The **logging** command identifies a syslog server host to receive logging messages. By issuing this command more than once, you build a list of syslog servers that receive logging messages. To delete the syslog server with the specified IP address (IPv4 and IPv6 are supported) or hostname from the list of available syslog servers, use the **no logging** command in global configuration mode.

The **logging monitor** command globally enables the logging of syslog messages to terminal lines other than the console, such as vtys. To disable logging to terminal lines other than the console, use the **no logging monitor** command in global configuration mode.

Guidelines for Sending Syslog Messages to Destinations Other Than the Console

The logging process sends syslog messages to destinations other than the console terminal and the process is enabled by default. Logging is enabled to the logging buffer, terminal lines and syslog servers.

Logging for the Current Terminal Session

The **logging monitor** command globally enables the logging of syslog messages to terminal lines other than console terminal. Once the **logging monitor** command is enabled, use the **terminal monitor** command to display syslog messages during a terminal session.

To disable the logging of syslog messages to a terminal during a terminal session, use the **terminal monitor disable** command in EXEC mode. The **terminal monitor disable** command disables logging for only the current terminal session.

To reenable the logging of syslog messages for the current terminal session, use the **terminal monitor** command in EXEC mode.

Note The **terminal monitor** and **terminal monitor disable** commands are set locally and will not remain in effect after the terminal session is ended.

Syslog Messages Sent to Syslog Servers

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

- UNIX system facilities
- Hostname prefix logging
- Source interface logging

UNIX System Logging Facilities

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

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

Table 27: Logging Facility Type Keywords

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

Facility Type Keyword	Description
news	Indicates USENET news.
sys9	Indicates system use.
sys10	Indicates system use.
sys11	Indicates system use.
sys12	Indicates system use.
sys13	Indicates system use.
sys14	Indicates system use.
syslog	Indicates the system log.
user	Indicates user process.
uucp	Indicates UNIX-to-UNIX copy system.

Hostname Prefix Logging

To help manage system logging messages sent to syslog servers, Cisco IOS XR Software supports hostname prefix logging. When enabled, hostname prefix logging appends a hostname prefix to syslog messages being sent from the router to syslog servers. You can use hostname prefixes to sort the messages being sent to a given syslog server from different networking devices.

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

Syslog Source Address Logging

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

UNIX Syslog Daemon Configuration

To configure the syslog daemon on a 4.3 BSD UNIX system, include a line such as the following in the /etc/syslog.conf file:

```
local7.debug /usr/adm/logs/cisco.log
```

The **debugging** keyword specifies the syslog level; see Table 31: Syslog Message Severity Levels, on page 284 for a general description of other keywords. The **local7** keyword specifies the logging facility to be used; see Table 31: Syslog Message Severity Levels, on page 284 for a general description of other keywords.

The syslog daemon sends messages at this level or at a more severe level to the file specified in the next field. The file must already exist, and the syslog daemon must have permission to write to it.

Archiving Logging Messages on 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.

Setting Archive Attributes

To create a logging archive and specify how the logging messages will be collected and stored, use the **logging archive** command in global configuration mode. The **logging archive** command enters the logging archive submode where you can configure the attributes for archiving syslogs.

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

Table 28: Commands Used to Set Syslog Archive Attributes

Command	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{**dailyweekly**}	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. See the Severity Levels, on page 283 for more information.
threshold	Specifics the threshold percentage for archive logs.

Archive Storage Directories

By default, syslog archives are stored in the directory <device>/var/log. Individual archive files are saved to sub directories based on the year, month, and day the archive was created. For example, archive files created on February 26, 2006 are stored in this directory:

```
harddisk:/var/log/2006/02/26
```

Severity Levels

You can limit the number of messages sent to the console, monitor and trap logging destinations by specifying the severity level of syslog messages sent to that destination (see Table 31: Syslog Message Severity Levels, on page 284 for severity level definitions). However, for the logging buffer destination, syslog messages of all severity will be sent to it.

This table lists the commands used to control the severity level of syslog messages.

Table 29: Commands Used to Control the Severity Level of Syslog Messages

Command	Description
logging buffered [*severity*]	Limits the syslog messages that are displayed in the output of **show logging** based on severity. However, syslog messages of all severity will be sent to the logging buffer.
logging console [*severity*]	Limits the syslog messages sent to the console terminal based on severity.
logging monitor [*severity*]	Limits the syslog messages sent to terminal lines based on severity.
logging trap [*severity*]	Limits the syslog messages sent to syslog servers based on severity.
severity *severity*	Limits the syslog messages sent to a syslog archive based on severity.

The **logging console**, **logging monitor**, and **logging traps** commands limit syslog messages sent to their respective destinations to messages with a level number at or below the specified severity level, which is specified with the *severity* argument. However, in the case of the **logging buffered** command, messages of all severity will continue to be sent to the logging buffer. This command only limits the syslog messages displayed in the output of **show logging** to messages with a level number at or below the specified *severity* argument.

Note Syslog messages of lower severity level indicate events of higher importance. See Table 31: Syslog Message Severity Levels, on page 284 for severity level definitions.

Logging History Table

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

Messages are stored in the history table, because SNMP traps are not guaranteed to reach their destination. By default, one message of the level warning and above (see Table 31: Syslog Message Severity Levels, on page 284) is stored in the history table even if syslog traps are not enabled.

This table lists the commands used to change the severity level and table size defaults of the logging history table

Table 30: Logging History Table Commands

Command	Description
logging history *severity*	Changes the default severity level of syslog messages stored in the history file and sent to the SNMP server.
logging history size *number*	Changes the number of syslog messages that can be stored in the history table.

Note Table 31: Syslog Message Severity Levels, on page 284 lists the level keywords and severity level. For SNMP usage, the severity level values use +1. For example, **emergency** equals 1 not 0 and **critical** equals 3 not 2.

Syslog Message Severity Level Definitions

This table lists the severity level keywords that can be supplied for the *severity* argument and corresponding UNIX syslog definitions in order from the most severe level to the least severe level.

Table 31: Syslog Message Severity Levels

Severity Keyword	Level	Description	Syslog Definition
emergencies	0	System unusable	LOG_EMERG
alerts	1	Immediate action needed	LOG_ALERT
critical	2	Critical conditions	LOG_CRIT
errors	3	Error conditions	LOG_ERR
warnings	4	Warning conditions	LOG_WARNING
notifications	5	Normal but significant condition	LOG_NOTICE
informational	6	Informational messages only	LOG_INFO
debugging	7	Debugging messages	LOG_DEBUG

Syslog Severity Level Command Defaults

This table lists the default severity level settings for the commands that support the *severity* argument.

Table 32: Severity Level Command Defaults

Command	Default Severity Keyword	Level
logging buffered	debugging	7
logging console	informational	6

Command	Default Severity Keyword	Level
logging history	warnings	4
logging monitor	debugging	7
logging trap	informational	6

How to Implement Logging Services

Setting Up Destinations for System Logging Messages

This task explains how to configure logging to destinations other than the console terminal.

For conceptual information, see the Syslog Message Destinations, on page 279 section.

SUMMARY STEPS

1. **configure**
2. **logging buffered** [*size* | *severity*]
3. **logging monitor** [*severity*]
4. Use the **commit** or **end** command.
5. **terminal monitor**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**logging buffered** [*size*	*severity*] **Example:** `RP/0/RSP0/CPU0:router(config)# logging buffered severity warnings`	Specifies the logging buffer as a destination for syslog messages, sets the size of the logging buffer, and limits syslog messages displayed in the output of **show logging** based on severity. • The default value for the *size* argument is 4096 bytes. • The default value for the *severity* argument is **debugging**. • Keyword options for the *severity* argument are **emergencies**, **alerts**, **critical**, **errors**, **warnings**, **notifications**, **informational**, and **debugging**. • By default, entering this command without specifying a severity level for the *severity* argument or specifying the size of the buffer for the *size* argument sets the

	Command or Action	Purpose
		severity level to **debugging** and the buffer size to 4096 bytes.
Step 3	**logging monitor** [*severity*] **Example:** `RP/0/RSP0/CPU0:router(config)# logging monitor critical`	Specifies terminal lines other than console terminal as destinations for syslog messages and limits the number of messages sent to terminal lines based on severity. • Keyword options for the *severity* argument are **emergencies**, **alerts**, **critical**, **errors**, **warnings**, **notifications**, **informational**, and **debugging**. • By default, entering this command without specifying a severity level for the *severity* argument sets the severity level to **debugging**.
Step 4	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 5	**terminal monitor** **Example:** `RP/0/RSP0/CPU0:router# terminal monitor`	Enables the display of syslog messages for the current terminal session. **Note** The logging of syslog message for the current terminal can be disabled with the **terminal monitor disable** command. • Use this command to reenable the display of syslog messages for the current session if the logging of messages for the current session was disabled with **terminal monitor disable** command. **Note** Because this command is an EXEC mode command, it is set locally and will not remain in effect after the current session is ended.

Configuring Logging to a Remote Server

You must have connectivity with syslog servers and snmp servers to configure them as the recipients for syslog messages.

Configuration Example for Logging to Syslog Server

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

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

Configuration Example for Logging to SNMP Server

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

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

For more information on SNMP server configurations, see the *Configuring Simple Network Management Protocol* chapter in the *System Management Configuration Guide for Cisco ASR 9000 Series Routers*

Configuring the Settings for the Logging History Table

This task explains how to configure the settings for the logging history table.

For conceptual information, see the Severity Levels, on page 283 section.

Before you begin

Logging of messages to an SNMP NMS is enabled by the **snmp-server enable traps syslog** command. For more information about SNMP, see the Related Documents, on page 305 section.

SUMMARY STEPS

1. **configure**
2. **logging history** *severity*
3. **logging history size** *number*
4. Use the **commit** or **end** command.
5. **show logging history**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**logging history** *severity* **Example:**	Changes the default severity level of syslog messages stored in the history file and sent to the SNMP server.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config)# logging history errors`	• By default, syslog messages at or below the **warnings** severity level are stored in the history file and sent to the SNMP server.
Step 3	**logging history size** *number* **Example:** `RP/0/RSP0/CPU0:router(config)# logging history size 200`	Changes the number of syslog messages that can be stored in the history table. • By default, one syslog message is stored in the history table. **Note** When the history table is full (that is, when it contains the maximum number of messages specified with this command), the oldest message is deleted from the table to allow the new message to be stored.
Step 4	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 5	**show logging history** **Example:** `RP/0/RSP0/CPU0:router# show logging history`	(Optional) Displays information about the state of the syslog history table.

Modifying Logging to the Console Terminal and the Logging Buffer

This task explains how to modify logging configuration for the console terminal and the logging buffer.

Note Logging is enabled by default.

SUMMARY STEPS

1. **configure**
2. **logging buffered** [*size* | *severity*]
3. **logging console** [*severity*]
4. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**logging buffered** [*size* | *severity*] **Example:** `RP/0/RSP0/CPU0:router(config)# logging buffered size 60000`	Specifies the logging buffer as a destination for syslog messages, sets the size of the logging buffer, and limits the syslog messages displayed in the output of **show logging** based on severity. • The default for the *size* argument is 4096 bytes. • The default for the *severity* argument is **debugging**. • Keyword options for the *severity* argument are **emergencies**, **alerts**, **critical**, **errors**, **warnings**, **notifications**, **informational**, and **debugging**. • By default, entering this command without specifying a severity level for the *severity* argument or specifying the size of the buffer for the *size* argument sets the severity level to **debugging** and the buffer size to 4096 bytes.
Step 3	**logging console** [*severity*] **Example:** `RP/0/RSP0/CPU0:router(config)# logging console alerts`	Limits messages sent to the console terminal based on severity. • Syslog messages are logged to the console terminal at the **informational** severity level by default. • Keyword options for the *severity* argument are **emergencies**, **alerts**, **critical**, **errors**, **warnings**, **notifications**, **informational**, and **debugging**. • Entering this command without specifying a severity level for the *severity* argument sets the severity level to **informational**. **Note** Use this command to reenable logging to the console terminal if it was disabled with the **logging console disable** command.
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.

	Command or Action	Purpose
		• **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Modifying the Format of Time Stamps

This task explains how to modify the time-stamp format for syslog and debugging messages.

SUMMARY STEPS

1. **configure**
2. Do one of the following:
   - **service timestamps log datetime** [**localtime**] [**msec**] [**show-timezone**]
   - **service timestamps log uptime**
3. Do one of the following:
   - **service timestamps debug datetime** [**localtime**] [**msec**] [**show-timezone**]
   - **service timestamps debug uptime**
4. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.

Step 2	Do one of the following:
	• **service timestamps log datetime** [**localtime**] [**msec**] [**show-timezone**]
	• **service timestamps log uptime**
	Example:
	`RP/0/RSP0/CPU0:router(config)# service timestamps log datetime localtime msec`
	or
	`RP/0/RSP0/CPU0:router(config)# service timestamps log uptime`
		Modifies the time-stamp format for syslog messages.
		• By default, time stamps are enabled. The default time-stamp format is month day HH:MM:SS.
		• Issuing the **service timestamps log datetime** command configures syslog messages to be time-stamped with the date and time.
		  • The optional **localtime** keyword includes the local time zone in time stamps.
		  • The optional **msec** keyword includes milliseconds in time stamps.
		  • The optional **show-timezone** keyword includes time zone information in time stamps.

	Command or Action	Purpose
		• Issuing the **service timestamps log uptime** command configures syslog messages to be time-stamped with the time that has elapsed since the router last rebooted. • The **service timestamps log uptime** command configures time-stamps to be configured in HHHH:MM:SS, indicating the time since the router last rebooted.

Step 3	Do one of the following:
	• **service timestamps debug datetime** [**localtime**] [**msec**] [**show-timezone**]
	• **service timestamps debug uptime**
	Example:
	`RP/0/RSP0/CPU0:router(config)# service timestamps debug datetime msec show-timezone`
	or
	`RP/0/RSP0/CPU0:router(config)# service timestamps debug uptime`
		Modifies the time-stamp format for debugging messages.
		• By default, time-stamps are enabled. The default time stamp format is month day HH:MM:SS.
		• Issuing the **service timestamps log datetime** command configures debugging messages to be time-stamped with the date and time.
		  • The optional **localtime** keyword includes the local time zone in time stamps.
		  • The optional **msec** keyword includes milliseconds in time stamps.
		  • The optional **show-timezone** keyword includes time zone information in time stamps.
		• Issuing the **service timestamps log uptime** command configures debugging messages to be time-stamped with the time that has elapsed since the networking device last rebooted.
		Tip Entering the **service timestamps** command without any keywords or arguments is equivalent to entering the **service timestamps debug uptime** command.

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.

Disabling Time Stamps

This tasks explains how to disable the inclusion of time stamps in syslog messages.

SUMMARY STEPS

1. **configure**
2. Do one of the following:
   - **service timestamps disable**
   - **no service timestamps** [**debug** | **log**] [**datetime** [**localtime**] [**msec**] [**show-timezone**]] | **uptime**]
3. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	Do one of the following: • **service timestamps disable** • **no service timestamps** [**debug** | **log**] [**datetime** [**localtime**] [**msec**] [**show-timezone**]] | **uptime**]	Disables the inclusion of time stamps in syslog messages. **Note** Both commands disable the inclusion of time stamps in syslog messages; however, specifying the **service timestamps disable** command saves the command to the configuration, whereas specifying the **no** form of the **service timestamps** command removes the command from the configuration.
Step 3	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.

Suppressing Duplicate Syslog Messages

This task explains how to suppress the consecutive logging of duplicate syslog messages.

SUMMARY STEPS

1. **configure**
2. **logging suppress duplicates**
3. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**logging suppress duplicates** **Example:** `RP/0/RSP0/CPU0:router(config)# logging suppress duplicates`	Prevents the consecutive logging of duplicate syslog messages. **Caution** If this command is enabled during debugging sessions, you could miss important information related to problems that you are attempting to isolate and resolve. In such a case, you might consider disabling this command.
Step 3	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.

Disabling the Logging of Link-Status Syslog Messages

This task explains how to disable the logging of link-status syslog messages for logical and physical links.

When the logging of link-status messages is enabled, the router can generate a high volume of link-status updown syslog messages. Disabling the logging of link-status syslog messages reduces the number of messages logged.

SUMMARY STEPS

1. **configure**
2. **logging events link-status disable**
3. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**logging events link-status disable** **Example:** `RP/0/RSP0/CPU0:router(config)# logging events link-status disable`	Disables the logging of link-status syslog messages for software (logical) and physical links. • The logging of link-status syslog messages is enabled by default for physical links. • To enable link-status syslog messages for both physical and logical links, use the **logging events link-status software-interfaces** command. • Use the **no logging events link-status** command to enable link-status syslog messages on physical links only.
Step 3	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.

Displaying System Logging Messages

This task explains how to display the syslog messages stored in the logging buffer.

Note The commands can be entered in any order.

SUMMARY STEPS

1. **show logging**
2. **show logging location** *node-id*
3. **show logging process** *name*
4. **show logging string** *string*
5. **show logging start** *month day hh*:*mm*:*ss*

6. **show logging end** *month day hh*:*mm*:*ss*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**show logging** **Example:** `RP/0/RSP0/CPU0:router# show logging`	Displays all syslog messages stored in the buffer.
Step 2	**show logging location** *node-id* **Example:** `RP/0/RSP0/CPU0:router# show logging location 0/1/CPU0`	Displays syslog messages that have originated from the designated node.
Step 3	**show logging process** *name* **Example:** `RP/0/RSP0/CPU0:router# show logging process init`	Displays syslog messages that are related to the specified process.
Step 4	**show logging string** *string* **Example:** `RP/0/RSP0/CPU0:router# show logging string install`	Displays syslog messages that contain the specified string.
Step 5	**show logging start** *month day hh*:*mm*:*ss* **Example:** `RP/0/RSP0/CPU0:router# show logging start december 1 10:30:00`	Displays syslog messages in the logging buffer that were generated on or after the specified date and time.
Step 6	**show logging end** *month day hh*:*mm*:*ss* **Example:** `RP/0/RSP0/CPU0:router# show logging end december 2 22:16:00`	Displays syslog messages in the logging buffer that were generated on or before the specified date and time.

Archiving System Logging Messages to a Local Storage Device

This task explains how to display save syslog messages to an archive on a local storage device.

Before you begin

Note The local storage device must have enough space available to store the archive files. We recommend that syslogs be archived to the harddisk because it has more capacity than flash disks.

SUMMARY STEPS

1. **configure**
2. **logging archive**
3. **device** {**disk0** | **disk1** | **harddisk**}
4. **frequency** {**daily** | **weekly**}
5. **severity** *severity*
6. **archive-length** *weeks*
7. **archive-size** *size*
8. **file-size** *size*
9. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**logging archive** **Example:** `RP/0/RSP0/CPU0:router(config)# logging archive`	Enters logging archive configuration mode.
Step 3	**device** {**disk0**	**disk1**	**harddisk**} **Example:** `RP/0/RSP0/CPU0:router(config-logging-arch)# device disk1`	Specify the device to be used for logging syslogs. • This step is required. 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. • By default, the logs are created under the directory <device>/var/log
Step 4	**frequency** {**daily**	**weekly**} **Example:** `RP/0/RSP0/CPU0:router(config-logging-arch)# frequency weekly`	(Optional) Specifies if logs are collected on a daily or weekly basis. Logs are collected daily by default.
Step 5	**severity** *severity* **Example:** `RP/0/RSP0/CPU0:router(config-logging-arch)# severity warnings`	(Optional) 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. The severity levels are: • emergencies • alerts

	Command or Action	Purpose
		• critical • errors • warnings • notifications • informational • debugging See the Syslog Message Severity Level Definitions, on page 284 section for information.
Step 6	**archive-length** *weeks* **Example:** `RP/0/RSP0/CPU0:router(config-logging-arch)# archive-length 6`	(Optional) 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. By default, archive logs are stored for 4 weeks.
Step 7	**archive-size** *size* **Example:** `RP/0/RSP0/CPU0:router(config-logging-arch)# archive-size 50`	(Optional) 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. The default archive size is 20 MB.
Step 8	**file-size** *size* **Example:** `RP/0/RSP0/CPU0:router(config-logging-arch)# file-size 10`	(Optional) 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. By default, the maximum file size is 1 megabyte.
Step 9	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Platform Automated Monitoring

Platform Automated Monitoring (PAM) is a system monitoring tool integrated into Cisco IOS XR software image to monitor issues such as process crash, memory leak, CPU hog, tracebacks, syslog and disk usage.

PAM is enabled by default on all Cisco IOS XR 64 bit platforms. When the PAM tool detects any of these system issues, it collects the required data to troubleshoot the issue, and generates a syslog message stating the issue. The auto-collected troubleshooting information is then stored as a separate file in harddisk:/cisco_support/ or in /misc/disk1/cisco_support/ directory.

PAM Events

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

The format of the .tgz file is: *PAM-<platform>-<PAM event>-<node-name>-<PAM process>-<YYYYMMDD>-<checksum>.tgz*.For example, *PAM-asr9k-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 301.

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-asr9k-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-asr9k-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-asr9k-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-asr9k-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-asr9k-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 EXEC mode:

For local RP:

```
Router# process shutdown pam_manager
```

For all RPs:

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

Re-enable PAM

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

For local RP:

```
Router# process start pam_manager
```

For all RPs:

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

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

Data Archiving in PAM

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

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

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

Files Collected by PAM Tool

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

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

Event Name	Commands and Files Collected by PAM
Process crash	• **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

Configuration Examples for Implementing Logging Services

This section provides these configuration examples:

Configuring Logging to the Console Terminal and the Logging Buffer: Example

This example shows a logging configuration where logging to the logging buffer is enabled, the severity level of syslog messages sent to the console terminal is limited to syslog messages at or below the **critical** severity level, and the size of the logging buffer is set to 60,000 bytes.

```
!
logging console critical
logging buffered 60000
!
```

Setting Up Destinations for Syslog Messages: Example

This example shows a logging configuration where logging is configured to destinations other than the console terminal. In this configuration, the following is configured:

- Logging is enabled to destinations other than the console terminal.
- Syslog messages at or below the **warnings** severity level are sent to syslog server hosts.
- Syslog messages at or below the **critical** severity level are sent to terminal lines.
- The size of the logging buffer is set to 60,000 bytes.
- The syslog server host at IP addresses 172.19.72.224 (IPv4) and 2001:DB8:A00:1::1/64 (IPv6) are configured as recipients for syslog messages.

```
!
logging trap warnings
logging monitor critical
logging buffered 60000
logging 172.19.72.224
logging 2001:DB8:A00:1::1/64
!
```

Configuring the Settings for the Logging History Table: Example

This example shows a logging configuration in which the size of the logging history table is to 200 entries and the severity of level of syslog messages sent to the logging history table is limited to messages at or below the **errors** severity level:

```
logging history size 200
logging history errors
```

Modifying Time Stamps: Example

This example shows a time-stamp configuration in which time stamps are configured to follow the format month date HH:MM:SS time zone:

```
service timestamps log datetime show-timezone
```

This example shows a time-stamp configuration in which time stamps are configured to follow the format month date HH:MM:SS.milliseconds time zone:

```
service timestamps log datetime msec show-timezone
```

Configuring a Logging Archive: Example

This example shows how to configure a logging archive, and define the archive attributes:

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# logging archive
RP/0/RSP0/CPU0:router(config-logging-arch)# device disk1
RP/0/RSP0/CPU0:router(config-logging-arch)# frequency weekly
RP/0/RSP0/CPU0:router(config-logging-arch)# severity warnings
RP/0/RSP0/CPU0:router(config-logging-arch)# archive-length 6
RP/0/RSP0/CPU0:router(config-logging-arch)# archive-size 50
RP/0/RSP0/CPU0:router(config-logging-arch)# file-size 10
```

Where to Go Next

To configure alarm log correlation, see the *Implementing and Monitoring Alarms and Logging Correlation* module in the *System Monitoring Configuration Guide for Cisco ASR 9000 Series Routers*.

Additional References

The following sections provide references related to implementing logging services on Cisco IOS XR software .

Related Documents

Related Topic	Document Title
Logging services command reference	*Logging Services Commands* module in the *System Monitoring Command Reference for Cisco ASR 9000 Series Routers*
Onboard Failure Logging (OBFL) configuration	*Onboard Failure Logging Commands* module in the *System Monitoring Configuration Guide for Cisco ASR 9000 Series Routers*.
Onboard Failure Logging (OBFL) commands	*Onboard Failure Logging Commands* module in the *System Monitoring Command Reference for Cisco ASR 9000 Series Routers*.
Alarm and logging correlation commands	*Alarm Management and Logging Correlation Commands* module in the *System Monitoring Command Reference for Cisco ASR 9000 Series Routers*.
Alarm and logging correlation configuration and monitoring tasks	*Implementing and Monitoring Alarms and Alarm Log Correlation* module in the *System Monitoring Configuration Guide for Cisco ASR 9000 Series Routers*.
SNMP commands	*SNMP Commands* module in the *System Monitoring Command Reference for Cisco ASR 9000 Series Routers*.
SNMP configuration tasks	*Implementing SNMP* module in the *System Monitoring Configuration Guide for Cisco ASR 9000 Series Routers*
Cisco IOS XR getting started material	*Cisco ASR 9000 Series Aggregation Services Router Getting Started Guide*
Information about user groups and task IDs	*Configuring AAA Services* module in the *System Security Command Reference for Cisco ASR 9000 Series Routers*.

Standards

Standards	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	—

MIBs

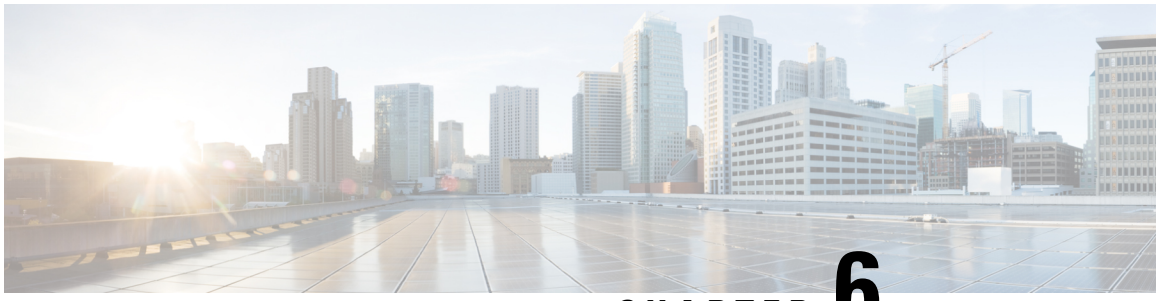
MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs	Title
No new or modified RFCs are supported by this feature, and support for existing RFCs has not been modified by this feature.	—

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	*http://www.cisco.com/cisco/web/support/index.html*



CHAPTER 6

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

Note For information about OBFL commands, console logging, alarms, and logging correlation, see Related Documents, on page 305.

Feature History for Implementing OBFL

Release	Modification
Release 3.7.2	This feature was introduced.
Release 5.2.2	Generic string logging feature was introduced.

Prerequisites

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

Information About Implementing OBFL

Data Collection Types

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 these:

- FRU part serial number
- OS version
- Boot time
- Total run time (hours in use)
- Boot status
- Temperature and voltage at boot
- Temperature and voltage history
- Other board specific errors

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. This includes:

Data Type	Details
Installation	Chassis name and slot number are stored at initial boot and for the most recent nine boots.
Temperature	Inlet and hotpoint temperatures are recorded 10 minutes after boot.
Run-time	Total run-time since initial installation. This is based on the local router clock with a 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.

Data Type	Details	
Environmental Factors	Temperature Value	Inlet and hot point temperature value change beyond the threshold set in the hardware inventory XML files.
	Voltage Value	+5, and MBUS +5, +3.3, and +2.2 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	Disabled	The time when OBFL logging was disabled with the **hw-module** {**all** | **subslot** *node-id*} **logging onboard disable** command in global configuration or administration configuration mode.
	Cleared	The time when OBFL logging was cleared with the **clear logging onboard** command in EXEC or administration EXEC mode.
	Reset to 0	The time when total line card runtime is reset to zero with the **clear logging onboard** command in EXEC or administration EXEC mode.

Supported Cards and Platforms

OBFL data collection is supported.

FRUs that have sufficient nonvolatile memory available for OBFL data storage support OBFL. For example, the processor supports the OBFL.

Table 33: OBFL Support by Card Type

Card Type	Cisco ASR 9000 Series Router
Route-switch processor (RSP)	Supported
Power supply cards: AC rectifier modules and DC power entry modules (PEMs)	Not Supported
Fan controller cards	Supported
Shared port adapters (SPA)	Not Supported

How to Implement OBFL

OBFL logging is configured for the router. If a new node is inserted, and OBFL is enabled for that slot or for all slots, then OBFL is enabled for the new node. If a card is removed from a router and inserted into a different router, the card assumes the OBFL configuration for the new router.

This section contains these procedures:

Enabling or Disabling OBFL

OBFL is enabled for all nodes by default and is active until disabled for a specified node or for all nodes.

Caution Do not deactivate OBFL without specific reasons since the OBFL data is used to diagnose and resolve problems in FRUs.

There are no configuration requirements other than to enable and disable OBFL.

SUMMARY STEPS

1. **admin**
2. **configure**
3. **hw-module** {**all** | **subslot** *node-id*} **logging onboard** [**disable** | **severity** {**alerts** | **emergencies**}]
4. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**admin** **Example:** `RP/0/RSP0/CPU0:router# admin`	Enters administration EXEC mode.

Step 2	**configure**
	Example:
	`RP/0/RSP0/CPU0:router(admin)# configure`
	`RP/0/RSP0/CPU0:router(admin-config)#`
		Enters administration configuration mode.

Step 3	**hw-module** {**all**	**subslot** *node-id*} **logging onboard** [**disable**	**severity** {**alerts**	**emergencies**}] **Example:** `RP/0/RSP0/CPU0:router(admin-config)# hw-module all logging onboard severity alerts`	Configures the severity level for the syslog messages that are logged into the OBFL storage device. • Use the **severity** keyword to specify the severity for the syslog message that is logged into the OBFL storage device. • Use the **alerts** keyword to specify that both emergency and alert syslog messages are logged. The default is the **alerts** keyword.

	Command or Action	Purpose
		• Use the **emergencies** keyword to specify only the emergency syslog messages are logged.
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 Message Severity Levels

Perform this task to configure message severity levels.

SUMMARY STEPS

1. **admin**
2. **configure**
3. **hw-module** {**all** | **subslot** *node-id*} **logging onboard** [**disable** | **severity** {**alerts** | **emergencies**}]
4. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**admin** **Example:** `RP/0/RSP0/CPU0:router# admin`	Enters administration EXEC mode.

Step 2	**configure**
	Example:
	`RP/0/RSP0/CPU0:router(admin)# configure`
	`RP/0/RSP0/CPU0:router(admin-config)#`
		Enters administration configuration mode.

Step 3	**hw-module** {**all**	**subslot** *node-id*} **logging onboard** [**disable**	**severity** {**alerts**	**emergencies**}] **Example:** `RP/0/RSP0/CPU0:router(admin-config)# hw-module all logging onboard severity alerts`	Configures the severity level for the syslog messages that are logged into the OBFL storage device. • Use the **severity** keyword to specify the severity for the syslog message that is logged into the OBFL storage device.

	Command or Action	Purpose
		• Use the **alerts** keyword to specify that both emergency and alert syslog messages are logged. The default is the **alerts** keyword. • Use the **emergencies** keyword to specify only the emergency syslog messages are logged.
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.

Monitoring and Maintaining OBFL

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

SUMMARY STEPS

1. **admin**
2. **show logging onboard** [**all** | *cbc* {**dump-all** | **dump-range** {*start-address* | *end-address* | **most-recent** {**fans** *fan-tray-slot* | [**location** *node-id*]} | **diagnostic** | **environment** | **error** | **temperature** | **uptime** | **verbose** | **voltage**] [**continuous** | **historical** | **static-data**] [**detail** | **raw** | **summary**] [**location** *node-id*]
3. **show processes include obfl**
4. **show running-config**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**admin** **Example:** `RP/0/RSP0/CPU0:router# admin`	Enters administration EXEC mode.
Step 2	**show logging onboard** [**all**	*cbc* {**dump-all**	**dump-range** {*start-address*	*end-address*	**most-recent** {**fans** *fan-tray-slot*	[**location** *node-id*]}	**diagnostic**	**environment**	**error**	**temperature**	**uptime**	**verbose**	**voltage**] [**continuous**	**historical**	**static-data**] [**detail**	**raw**	**summary**] [**location** *node-id*]	Displays stored OBFL data for all nodes or for a specified node. See the *Onboard Failure Logging Commands* module in the *System Monitoring Command Reference for Cisco ASR 9000 Series Routers*.

	Command or Action	Purpose
	Example: `RP/0/RSP0/CPU0:router(admin)# show logging onboard uptime`	
Step 3	**show processes include obfl** **Example:** `RP/0/RSP0/CPU0:router# show processes include obfl`	Confirms that the OBFL environmental monitor process is operating.
Step 4	**show running-config** **Example:** `RP/0/RSP0/CPU0:router# show running-config`	Displays the status of OBFL configuration.

Clearing OBFL Data

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

clear logging onboard [**all** | **cbc** {**dump-all** | **dump-range** {*start-address* | *end-address* | **most-recent** {**fans** *fan-tray-slot* | [**location** *node-id*]} | **corrupted-files** | **diagnostic** | **environment** | **error** | **poweron-time** | **temperature** | **uptime** | **voltage**] [**location** *node-id*]

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

Caution If OBFL is actively running on a card, issuing the **clear logging onboard** command can result in a corrupt or incomplete log at a later point in time. OBFL should always be disabled before this command is issued.

For more information, see the *Onboard Failure Logging Commands* module in the *System Monitoring Command Reference for Cisco ASR 9000 Series Routers*.

Configuration Examples for OBFL

This section provides these configuration examples:

Enabling and Disabling OBFL: Example

The following example shows how to disable OBFL:

```
RP/0/RSP0/CPU0:router(admin-config)# hw-module all logging onboard disable
```

The following example shows how to enable OBFL again:

```
RP/0/RSP0/CPU0:router(admin-config)# no hw-module all logging onboard disable
```

The following example shows that OBFL is enabled and message severity level is reset to the default:

```
RP/0/RSP0/CPU0:router(admin-config)# no hw-module all logging onboard
```

Configuring Message Severity Levels: Example

The following example shows how to save only the syslog message in which the severity level is set to 0 (emergency) to a storage device:

```
RP/0/RSP0/CPU0:router(admin-config)# hw-module subslot 0/2/CPU0 logging onboard severity
emergencies
```

The following example shows how to save the syslog message in which the severity level is set to 0 (emergency) and 1 (alert) to a storage device:

```
RP/0/RSP0/CPU0:router(admin-config)# hw-module subslot 0/2/CPU0 logging onboard severity
            alerts
```

Clearing OBFL Messages: Example

In the following example, all OBFL messages are cleared for all nodes in the system:

```
RP/0/RSP0/CPU0:router(admin)# clear logging onboard
```

Displaying OBFL Data: Example

The following example shows how to display uptime information from the OBFL feature:

```
RP/0/RSP0/CPU0:router(admin)# show logging onboard uptime detail location 0/7/cpu0

--------------------------------------------------------------------------------
UPTIME CONTINUOUS DETAIL INFORMATION (Node: node0_7_CPU0)
--------------------------------------------------------------------------------
The first record     : 01/05/2009 00:58:41
The last record      : 01/17/2007916:07:13
Number of records    :        478
File size            :      15288 bytes
Current reset reason : 0x00
Current uptime       :    0 years  0 weeks 0 days  3 hours  0 minutes
--------------------------------------------------------------------------------
Time Stamp          |
MM/DD/YYYY HH:MM:SS | Users operation
--------------------------------------------------------------------------------
01/05/2009 01:44:35   File cleared by user request.
--------------------------------------------------------------------------------
```

Where to Go Next

To configure alarm log correlation, see the *Implementing and Monitoring Alarms and Logging Correlation* module in the *System Monitoring Configuration Guide for Cisco ASR 9000 Series Routers*.

Additional References

The following sections provide references related to implementing logging services on Cisco IOS XR software .

Related Documents

Related Topic	Document Title
Logging services command reference	*Logging Services Commands* module in the *System Monitoring Command Reference for Cisco ASR 9000 Series Routers*
Onboard Failure Logging (OBFL) configuration	*Onboard Failure Logging Commands* module in the *System Monitoring Configuration Guide for Cisco ASR 9000 Series Routers*.
Onboard Failure Logging (OBFL) commands	*Onboard Failure Logging Commands* module in the *System Monitoring Command Reference for Cisco ASR 9000 Series Routers*.
Alarm and logging correlation commands	*Alarm Management and Logging Correlation Commands* module in the *System Monitoring Command Reference for Cisco ASR 9000 Series Routers*.
Alarm and logging correlation configuration and monitoring tasks	*Implementing and Monitoring Alarms and Alarm Log Correlation* module in the *System Monitoring Configuration Guide for Cisco ASR 9000 Series Routers*.
SNMP commands	*SNMP Commands* module in the *System Monitoring Command Reference for Cisco ASR 9000 Series Routers*.
SNMP configuration tasks	*Implementing SNMP* module in the *System Monitoring Configuration Guide for Cisco ASR 9000 Series Routers*
Cisco IOS XR getting started material	*Cisco ASR 9000 Series Aggregation Services Router Getting Started Guide*
Information about user groups and task IDs	*Configuring AAA Services* module in the *System Security Command Reference for Cisco ASR 9000 Series Routers*.

Standards

Standards	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	—

MIBs

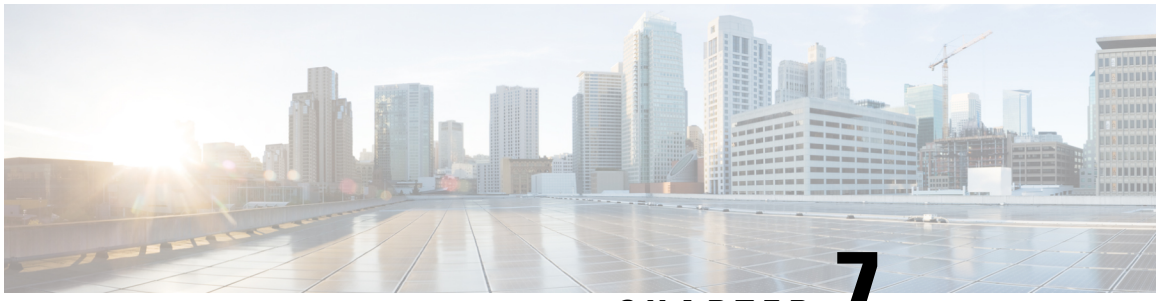
MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs	Title
No new or modified RFCs are supported by this feature, and support for existing RFCs has not been modified by this feature.	—

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	*http://www.cisco.com/cisco/web/support/index.html*



CHAPTER 7

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.

Note For more information about PM on the Cisco IOS XR Software and complete descriptions of the PM commands listed in this module, you can refer to the Related Documents, on page 353 section of this module.

Feature History for Implementing Performance Management

Release	Modification
Release 3.7.2	This feature was introduced.
Release 4.0.1	Support for **interface basic-counters** keyword was added. Configuring local disk dump for PM statistics collections and configuring instance filtering by regular-expression sections were added.

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.

  For detailed information about optional PIE installation, refer to the *Cisco ASR 9000 Series Aggregation Services Router Getting Started Guide*.

- 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

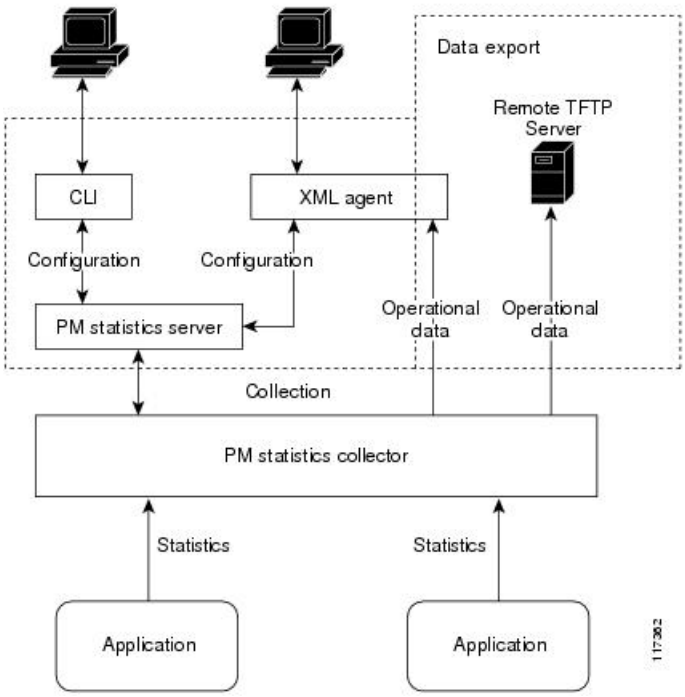
The PM statistics server is the front end for statistic collections, entity instance monitoring collections, and threshold monitoring. All PM statistic collections and threshold conditions configured through the command-line interface (CLI) or through XML schemas are processed by the PM statistics server 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 8: PM Component Communications, on page 319 illustrates the relationship between the components that constitute the PM system.

Figure 8: 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 34: 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 42: Attributes and Values, on page 332.

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.

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.

Guidelines for Creating PM Statistics Collection Templates

When creating PM statistics collection templates, follow these guidelines:

- Use the **performance-mgmt statistics** command to create a PM statistics collection 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.

- When configuring a template, you must name the template. You can designate the template for the entity as the default template using the **default** keyword or name the template with the **template** keyword and *template-name* argument. The default template contains the following default settings:
  - A sample interval of 10 minutes.
  - A sample size of five sampling operations.
- Configure the settings for the sample interval and sample size in the template.
  - 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** keyword and *minutes* argument. The range is from 1 to 60 minutes. The default is 10 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** keyword and *minutes* argument. The range is from 1 to 60 samples. The default is five samples.
- 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). For example, suppose that the sample interval is set at a frequency of 10 minutes, and the sample size is set to five sampling operations. Given that, a total of five sampling operations would be performed at a frequency of one sampling operation every 10 minutes. This cycle is referred to as the sampling cycle. A binary file containing the data collected from those samples would be exported to the TFTP server once every 50 (5 x 10) minutes. This cycle is referred to as the export cycle.

Caution 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.

Guidelines for Enabling and Disabling PM Statistics Collection Templates

When enabling PM statistics collection templates, follow these guidelines:

- Use the **performance-mgmt apply statistics** command to enable a PM statistics collection template.
- Only one PM statistics collection template for a given entity can be enabled at a time.

Note Data collection will begin one sampling cycle after you enable the PM statistics collection template with the **performance-mgmt enable statistics** command.

- 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 location with the **location** keyword and *node-id* argument or the **location all** keywords when enabling or disabling a PM statistic collections for the following entities:
  - Node CPU

- Node memory
- Node process

The **location** keyword with the *node-id* argument enables the PM statistic collections for the specified node. The *node-id* argument is expressed in the *rack/slot/module* notation. The **location all** keywords enable the PM statistic collections for all nodes.

- Because only one PM statistics collection can be enabled for any given entity at any given time, you are not required to specify the template name with the **default** keyword or **template** keyword and *template-name* argument when disabling a PM statistics collection.

Exporting Statistics Data

The current PM supports exporting of data onto the following:

- **performance-mgmt resource tftp-server** *ip-address* **directory** *directory-name*
- **performance-mgmt resource dump local**

You can dump PM statistics collections onto local filesystem, for example, /disk0: or /harddisk:. By default, this location is not configured but PM automatically selects the location on the local filesystem. Or, you can also configure a TFTP server for PM statistics collections and export the statistics data on the remote location.

Note Both the local and TFTP destinations are mutually exclusive and you can configure either one of them at a given time.

Binary File Format

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 35: Binary Format Values and Keys

Entity	Subentity	Key	StatsCounters
Node (1)	CPU (1)	CPU Key <Node ID>	See Table 36: Supported StatsCounters for Entities and Subentites, on page 324
	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 35: Binary Format Values and Keys, on page 323) 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 36: Supported StatsCounters for Entities and Subentites

Entity	Subentity	StatsCounters
Node (1)	CPU (1)	AverageCPUUsed, NoProcesses
	Memory (2)	CurrMemory, PeakMemory
	Process (3)	PeakMemory, AverageCPUUsed, 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 (3)	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
	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>

PM Entity Instance Monitoring Overview

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.

Note The data from entity instance monitoring can be retrieved through only a XML interface.

This table describes the command used to enable entity instance monitoring for the BGP entity instance.

Table 37: BGP Entity Instance Monitoring

Entity	Command Description
BGP	Use the **performance-mgmt apply monitor bgp** command in global configuration mode to enable entity instance monitoring for a BGP entity instance. **Syntax:** **performance-mgmt apply monitor bgp** *ip-address* *template-name* | **default**} `RP/0/RSP0/CPU0:router(config)# performance-mgmt apply monitor bgp 10.12.0.4 default`

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

Table 38: Interface Entity Instance Monitoring

Entity	Command Descriptions
Interface Data Rates	Use the **performance-mgmt apply monitor data-rates** command in global configuration mode 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/RSP0/CPU0:router(config)# performance-mgmt apply monitor interface data-rates gigabitethernet 0/2/0/0 default`
Interface Basic Counters	Use the **performance-mgmt apply monitor interface basic-counters** command in global configuration mode 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/RSP0/CPU0:router(config)# performance-mgmt apply monitor interface basic-counters gigabitethernet 0/2/0/0 default`

Entity	Command Descriptions
Interface Generic Counters	Use the **performance-mgmt apply monitor interface generic-counters** command in global configuration mode 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/RSP0/CPU0:router(config)# performance-mgmt apply monitor interface generic-counters gigabitethernet 0/2/0/0 default`

This table describes the command used to enable entity instance monitoring for the MPLS entity instances.

Table 39: MPLS Entity Instance Monitoring

Entity	Command Descriptions
MPLS LDP	Use the **performance-mgmt apply monitor mpls ldp** command in global configuration mode 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/RSP0/CPU0:router(config)# performance-mgmt apply monitor mpls ldp 10.34.64.154 default`

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

Table 40: Node Entity Instance Monitoring

Entity	Command Descriptions
Node CPU	Use the **performance-mgmt apply monitor node cpu** command in global configuration mode 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/RSP0/CPU0:router(config)# performance-mgmt apply monitor node cpu location 0/RP1/CPU0 default`
Node Memory	Use the **performance-mgmt apply monitor node memory** command in global configuration mode to enable an entity instance monitoring for a node memory entity instance. **Syntax:** **performance-mgmt** **apply** **monitor** **node** **memory** **location** *node-id* {*template-name* | **default**} `RP/0/RSP0/CPU0:router(config)# performance-mgmt apply monitor node memory location 0/RP1/CPU0 default`
Node Process	Use the **performance-mgmt apply monitor node process** command in global configuration mode to enable an 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/RSP0/CPU0:router(config)# performance-mgmt apply monitor node process location p 0/RP1/CPU0 275 default`

PM Threshold Monitoring Overview

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.

Guidelines for Creating PM Threshold Monitoring Templates

When creating a PM threshold template, follow these guidelines:

- Use the **performance-mgmt thresholds** command to create a PM threshold template.
- Specify entity for the *entity* argument.
- You can define multiple PM thresholds templates for an entity; however, note that at a time only one PM threshold template can be enabled.
- Specify a name for an entity's template when you configure it. You can designate the template as the default template using the **default** keyword, or you can name the template with the **template** keyword and *template-name* argument. The default setting for the default template is a sample interval of 10 minutes.
- Specify the attribute associated with the entity to be monitored for threshold violations, for the *attribute* argument.

Note For a list of the attributes associated with each entity, refer to Table 42: Attributes and Values, on page 332.

- Configure the sample interval for PM threshold monitoring with the **sample-interval** keyword and *interval* argument. The sample interval sets the frequency (in minutes) that the PM system waits before determining if any instances of the attribute match the threshold condition.
- Specify the threshold condition for the attribute (or attributes) that are to be monitored. A threshold condition consists of an attribute, an operation, and the threshold value. The threshold condition applies to all instances of the attribute.

Note A PM threshold template may contain multiple threshold conditions. You must define each threshold condition that is to be monitored and apply it to the specified template with the **performance-mgmt thresholds** command.

- Specify the operation to be performed in the threshold condition. The supported operations are as follows:
  - **EQ** —Equal to
  - **GE** —Greater than or equal to
  - **GT** —Greater than

- **LE** —Less than or equal to
- **LT** —Less than
- **NE** —Not equal to
- **RG** —Not in range

- Specify a value for the *value* argument. If you express the *value* argument, the PM system considers the threshold condition absolute, and after each sample interval determines whether any instance of the attribute matches the threshold condition. If you specify the *not in range* operation with the **RG** keyword, you must supply a pair of values that specify the range.
- If you specify the optional **percent** keyword, the *value* argument must be expressed as a percentage from 0 to 100. If you express the value as a percentage with the *value* argument and **percent** keyword, the threshold condition compares the value with the difference between the current and previous sample for each instance of attribute as a percentage.
- You can also specify the optional **rearm toggle** keywords or the optional **rearm window** keywords and *window-size* argument:
  - **rearm toggle**— Suppresses system logging messages for an instance of an attribute when an instance of the attribute matches the threshold condition. System logging messages for that instance of the attribute are suppressed in successive sample intervals until that instance of the attribute does not match the threshold condition.
  - **rearm window** *window-size*—Suppresses system logging messages for the number of intervals specified for the *window-size* argument when an instance of attribute matches the threshold condition.

Note For more information about how the PM system determines whether a threshold condition is met, refer to Table 41: How the PM System Determines if a Threshold Condition Is Met, on page 331.

This table describes how the PM system determines whether a threshold condition is met.

Table 41: How the PM System Determines if a Threshold Condition Is Met

If the threshold condition is composed of...	Then...
...an attribute, an operation, and a specific value,	The threshold condition is absolute because the PM system determines whether any instance of the attribute exactly matches the threshold condition after each sample interval elapses. • For example, suppose that a threshold condition for an entity is configured to check whether an attribute for an instance is greater than 2000. After the sample interval elapses, the PM system, accordingly, determines whether any instance of the attribute matches the condition. • The PM system generates a system logging message for each instance of the attribute that matches the threshold condition after the sample interval elapses. • If no instances of the attribute match the threshold condition, system logging messages are not generated for that sample interval.
...an attribute, an operation, and a value expressed as a percentage,	The threshold condition is relative because the threshold value that is used for comparison is taken as a percentage of the previous sample. • For example, suppose that a threshold condition for an entity is configured to check whether an attribute for an instance increases by more than 50 percent of the threshold value in the previous sample. Now, suppose that after the sample interval elapses, the value of an instance of the attribute is 250. Because the threshold condition is configured to generate a system logging message when any instance of the attribute is greater than 50 percent of the previous threshold value, the PM system would check to see whether that particular instance of the attribute is greater than 375 (250 + 125 [50 percent of 250]) in the following sample interval. **Note** The PM system matches the threshold condition against all instances of the attribute; therefore, the threshold value for this type of threshold condition is relative to the value of each instance of the attribute. • The PM system generates a system logging message for each instance of the attribute that matches the threshold condition after the sample interval elapses. • If no instances of the attribute match the threshold condition, system logging messages are not generated for that sample interval.

If the threshold condition is composed of...	Then...
...an attribute, an operation, a specific value, and the **rearm toggle** keywords...	The threshold condition is modified such that if an instance of an attribute matches the threshold condition, a system logging message is generated for that instance of the attribute, after the sample interval elapses. However, if the same instance of the attribute matches the threshold condition in successive sample intervals following the initial match, system logging messages for that instance of the attribute are suppressed until the instance does not match the threshold condition for a sample interval.
...an attribute, an operation, a specific value, and the **rearm window** keywords and *window-size* argument...	The threshold condition is modified such that if an instance of an attribute matches the threshold condition, a system logging message is generated. However, once an instance of the attribute matches the threshold condition, system logging messages for that instance of the attribute are suppressed for the number of intervals specified with the *window-size* argument.

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

Table 42: 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.

Entity	Attributes	Description	Values
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.
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.

Entity	Attributes	Description	Values
	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.

Entity	Attributes	Description	Values
	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.

Entity	Attributes	Description	Values
node cpu	AverageCPUUsed	Average percent CPU utilization.	Range is a percentage from 0 to 100.
	NoProcesses	Number of processes.	Range is from 0 to 4294967295.
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	AverageCPUUsed	Average percent CPU utilization.	Range is a percentage from 0 to 100.
	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.
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.

Entity	Attributes	Description	Values
	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.
	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.
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.

Entity	Attributes	Description	Values
	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.
	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.

Guidelines for Enabling and Disabling PM Threshold Monitoring Templates

When enabling PM threshold monitoring templates, follow these guidelines:

- Use the **performance-mgmt apply thresholds** command to enable a PM threshold monitoring template.
- 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 location with the **location** keyword and *node-id* argument or with **location all** keywords when enabling or disabling a PM threshold monitoring template for these entities:
  - Node CPU
  - Node memory
  - Node process

  The **location** keyword and *node-id* argument enables or disables PM statistic collections for the specified node. The *node-id* argument is expressed in the *rack/slot/module* notation. The **location all** keywords enable or disable the PM statistic collections for all nodes.
- Because only one PM threshold monitoring template for an entity at any given time, you are not required to specify the template name with the **default** keyword or **template** keyword and *template-name* argument when disabling a PM statistics collection.

How to Implement Performance Management

Configuring an External TFTP Server for PM Statistic Collections

This task explains how to configure an external TFTP server for PM statistic collections.

Note Perform this task before enabling a PM statistics collection template for PM statistic collections. For more information about enabling a PM statistics collection templates, see the Enabling and Disabling PM Statistics Collection Templates, on page 345 task.

Before you begin

You must have access to and connectivity with a TFTP server before performing this task.

SUMMARY STEPS

1. **configure**
2. **performance-mgmt resources tftp-server** *ip-address* **directory** *dir-name*
3. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.

	Command or Action	Purpose
Step 2	**performance-mgmt resources tftp-server** *ip-address* **directory** *dir-name* **Example:** `RP/0/RSP0/CPU0:router(config)# performance-mgmt resources tftp-server 10.3.40.161 directory mypmdata/datafiles`	Sets the IP address and the directory path for PM data collection. • Include the entire directory path name for the *dir-name* argument. **Note** Files copied to the TFTP server contain a timestamp in their name, which makes them unique. For that reason the TFTP server used should support creation of files as data is transferred, without requiring users to manually create them at the TFTP server host in advance.
Step 3	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 Local Disk Dump for PM Statistics Collections

This task explains how to configure local disk or external TFTP server for PM statistic collections.

SUMMARY STEPS

1. **configure**
2. **performance-mgmt resources dump local**
3. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**performance-mgmt resources dump local** **Example:**	Sets the local filesystem on which the statistics data is dumped.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config)# performance-mgmt resources dump local`	**Note** You can also dump the statistics data on the TFTP server location. However, the configuration is rejected if you configure both local dump and TFTP server at the same time.
Step 3	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 Instance Filtering by Regular-expression

This task explains how to apply a defined regular expression group to one or more statistics or threshold templates. You can also define a regular expression group that includes multiple regular expression indices.

The benefits of instance filtering using the regular expression group is:

- 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.

SUMMARY STEPS

1. **configure**
2. **performance-mgmt regular-expression** *regular-expression name*
3. **index** *index-number regular-expression-string*
4. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**performance-mgmt regular-expression** *regular-expression name*	Sets a defined regular expression group to one or more statistics or threshold template.

	Command or Action	Purpose
	Example: `RP/0/RSP0/CPU0:router(config)# performance-mgmt regular-expression regexp`	**Note** By default, no regular expression group is configured. Once the regular expression group is configured, you can apply it to multiple templates.
Step 3	**index** *index-number regular-expression-string* **Example:** `RP/0/RSP0/CPU0:router(config-perfmgmt-regex)# index 10 match`	Specifies a regular expression index to the defined group. **Note** The Instance filtering by regular-expression is currently supported in interface entities only (Interface basic-counters, generic-counters, data-rates.
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.

Creating PM Statistics Collection Templates

This task explains how to create a PM statistics collection template.

SUMMARY STEPS

1. **configure**
2. **performance-mgmt statistics** *entity* {**default** | **template** *template-name*} [**sample-size** *size*] [**sample-interval** *minutes*]
3. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**performance-mgmt statistics** *entity* {**default**	**template** *template-name*} [**sample-size** *size*] [**sample-interval** *minutes*] **Example:**	Creates a PM statistics collection template for the specified entity. • Use the *entity* argument to specify the entity for which you want to create a PM statistics collection template.

	Command or Action	Purpose
	`RP/0/RSP0/CPU0:router(config)# performance-mgmt statistics interface data-rates default`	• Use the **default** keyword to apply the default template to the PM statistics template for the specified entity. The default template contains a default sample interval of 10 minutes and a default sample size of 5 sampling operations. • Use the **template** keyword and *template-name* argument to designate a unique name for a template. • The **sample-size** keyword and *size* argument set the number of sampling operations to be performed before exporting the data to the TFTP server. The range is from 1 to 60 samples. The default is 5 samples. • The **sample-interval** keyword and *minutes* argument set the frequency of the sampling operations performed during the sampling cycle. The range is from 1 to 60 minutes. The default is 10 minutes. **Note** For more information about creating PM collection templates, see the Guidelines for Creating PM Statistics Collection Templates, on page 320 section.
Step 3	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 creating a PM statistics collection template, you must enable the template to start the PM statistics collection. See the Enabling and Disabling PM Statistics Collection Templates, on page 345 section for more information about enabling PM statistics collection templates.

Enabling and Disabling PM Statistics Collection Templates

This task explains how to enable and disable PM statistics collection templates.

Before you begin

You must create a PM statistics collection template before performing this task, or you can use a predefined template (default). 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.

Refer to the Configuring an External TFTP Server for PM Statistic Collections, on page 341 and Creating PM Statistics Collection Templates, on page 344 tasks for more information.

SUMMARY STEPS

1. **configure**
2. Do one of the following:
   - **performance-mgmt apply statistics** {*entity* | **interface** {**basic-counters** | **data-rates** | **generic-counters**} *type interface-path-id* } [**location** {**all** | *node-id*}] {*template-name* | **default**}
   - **no performance-mgmt apply statistics** {*entity* | **interface** {**basic-counters** | **data-rates** | **generic-counters**} *type interface-path-id* } [**location** {**all** | *node-id*}]
3. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.

Step 2	Do one of the following:
	• **performance-mgmt apply statistics** {*entity* | **interface** {**basic-counters** | **data-rates** | **generic-counters**} *type interface-path-id* } [**location** {**all** | *node-id*}] {*template-name* | **default**}
	• **no performance-mgmt apply statistics** {*entity* | **interface** {**basic-counters** | **data-rates** | **generic-counters**} *type interface-path-id* } [**location** {**all** | *node-id*}]
	Example:
	`RP/0/RSP0/CPU0:router(config)# performance-mgmt apply statistics mpls ldp default`
	or
	`RP/0/RSP0/CPU0:router(config)# no performance-mgmt apply statistics mpls ldp`
		Enables or disables a PM statistics collection template.
		• Only one PM statistics collection template for a given entity can be enabled at a time.
		• You must specify either a location with the **location** keyword and *node-id* argument or the **location all** keywords when enabling a PM statistic collections for these entities:
		• Node CPU
		• Node memory
		• Node process
		The **location** keyword with the *node-id* argument enables PM statistic collections for the specified node. The *node-id* argument is expressed in the *rack*/*slot*/*module* notation. The **location all** keywords enable a PM statistic collection for all nodes.
		• Because only one PM statistics collection can be enabled for any given entity at any given time, you are not required to specify the template name with the **default** keyword or **template** keyword and

	Command or Action	Purpose
		template-name argument when disabling a PM statistics collection. **Note** Data collection will begin one sampling cycle after you enable the PM statistics collection template with the **performance-mgmt apply statistics** command. • When 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 location with the **location** keyword and *node-id* argument or the **location all** keywords when disabling a PM statistic collections for these entities: • Node CPU • Node memory • Node process The **location** keyword with the *node-id* argument disables PM statistic collections for the specified node. The *node-id* argument is expressed in the *rack*/*slot*/*module* notation. The **location all** keyword disables the PM statistic collections for all nodes. • Because only one PM statistics collection can be enabled for any given entity at any given time, you are not required to specify the template name with the **default** keyword or **template** keyword and *template-name* argument when disabling a PM statistics collection.
Step 3	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.

Enabling PM Entity Instance Monitoring

This task explains how to enable entity instance monitoring.

Before you begin

You must create PM statistics collection template for an entity before performing this task.

SUMMARY STEPS

1. **configure**
2. **performance-mgmt apply monitor** {*entity instance* | **interface** {**basic-counters** | **data-rates** | **generic-counters**} *type interface-path-id* } {*template-name* | **default**}
3. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**performance-mgmt apply monitor** {*entity instance*	**interface** {**basic-counters**	**data-rates**	**generic-counters**} *type interface-path-id* } {*template-name*	**default**} **Example:** `RP/0/RSP0/CPU0:router(config)# performance-mgmt apply monitor node cpu 0/RSP1/CPU0 default`	Enables entity instance monitoring for the specified instance. • Use the *entity* and *instance* arguments to specify the name of the entity and the instance to be monitored, respectively. • Use either the **default** keyword or the *template-name* argument to specify the template associated with the entity instance to be monitored.
Step 3	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 PM Threshold Monitoring Templates

This task explains how to create a PM threshold monitoring template.

SUMMARY STEPS

1. **configure**
2. **performance-mgmt thresholds** {*entity* | **interface** {**basic-counters** | **data-rates** | **generic-counters**} *type interface-path-id* } {**template** *name* } *attribute operation value* [*value2*] [**percent**] [**rearm** {**toggle** | **window** *window-size*}]
3. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.

Step 2	**performance-mgmt thresholds** {*entity*	**interface** {**basic-counters**	**data-rates**	**generic-counters**} *type interface-path-id* } {**template** *name* } *attribute operation value* [*value2*] [**percent**] [**rearm** {**toggle**	**window** *window-size*}]
					Example:
					`RP/0/RSP0/CPU0:router(config)# performance-mgmt thresholds node cpu template cpu_thresh1`
					`RP/0/RSP0/CPU0:router(config-threshold-bgp)# AverageCPUUsed GT 25 percent`
						Creates a PM threshold monitoring template.
						Note For more detailed information about creating PM threshold monitoring templates, see the Guidelines for Creating PM Threshold Monitoring Templates, on page 329 section.

Step 3	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 creating a PM threshold monitoring template, you must enable the template to start PM threshold monitoring. Refer to the Enabling and Disabling PM Threshold Monitoring Templates, on page 349 task for more information about enabling PM statistics threshold monitoring templates.

Enabling and Disabling PM Threshold Monitoring Templates

This task explains how to enable and disable PM threshold monitoring templates.

Before you begin

You must create a PM threshold template before performing this task. Refer to Creating PM Threshold Monitoring Templates, on page 348 tasks for more information.

SUMMARY STEPS

1. **configure**
2. Do one of the following:
   - **performance-mgmt apply thresholds** {*entity* | **interface** {**basic-counters** | **data-rates** | **generic-counters**} *type interface-path-id* } [**location** {**all** | *node-id*}] {*template-name* | **default**}
   - **no performance-mgmt apply thresholds** {*entity* | **interface** {**basic-counters** | **data-rates** | **generic-counters**} *type interface-path-id* } [**location** {**all** | *node-id*}]
3. Use the **commit** or **end** command.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters global configuration mode.

Step 2	Do one of the following:
	• **performance-mgmt apply thresholds** {*entity* | **interface** {**basic-counters** | **data-rates** | **generic-counters**} *type interface-path-id* } [**location** {**all** | *node-id*}] {*template-name* | **default**}
	• **no performance-mgmt apply thresholds** {*entity* | **interface** {**basic-counters** | **data-rates** | **generic-counters**} *type interface-path-id* } [**location** {**all** | *node-id*}]
	Example:
	`RP/0/RSP0/CPU0:router(config)# performance-mgmt enable thresholds node cpu location all template20`
	or
	`RP/0/RSP0/CPU0:router(config)# no performance-mgmt apply thresholds node cpu location all`
		Enables or disables PM threshold monitoring templates for the specified template.
		• Only one PM threshold monitoring template for an entity can be enabled at a time.
		• You must specify either a location with the **location** keyword and *node-id* argument or the **locationall** keywords when enabling a PM threshold monitoring template for these entities:
		• Node CPU
		• Node memory
		• Node process
		The **location** keyword with the *node-id* argument enables the PM threshold monitoring template for the specified node. The *node-id* argument is expressed in the *rack/slot/module* notation. The **location all** keywords enable the PM threshold monitoring template for all nodes.
		• Because only one PM threshold monitoring template for an entity at any given time, you are not required to specify the template name with the **default** keyword or **template** keyword and *template-name* argument when disabling a PM statistics collection.

	Command or Action	Purpose

		• Once a template has been enabled, threshold monitoring continues until the template is disabled with the **no** form of the **performance-mgmt apply thresholds** command.
		• You must specify either a location with the **location** keyword and *node-id* argument or the **location all** keywords when disabling a PM threshold monitoring template for these entities:
		  • Node CPU
		  • Node memory
		  • Node process
		The **location** keyword with the *node-id* argument disables the PM threshold monitoring template for the specified node. The *node-id* argument is expressed in the *rack*/*slot*/*module* notation. The **location all** keywords disable the PM threshold monitoring template for all nodes.
		• Because only one PM threshold monitoring template for an entity can be enabled at a time, you are not required to specify the template name with **default** keyword or *template-name* argument when disabling a PM statistics collection.

Step 3	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 Performance Management

This section provides these configuration examples:

Creating and Enabling PM Statistics Collection Templates: Example

This example shows how to configure the TFTP server resource, and how to create and enable a PM statistics collection templates. In this example, the following PM template collection templates are created and enabled:

- A template named template1 with a sample size of 10 and a sample interval of 5 for the interface generic counters entity.
- A template named template2 with a sample size of 30 and a sample interval of 2 for the node memory entity. The template is enabled globally.
- A template name template3 with a sample size of 10 and a sample interval of 5 for the node process entity. The template is enabled for node 0/0/CPU0.

```
performance-mgmt resources tftp-server 10.30.62.154 directory pm/pm_data/pmtest
performance-mgmt statistics interface generic-counters template template1
 sample-size 10
 sample-interval 5
!
performance-mgmt statistics node memory template template2
 sample-size 30
 sample-interval 2
!
performance-mgmt statistics node process template template3
 sample-size 10
 sample-interval 5
!
performance-mgmt apply statistics interface generic-counters template1
performance-mgmt apply statistics node memory global template2
performance-mgmt apply statistics node process 0/0/CPU0 template3
```

Creating and Enabling PM Threshold Monitoring Templates: Example

This example shows how to create and enable a PM threshold monitoring template. In this example, a PM threshold template is created for the AverageCpuUsed attribute of the node CPU entity. The threshold condition in this PM threshold condition monitors the AverageCpuUsed attribute to determine whether the average CPU use is greater than 75 percent. The sample interval for the template is set to 5 minutes, and the template is enabled globally.

```
performance-mgmt thresholds node cpu template template20
 AverageCpuUsed GT 75
 sample-interval 5
!
performance-mgmt apply thresholds node cpu global template20
```

Additional References

The following sections provide references related to implementing performance management.

Related Documents

Related Topic	Document Title
Performance management commands	Performance Management Commands on the Cisco ASR 9000 Series Router module in the *System Monitoring Command Reference for Cisco ASR 9000 Series Routers*
Cisco IOS XR Software XML API material	*Cisco IOS XR XML API Guide*
Cisco IOS XR Software getting started material	*Cisco ASR 9000 Series Aggregation Services Router Getting Started Guide*
Information about user groups and task IDs	Configuring AAA Services on the Cisco ASR 9000 Series Router module in the *System Security Configuration Guide for Cisco ASR 9000 Series Routers*

Standards

Standards	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	—

MIBs

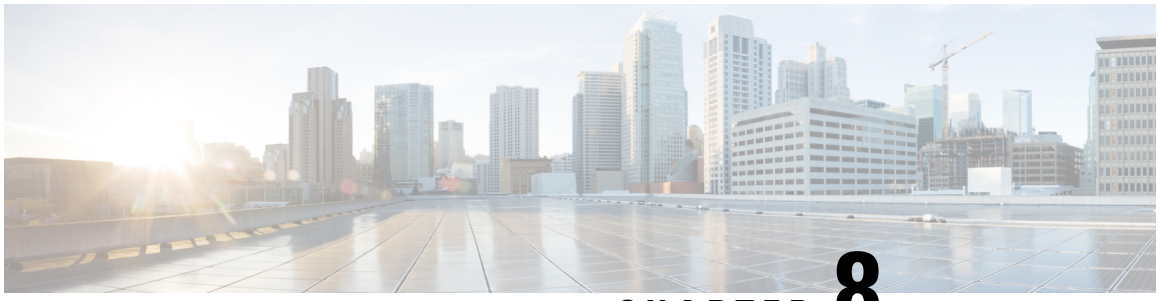
MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs	Title
No new or modified RFCs are supported by this feature, and support for existing RFCs has not been modified by this feature.	—

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	*http://www.cisco.com/cisco/web/support/index.html*



CHAPTER 8

Testing Throughput Using Test TCP (TTCP)

The Test TCP utility (TTCP) is used to measure TCP throughput through an IP path. This utility is effective in determining the actual bit rate of a particular WAN or modem connection. This feature is also used to test the connection speed between any two devices with IP connectivity between them.

For information on the commands for configuring TTCP, see the *TTCP Commands* module in the *Cisco ASR 9000 Series Aggregation Services Router System Monitoring Command Reference* guide.

Feature History for Implementing TTCP Utility

Release	Modification
Release 5.2.2	This feature was introduced.

• Using Test TCP (TTCP) to Test Throughput, on page 355

Using Test TCP (TTCP) to Test Throughput

You can use the Test TCP utility (TTCP) to measure TCP throughput through an IP path. In order to use it, start the receiver on one side of the path, then start the transmitter on the other side. The transmitting side sends a specified number of TCP packets to the receiving side. At the end of the test, the two sides display the number of bytes transmitted and the time elapsed for the packets to pass from one end to the other. You can then use these figures to calculate the actual throughput on the link.

Since it is most common to evaluate connect speeds in kbps (kilobits per second, or 1000 bits per second) rather that kbps (kilobytes per second, or 1024 bytes per second), we must use the information from TTCP to calculate the bit rate (in kbps). Use the number of bytes received and the transfer time to calculate the actual bit rate for the connection. Calculate the bit rate by converting the number of bytes into bits and then divide this by the time for the transfer. For example, if the host received 409600 bytes in 84.94 seconds, you can calculate the bit rate to be (409600 bytes * 8 bits per byte) divided by 84.94 seconds=38577 bps or 38.577 kbps.