Preface

Revised: November 24, 2010, OL-17665-04

This preface describes the objectives and organization of this document and explains how to find additional information on related products and services. This preface contains the following sections:

Objectives

This document provides operations and maintenance information that is specific to the Cisco ASR 1000 Series Aggregation Services Routers. It does not repeat operations information that is standard for all Cisco routers, such as setting up a syslog server to monitor alarms and other messages sent to the system console.

Document Revision History

This Document Revision History table records technical changes to this document.

Release Number	Date	Change Summary
Cisco IOS XE 2.2	October, 2008	Initial publication, including the following chapters: • Verifying Hardware Installation • Automatic Shutdown • Monitoring Hardware Using Alarms
Cisco IOS XE 2.2	December, 2008	Modified the number of minutes from two to five for the router to shut down when a fan fails, per CSCsr59868. "Automatic Shutdown" chapter.
Cisco IOS XE 2.4	June, 2009	Added the following chapters: • Monitoring the Control Plane • Performing File System Cleanups • Upgrading System Software
Cisco IOS XE 3.2S	November, 2010	Added the following chapter: Configuring the Common Criteria Tcl Scripts

Audience

This document is intended for network operators who monitor and maintain networks for Cisco enterprise and service provider customers. Users of this document need a broad understanding of networks in general, networking principles, network configuration, and routing protocols.

Organization

This document contains the following sections:

Chapter	Title	Description
1	Verifying Hardware Installation	Using LEDs and show commands to verify successful installation, and what to check if installation is unsuccessful.
2	Automatic Shutdown	Conditions under which the router and power supplies automatically shut down.
3	Monitoring Hardware Using Alarms	Using visual alarms, audible alarms, alarm messages sent to the console or syslog, and SNMP alarm notification to monitor hardware.
4	Configuring the Common Criteria Tcl Scripts	Configuring the Common Criteria Tcl scripts to monitor the packet drop event on the ASR 1000 Series Router.

Chapter	Title	Description
5	Monitoring the Control Plane	Verifying the overall health of the system by monitoring control plane resources.
6	Monitoring File Systems	Maintaining proper router operation by performing cleanups of core, trace, crashinfo, and sub-package files .
7	Upgrading System Software	Upgrading software packages, including offline and in-service software upgrades. (Referred to the appropriate chapters in the *Cisco ASR 1000 Series Aggregation Services Routers Software Configuration Guide.*)

Document Conventions

This documentation uses the following conventions:

Convention	Description
^ or **Ctrl**	The **^** and **Ctrl** symbols represent the Control key. For example, the key combination **^D** or **Ctrl-D** means hold down the **Control** key while you press the **D** key. Keys are indicated in capital letters but are not case sensitive.
string	A string is a nonquoted set of characters shown in italics. For example, when setting an SNMP *community* string to *public*, do not use quotation marks around the string or the string will include the quotation marks.

Command syntax descriptions use the following conventions:

Convention	Description
bold	Bold text indicates commands and keywords that you enter exactly as shown.
italics	Italic text indicates arguments for which you supply values.
[x]	Square brackets enclose an optional element (keyword or argument).
|	A vertical line indicates a choice within an optional or required set of keywords or arguments.
[x | y]	Square brackets enclosing keywords or arguments separated by a vertical line indicate an optional choice.
{x | y}	Braces enclosing keywords or arguments separated by a vertical line indicate a required choice.

Nested sets of square brackets or braces indicate optional or required choices within optional or required elements. For example:

Convention	Description
[x {y | z}]	Braces and a vertical line within square brackets indicate a required choice within an optional element.

Examples use the following conventions:

Convention	Description
`screen`	Examples of information displayed on the screen are set in Courier font.
`bold screen`	Examples of text that you must enter are set in Courier bold font.
< >	Angle brackets enclose text that is not printed to the screen, such as passwords.
!	An exclamation point at the beginning of a line indicates a comment line. (Exclamation points are also displayed by the Cisco IOS software for certain processes.)
[ ]	Square brackets enclose default responses to system prompts.

The following conventions are used to attract the attention of the reader:

Caution Means *reader be careful*. In this situation, you might do something that could result in equipment damage or loss of data.

Note Means *reader take note*. Notes contain helpful suggestions or references to materials that may not be contained in this manual.

Obtaining Documentation and Submitting a Service Request

For information on obtaining documentation, submitting a service request, and gathering additional information, see the monthly *What's New in Cisco Product Documentation*, which also lists all new and revised Cisco technical documentation, at:

http://www.cisco.com/en/US/docs/general/whatsnew/whatsnew.html

Subscribe to the *What's New in Cisco Product Documentation* as a Really Simple Syndication (RSS) feed and set content to be delivered directly to your desktop using a reader application. The RSS feeds are a free service and Cisco currently supports RSS Version 2.0.

CHAPTER 1

Automatic Shutdown

Under certain conditions, the Cisco ASR 1000 Series Aggregation Services Router or one of its power supplies can perform an automatic shutdown.

This chapter includes the following sections:

Automatic Router Shutdown

When the router detects a condition that could result in physical damage to system components, the router can shut down without operator intervention. When the router shuts down automatically, the system controller disables DC power to all internal components. All DC power remains disabled until you toggle the power switch.

The default for automatic router shutdown is off. To allow automatic router shutdown, the **facility-alarm critical exceed-action shutdown** command must be enabled. If the **facility-alarm critical exceed-action shutdown** command is enabled, the router performs an automatic shutdown under the following conditions:

Internal Temperature of Router or Power Supply Exceeds Temperature Threshold

A temperature threshold is exceeded if any of the following conditions occur:

- The internal temperature of the router (the ambient air temperature on the active Cisco ASR 1000 Series Route Processor) is over 100° C.
- The internal temperature of the AC power supply is over 100° C.
- The internal temperature of the DC power supply is over 100° C.

Note Temperature threshold values cannot be configured or changed.

Voltage of AC or DC Power Supplies Is Out of Tolerance

The voltage of a power supply must be within certain ranges (within tolerance). A power supply is out of tolerance if voltage is outside of the following ranges:

- AC input range: 85 VAC to 264 VAC
- DC input range: –40.5 VDC to –72 VDC

Note Voltage tolerance ranges cannot be configured or changed.

Automatic Power Supply Shutdown

Automatic power supply shutdown occurs independently of a router shutdown. If the internal temperature of a power supply exceeds 100° C, the power supply shuts down immediately. The **facility-alarm critical exceed-action shutdown** command does not need to be enabled.

Each power supply fail safe is independent of the other and independent of the router. The fans in the power supplies continue to operate as long as the second power entry module (PEM) is powering the system.

For More Information

For more information about the topics discussed in this chapter, see the following documents:

Topic	Document
Command descriptions	Cisco IOS Master Command List, All Releases Command Lookup Tool (Requires Cisco.com user ID and password)
Environmental monitoring and reporting	"Environmental Monitoring and Reporting" section in the "Cisco ASR 1000 Series Routers Hardware Overview" chapter in the *Cisco ASR 1000 Series Router Hardware Installation Guide*

CHAPTER 1

Verifying Hardware Installation

After installing the Cisco ASR 1000 Series Aggregation Services Router or replacing any of its hardware components that are field-replaceable units (FRUs), verify the installation.

This chapter includes the following sections:

Checking the LEDs

Check the LEDs on the faceplates of the following FRUs:

Cisco ASR 1000 Series Route Processors

Route processor LEDs vary according to the chassis model, as described in the following sections.

Cisco ASR 1013 Router

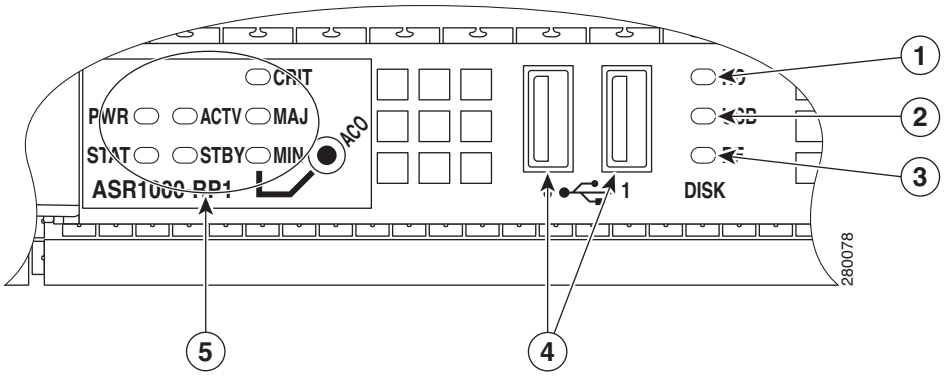
Table 1-1 shows the color or state of the LEDs in the Cisco ASR 1000 Series Route Processor-2 (RP-2) that indicate a successful installation. Figure 1-1 shows a view of the LEDs on the faceplate.

Note Only Route Processor-2 (RP-2) and ESP-40 (Embedded Service Processor) are supported for installation on the Cisco ASR 1013 Router.

Table 1-1 RP-2 Faceplate LEDs Indicating a Successful Installation (Cisco ASR 1013 Router)

LED Label	Color—State	Description
PWR	Solid green	All power requirements are within specification
	Off	Off indicates that the router is in standby mode.
STAT	Solid green	Cisco IOS has successfully booted.
	Yellow	BOOT ROM has successfully loaded.
	Red	System failure.
ACTV	Green	Lit when this is the active ASR 1000 Series route processor (Cisco ASR1000-RP1 or Cisco ASR1000-RP2).
STBY	Yellow	Lit when this is the standby ASR 1000 Series route processor.
CRIT	Solid red	Critical alarm indicator. This is on at power up, turned off by software.
MAJ	Solid red	Major alarm indicator.
MIN	Amber	Minor alarm indicator.
DISK HD	Flashing green	Active indicator.
	Off	No activity.
DISK USB	Flashing green	Active indicator.
	Off	No activity.
DISK BF	Flashing green	Active indicator.
	Off	No activity.

Figure 1-1 RP-2 Faceplate LEDs for an Active RP (Cisco ASR 1013 Router)



Cisco ASR 1001 Router

The Cisco ASR 1001 Router faceplate has common components for each type of ASR 1001 Router configuration. Figure 1-2 shows the Cisco ASR1000 front panel LEDs of the Cisco ASR 1001 Router. Table 1-2 shows the color or state of the LEDs in the Cisco ASR 1001 Series Router.

Figure 1-2 Common LEDs for Cisco ASR 1001 Router

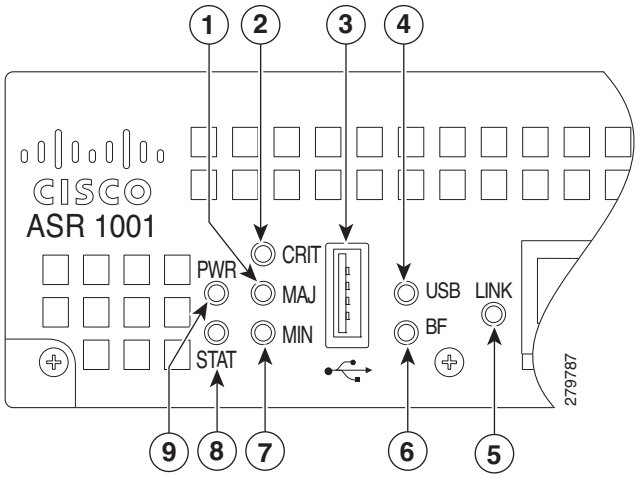


Table 1-2 Cisco ASR 1001 LED Color or State Details

LED Label	Color—State	Description
PWR	Solid green	Power requirements are within specification.
STAT	Solid green	Cisco IOS booted successfully.
MIN	Off	No minor alarms.
MAJ	Off	No major alarms.
CRIT	Off	No critical alarms.
BF	Green	Indicates activity of the EUSB device
Link	Green	Solid Green indicates Link, Flashing green indicates MGMT Ethernet port activity.
USB	Green	USB is green and flashes when accessed.

Cisco ASR 1004 Router, Cisco ASR 1006 Router

Table 1-3 shows the color or state of the LEDs in the Cisco ASR 1000 Series Route Processor (RP) that indicate a successful installation. Figure 1-3 shows a view of the LEDs on the faceplate.

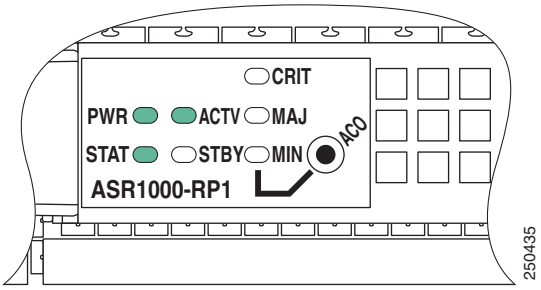
Table 1-3 RP LEDs Indicating a Successful Installation (Cisco ASR 1004 Router, Cisco ASR 1006 Router)

LED Label	Color—State	Description
PWR	Solid green	Power requirements are within specification.
STAT	Solid green	Cisco IOS booted successfully.
ACTV	Green	Active RP.
STBY	Yellow	Standby RP.
CRIT	Off	No critical alarms.

Table 1-3 RP LEDs Indicating a Successful Installation (Cisco ASR 1004 Router, Cisco ASR 1006 Router)

LED Label	Color—State	Description
MAJ	Off	No major alarms.
MIN	Off	No minor alarms.

Figure 1-3 RP Faceplate LEDs for an Active RP (Cisco ASR 1004 Router, Cisco ASR 1006 Router)



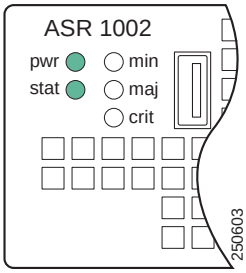
Cisco ASR 1002 Router

Table 1-4 shows the color or state of the LEDs in the Cisco ASR 1000 Series Route Processor (RP) that indicate a successful installation. Figure 1-4 shows a view of the LEDs on the faceplate.

Table 1-4 RP LEDs Indicating a Successful Installation (Cisco ASR 1002 Router)

LED Label	Color—State	Description
pwr	Solid green	Power requirements are within specification.
stat	Solid green	Cisco IOS booted successfully.
min	Off	No minor alarms.
maj	Off	No major alarms.
crit	Off	No critical alarms.

Figure 1-4 RP Faceplate LEDs for an Active RP (Cisco ASR 1002 Router)



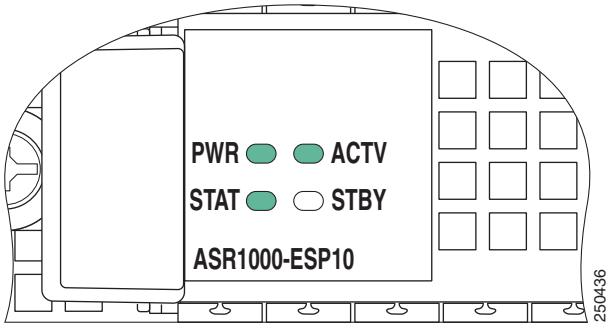
Cisco ASR 1000 Series Embedded Services Processors

Table 1-5 shows the color or state of the LEDs in the Cisco ASR 1000 Series Embedded Services Processor (ESP) that indicate a successful installation. Figure 1-5 shows a view of the LEDs on the faceplate.

Table 1-5 ESP LEDs Indicating a Successful Installation

LED Label	Color—State	Description
PWR	Solid green	Power requirements are within specification.
STAT	Solid green	Cisco IOS booted successfully.
ACTV	Green	Active ESP.
STBY	Yellow	Standby ESP.

Figure 1-5 ESP Faceplate LEDs for an Active ESP



Cisco ASR 1013 Router

Table 1-6 shows the color or state of the LEDs in the Cisco ASR 1000 Series SPA Interface Processors (SIP) that indicate a successful installation. Figure 1-6 shows a view of the LEDs on the faceplate.

Table 1-6 SIP LEDs Indicating a Successful Installation (Cisco ASR 1013 Router)

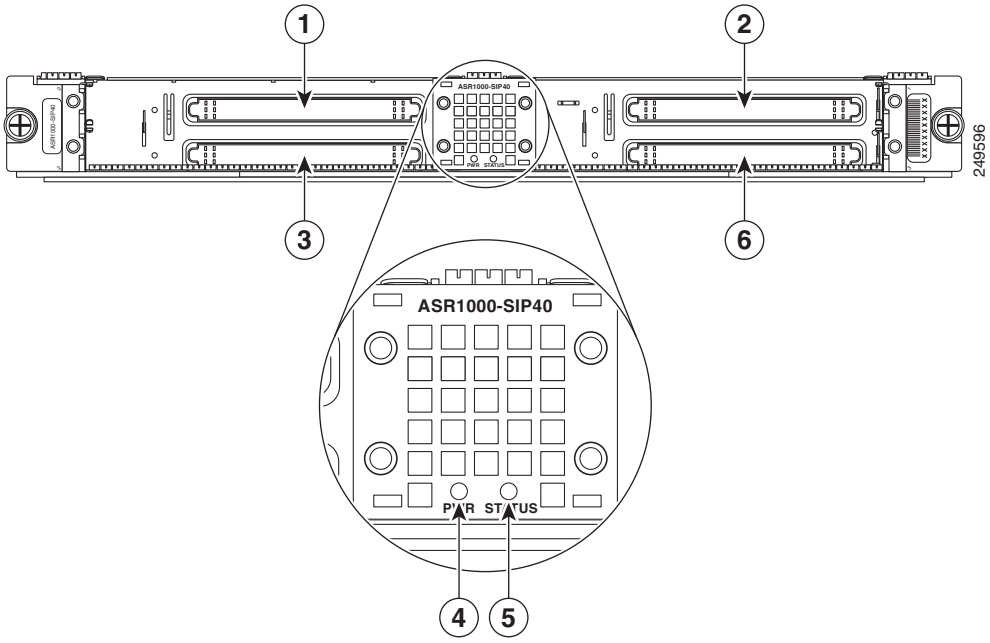
LED Label	Color—State	Description
PWR	Solid green	SIP is powered on.
STATUS	Solid green	SIP is online.

In the Cisco ASR 1013 Router, each Cisco ASR1000- SIP40 supports:

- Up to 6 ASR1000-SIP40G SIPs.
- Each SIP-40G supports:
  - Four half-height (¼ Rate or full rate or combination) SPAs with up to 24 ports per SPA
  - Two full-height (¼ Rate or full rate or combination) SPAs with up to 48 ports per SPA
  - Two half-height and 1 full-height combination that does not exceed 96 ports

Note If ASR-SIP10 is inserted in slot 0 to 5 of a Cisco ASR 1013 Router then you need to upgrade CPLD and ROMMON. If ASR-SIP40 is inserted in slot 4 or 5, it behaves like the ASR-SIP10.

Figure 1-6 SIP Faceplate LEDs (Cisco ASR 1013 Router)



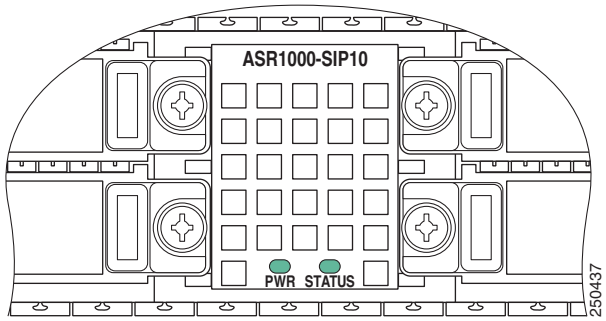
Cisco ASR 1004 Router, Cisco ASR 1006 Router

Table 1-7 shows the color or state of the LEDs in the Cisco ASR 1000 Series SPA Interface Processors (SIP) that indicate a successful installation. Figure 1-7 shows a view of the LEDs on the faceplate.

Table 1-7 SIP LEDs Indicating a Successful Installation (Cisco ASR 1004 Router, Cisco ASR 1006 Router)

LED Label	Color—State	Description
PWR	Solid green	SIP is powered on.
STATUS	Solid green	SIP is online.

Figure 1-7 SIP Faceplate LEDs (Cisco ASR 1004 Router, Cisco ASR 1006 Router)



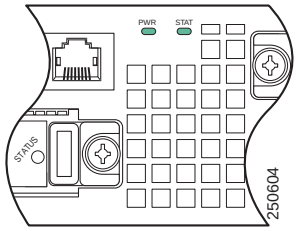
Cisco ASR 1002 Router

Table 1-8 shows the color or state of the LEDs in the Cisco ASR 1000 Series SPA Interface Processors (SIP) that indicate a successful installation. Figure 1-8 shows a view of the LEDs on the faceplate.

Table 1-8 SIP LEDs Indicating a Successful Installation (Cisco ASR 1002 Router)

LED Label	Color—State	Description
PWR	Solid green	SIP is powered on.
STAT	Solid green	SIP is online.

Figure 1-8 SIP Faceplate LEDs (Cisco ASR 1002 Router)



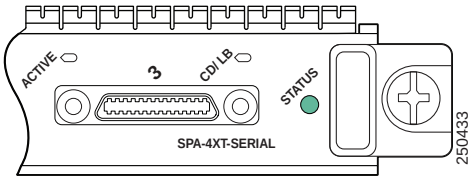
Shared Port Adapters

Table 1-9 shows the color or state of the LED the shared port adapter (SPA) that indicates a successful installation. Figure 1-9 shows a view of the LED on the faceplate.

Table 1-9 SPA LED Indicating a Successful Installation

LED Label	Color—State	Description
STATUS	Solid green	SPA is powered on and is operational.

Figure 1-9 SPA Faceplate LED



Cisco ASR 1001 Built-in Gigabit Ethernet SPA LEDs

The Cisco ASR 1001 Router has a Built-in Gigabit Ethernet SPA, which is installed. Table 1-10 shows the Built-in SPA LEDs details.

Table 1-10 Cisco ASR 1001 Router Built-in Gigabit Ethernet SPA Successful Installation

LED Label	Color—State	Description
GE SFP STATUS	Amber or Green	Off indicates port is not enabled by software. Amber indicates the port is enabled by software, but Ethernet Link is not yet established. Green indicates the port is enabled by software and that an Ethernet Link has been established.

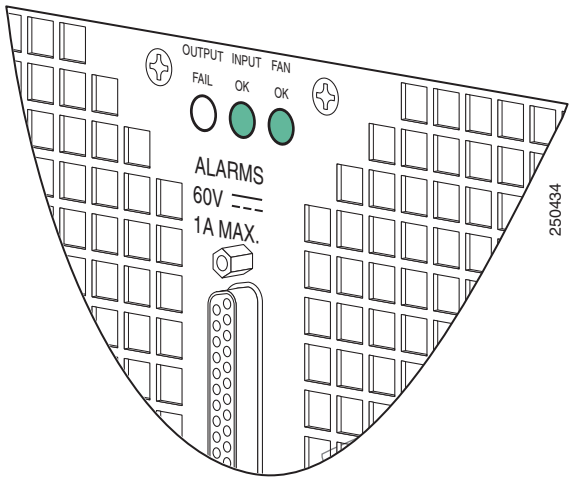
AC and DC Power Supplies

Table 1-11 shows the color or state of the LEDs that indicate a successful installation. Figure 1-10 shows a view of the LEDs on the faceplate.

Table 1-11 AC and DC Power Supply LEDs Indicating a Successful Installation

LED Label	Color—State	Description
INPUT OK	Green	Input voltage is within normal operating range.
FAN OK	Green	All fans are operational.
OUTPUT FAIL	Off	Output voltage is within normal operating range.

Figure 1-10 AC and DC Power Supply Faceplate LEDs



Checking Status Using show Commands

Use the **show platform** and **show environment all** commands to check the online and environmental status of each FRU after installation.

The **show platform** command displays the online status information for router FRUs. The State column in **show platform** command output should display “ok” for SIPs, SPAs, power supplies, and fans. For RPs (shown as R0, R1) and ESPs (shown as F0, F1), the State column should display “ok, active” or “ok, standby.”

Note There is only one LED for each Power Supply on Cisco ASR 1001 Router and it is green when powered-up.

```
Router# show platform
Chassis type: ASR1001

Slot      Type                State                 Insert time (ago)
--------- ------------------- --------------------- -----------------
0         ASR1001             ok                    23:28:16
 0/0      4XGE-BUILT-IN       ok                    23:27:23
 0/1      SPA-2XOC12-POS      ok                    23:27:21
 0/2      ASR1001-IDC-4XGE    ok                    23:27:23
R0        ASR1001             ok                    23:28:16
 R0/0                         ok, active            23:28:16
F0        ASR1001             ok, active            23:28:16
P0        Unknown             ps, fail              never
P1        ASR1001-PWR-AC      ok                    23:27:50
P2        ASR1001-FANTRAY     ok                    23:27:51

Slot      CPLD Version        Firmware Version
--------- ------------------- ------------------------------------
0         0902010A            12.2(20090526:143323) [gschnorr-mcp_...
R0        09020110            12.2(20090526:143323) [gschnorr-mcp_...
F0        0902010A            12.2(20090526:143323) [gschnorr-mcp_...

Router# show platform
Chassis type: ASR1013
Slot      Type                State                 Insert time (ago)
--------- ------------------- --------------------- -----------------
0         ASR1000-SIP10       ok                    1w0d
1         ASR1000-SIP40       ok                    1w0d
 1/1      SPA-5X1GE-V2        ok                    1w0d
2         ASR1000-SIP40       ok                    1w0d
 2/1      SPA-1X10GE-L-V2     ok                    1w0d
 2/3      SPA-1X10GE-L-V2     ok                    1w0d
3         ASR1000-SIP40       ok                    1w0d
 3/3      SPA-4XT3/E3         ok                    1w0d
4         ASR1000-SIP40       ok                    1w0d
 4/2      SPA-5X1GE-V2        ok                    1w0d
 4/3      SPA-4XCT3/DS0       ok                    1w0d
5         ASR1000-SIP40       ok                    1w0d
R0        ASR1000-RP2         ok, active            1w0d
R1        ASR1000-RP2         ok, standby           1w0d
F0        ASR1000-ESP40       ok, active            1w0d
F1        ASR1000-ESP40       ok, standby           1w0d
P0        ASR1013-PWR-AC      ok                    1w0d
P1        ASR1013-PWR-AC      ps, fail              1w0d
P2        ASR1013-PWR-AC      ok                    1w0d
P3        ASR1013-PWR-AC      ps, fail              1w0d
Slot      CPLD Version        Firmware Version
--------- ------------------- ------------------------------------
0         00200800            15.0(1r)S
1         00200800            15.0(1r)S
2         00200800            15.0(1r)S
3         00200800            15.0(1r)S
4         00200800            15.0(1r)S
5         00200800            15.0(1r)S
R0        10021901            15.0(1r)S
R1        10021901            15.0(1r)S
F0        1001270D            15.0(1r)S
F1        1001271D            15.0(1r)S

Router# show platform
Chassis type: ASR1006

Slot      Type                State                 Insert time (ago)
--------- ------------------- --------------------- -----------------
```

```
0          ASR1000-SIP10       ok                    18:23:58
 0/0       SPA-5X1GE-V2        ok                    18:22:38
 0/1       SPA-8X1FE-TX-V2     ok                    18:22:33
 0/2       SPA-2XCT3/DS0       ok                    18:22:38
1          ASR1000-SIP10       ok                    18:23:58
 1/0       SPA-2XOC3-POS       ok                    18:22:38
 1/1       SPA-8XCHT1/E1       ok                    18:22:38
 1/2       SPA-2XT3/E3         ok                    18:22:38
R0         ASR1000-RP1         ok, active            18:23:58
F0         ASR1000-ESP10       ok, active            18:23:58
P0         ASR1006-PWR-AC      ok                    18:23:09
P1         ASR1006-FAN         ok                    18:23:09

Slot       CPLD Version        Firmware Version
---------- ------------------- ----------------------------------------
0          06120701            12.2(33r)XN2
1          06120701            12.2(33r)XN2
R0         07082312            12.2(33r)XN2
F0         07051680            12.2(33r)XN2
```

The **show environment all** command displays system temperature, voltage, fan, and power supply conditions. (It does not display environmental information for SPAs.) The State column in **show environment all** output should show "Normal," except for fans where it indicates fan speed. A fan speed of 65% is normal.

```
Router# show environment all
Sensor List:  Environmental Monitoring
 Sensor            Location          State             Reading
 V1: VMA           F0                Normal            1801 mV
 V1: VMB           F0                Normal            1206 mV
 V1: VMC           F0                Normal            1206 mV
 V1: VMD           F0                Normal            1103 mV
 V1: VME           F0                Normal            1005 mV
 V1: 12v           F0                Normal            11967 mV
 V1: VDD           F0                Normal            3295 mV
 V1: GP1           F0                Normal            905 mV
 V2: VMA           F0                Normal            3295 mV
 V2: VMB           F0                Normal            2495 mV
 V2: VMC           F0                Normal            1499 mV
 V2: VMD           F0                Normal            1098 mV
 V2: VME           F0                Normal            1000 mV
 V2: VMF           F0                Normal            1000 mV
 V2: 12v           F0                Normal            11923 mV
 V2: VDD           F0                Normal            3295 mV
 V2: GP1           F0                Normal            751 mV
 Temp: Inlet       F0                Normal            27 Celsius
 Temp: Asic1       F0                Normal            44 Celsius
 Temp: Exhaust1    F0                Normal            36 Celsius
 Temp: Exhaust2    F0                Normal            34 Celsius
 Temp: Asic2       F0                Normal            40 Celsius
 V1: VMA           0                 Normal            1103 mV
 V1: VMB           0                 Normal            1201 mV
 V1: VMC           0                 Normal            1503 mV
 V1: VMD           0                 Normal            1801 mV
 V1: VME           0                 Normal            2495 mV
 V1: VMF           0                 Normal            3295 mV
 V1: 12v           0                 Normal            11967 mV
 V1: VDD           0                 Normal            3295 mV
 V1: GP1           0                 Normal            751 mV
 V1: GP2           0                 Normal            903 mV
 V2: VMB           0                 Normal            1201 mV
 V2: 12v           0                 Normal            11967 mV
```

```
 V2: VDD          0                Normal           3291 mV
 V2: GP2          0                Normal           903 mV
 Temp: Left       0                Normal           28 Celsius
 Temp: Center     0                Normal           29 Celsius
 Temp: Asic1      0                Normal           42 Celsius
 Temp: Right      0                Normal           27 Celsius
 V1: VMA          1                Normal           1103 mV
 V1: VMB          1                Normal           1201 mV
 V1: VMC          1                Normal           1503 mV
 V1: VMD          1                Normal           1801 mV
 V1: VME          1                Normal           2495 mV
 V1: VMF          1                Normal           3295 mV
 V1: 12v          1                Normal           11953 mV
 V1: VDD          1                Normal           3291 mV
 V1: GP1          1                Normal           754 mV
 V1: GP2          1                Normal           903 mV
 V2: VMB          1                Normal           1206 mV
 V2: 12v          1                Normal           11967 mV
 V2: VDD          1                Normal           3291 mV
 V2: GP2          1                Normal           905 mV
 Temp: Left       1                Normal           28 Celsius
 Temp: Center     1                Normal           30 Celsius
 Temp: Asic1      1                Normal           44 Celsius
 Temp: Right      1                Normal           28 Celsius
 PEM Iout         P0               Normal           37 A
 PEM Vout         P0               Normal           12 V AC
 PEM Vin          P0               Normal           116 V AC
 Temp: PEM        P0               Normal           28 Celsius
 Temp: FC         P0               Fan Speed 65%    25 Celsius
 Temp: FM         P1               Normal           1 Celsius
Temp: FC         P1               Fan Speed 65%    25 Celsius
 V1: VMA          R0               Normal           1118 mV
 V1: VMB          R0               Normal           3315 mV
 V1: VMC          R0               Normal           2519 mV
 V1: VMD          R0               Normal           1811 mV
 V1: VME          R0               Normal           1513 mV
 V1: VMF          R0               Normal           1220 mV
 V1: 12v          R0               Normal           12011 mV
 V1: VDD          R0               Normal           3300 mV
 V1: GP1          R0               Normal           913 mV
 V1: GP2          R0               Normal           1247 mV
 Temp: CPU        R0               Normal           29 Celsius
 Temp: Outlet     R0               Normal           30 Celsius
 Temp: Inlet      R0               Normal           25 Celsius
 Temp: Asic1      R0               Normal           30 Celsius
```

The **show environment all** command output shows an example of one power supply in the Cisco ASR 1001 Router:

```
Router# show environment all
Sensor List:  Environmental Monitoring
 Sensor           Location         State            Reading
 PEM Iout         P1               Normal           13 A
 PEM Vout         P1               Normal           12 V AC
 PEM Vin          P1               Normal           231 V AC
 Temp: Inlet      P1               Normal           27 Celsius
 Temp: Internal   P1               Normal           35 Celsius
 V1: VMA          R0               Normal           3295 mV
 V1: VMB          R0               Normal           1000 mV
 V1: VMC          R0               Normal           2495 mV
 V1: VMD          R0               Normal           2460 mV
 V1: VME          R0               Normal           1201 mV
 V1: VMF          R0               Normal           1796 mV
 V1: 12v          R0               Normal           11967 mV
```

```
 V1: VDD          R0                Normal            4970 mV
 V1: GP1          R0                Normal            1201 mV
 V1: GP2          R0                Normal            903 mV
 V2: VMA          R0                Normal            1098 mV
 V2: VMB          R0                Normal            1000 mV
 V2: VMC          R0                Normal            1499 mV
 V2: VMD          R0                5% high           1206 mV
 V2: VME          R0                Normal            1098 mV
 V2: VMF          R0                Normal            1054 mV
 V2: 12v          R0                Normal            11953 mV
 V2: VDD          R0                Normal            4985 mV
 V2: GP1          R0                5% high           812 mV
 V2: GP2          R0                20% low           2497 mV
 Temp: Middle     R0                Normal            54 Celsius
 Temp: CPU Die    R0                Normal            46 Celsius
 Temp: Top Left   R0                Normal            44 Celsius
 Temp: Asic1      R0                Normal            67 Celsius
 Temp: Inlet      R0                Normal            35 Celsius
 Temp: Asic3      R0                Normal            65 Celsius
 Temp: Rear       R0                Minor             60 Celsius
 Temp: Asic2      R0                Normal            60 Celsius
 Temp: Mid Frnt   R0                Normal            50 Celsius
 Temp: MCH Die    R0                Normal            70 Celsius
 Temp: FC         R0                Fan Speed 65%     35 Celsius
```

To display the Field Programmable Devices (FPD) on Cisco ASR 1001 Router, use the **show hw-module all fpd** command:

```
Router# show hw-module all fpd

==== ====================== ====== ============================================
                            H/W   Field Programmable   Current   Min. Required
Slot Card Type              Ver.  Device: "ID-Name"    Version      Version
==== ====================== ====== ================== =========== ==============
 0/0 4XGE-BUILT-IN          1.0   1-GE I/O FPGA          1.10         1.10
---- ---------------------- ------ ------------------ ----------- --------------
 0/1 SPA-2XOC12-POS         1.0   1-I/O FPGA             1.1          1.1
---- ---------------------- ------ ------------------ ----------- --------------
 0/2 ASR1001-IDC-4XGE       1.1   1-GE I/O FPGA          1.10         1.10
==== ====================== ====== ============================================
```

To display the Field Programmable Devices (FPD) on Cisco ASR 1013 Router, use the **show hw-module all fpd** command:

```
Router# show hw-module all fpd

==== ====================== ====== ============================================
                            H/W   Field Programmable   Current   Min. Required
Slot Card Type              Ver.  Device: "ID-Name"    Version      Version
==== ====================== ====== ================== =========== ==============
 4/2 SPA-2CHT3-CE-ATM       1.0   3-SPAMON               1.4          1.4
                                  6-IOFPGA               2.25         2.25
                                  9-UFE                  1.10         1.10
---- ---------------------- ------ ------------------ ----------- --------------
 5/0 SPA-5X1GE-V2           1.2   1-GE I/O FPGA          1.10         1.10
---- ---------------------- ------ ------------------ ----------- --------------
 5/1 SPA-8X1GE-V2           1.1   1-GE I/O FPGA          1.10         1.10
---- ---------------------- ------ ------------------ ----------- --------------
 5/2 SPA-4XT3/E3            1.1   1-ROMMON               2.12         2.12
                                  2-I/O FPGA             1.1          1.1
                                  3-E3 FPGA              1.4          1.4
                                  4-T3 FPGA              1.4          1.4
==== ====================== ====== ============================================
```

When Installation Is Not Successful

This section discusses the following items to check or troubleshoot when installation is not successful:

Physical Connections

Rule out an easily-fixed physical connection problem by verifying that:

- Power supplies are plugged in and switched on.
- Cables are connected.
- All FRUs are seated correctly.

Mechanical Damage

Examples of mechanical damage are a bent flange on a power supply or bent pins on a connector. If you detect mechanical damage:

- Do *not* attempt to straighten pins or repair mechanical damage.
- If you can see damaged pins, do *not* attempt to insert an assembly (SPA, SIP, ESP, or RP) into any slot. Doing so can damage the assembly or the chassis.
- Return the damaged equipment.

Alarm LED Is Illuminated

If the CRIT, MAJ, or MIN alarm LED is illuminated, determine the cause of the alarm by doing *one* of the following:

- Review the alarm message. The **logging alarm** command must be enabled for the system to send alarm messages to the console. The following is an example of an alarm message that was generated when a SPA was removed without a graceful deactivation of the SPA:

  *Aug 22 13:27:33.774: %ASR1000_OIR-6-REMSPA: SPA removed from subslot 1/1, interfaces disabled

  *Aug 22 13:27:33.775: %SPA_OIR-6-OFFLINECARD: SPA (SPA-4XT-SERIAL) offline in subslot 1/1

- Enter the **show facility-alarm status** command. The following example shows a critical alarm that is generated when a SPA is removed fr om the system:

```
Router# show facility-alarm status
System Totals  Critical: 1  Major: 0  Minor: 0

Source              Severity      Description [Index]
------              --------      -------------------
subslot 1/1         CRITICAL      Active Card Removed OIR Alarm [0]
```

Note A critical alarm "Active Card Removed OIR Alarm" is generated even if a SPA is removed after performing graceful deactivation.

Status LED Remains Amber

As Cisco IOS boots on a FRU, the status LED is amber or yellow. When Cisco IOS has successfully booted, the status LED becomes solid green.

If the status LED remains amber or yellow, check the console for alarm messages. The **logging alarm** command must be enabled for the system to send alarm messages to the console.

If there is no information on the console, some setting or error is not allowing Cisco IOS to boot. Contact Cisco Support; it is possible you might need to replace the FRU.

LEDS Are Not Illuminated on a Power Supply

DC Power Supply

If LEDs are not illuminated on the DC power supply, many times the problem is reversed polarity. Check the DC input power supply to see if the positive and negative lead wires are swapped.

AC Power Supply

If LEDs are not illuminated on the AC power supply, there is no input power or the power cord is not fully seated. If the power cord is fully seated, check the input power.

For More Information

For more information about the topics discussed in this chapter, see the following documents:

Topic	Document
Command descriptions	Cisco IOS Master Command List, All Releases Command Lookup Tool (Requires Cisco.com user ID and password)*OL-17665-04*
Graceful Deactivation of a SIP or SPA: Online insertion and removal (OIR)	“Installing and Removing a SIP” chapter in the *Cisco ASR 1000 Series Aggregation Services Routers SIP and SPA Hardware Installation Guide*

Topic	Document
LEDs for the RP, ESP, SIP, and AC and DC power supplies	"Cisco ASR 1000 Series Routers Components Overview" chapter in the *Cisco ASR 1000 Series Router Hardware Installation Guide*
LEDs for the SIP and SPA	*Cisco ASR 1000 Series Aggregation Services Routers SIP and SPA Hardware Installation Guide*
Cisco ASR 1001 Router Quick-Start	*Cisco ASR 1001 Router Quick Start Guide*
Overview, Installation, and Detailed information of Cisco ASR 1001 Router	*Cisco ASR 1000 Series Router Hardware Installation Guide*
Cisco ASR 1013 Router Quick-Start	*Cisco ASR 1013 Router Quick Start Guide*

CHAPTER 2

Monitoring Hardware Using Alarms

Once hardware is installed and operational, use alarms to monitor hardware status on a daily basis.

This chapter includes the following sections:

Router Design and Monitoring Hardware

The Cisco ASR 1000 Series Aggregation Services Routers are designed to send alarm notifications when problems are detected. Network administrators do not need to use **show** commands to poll devices on a routine basis and can monitor the network remotely. However, network administrators can perform onsite monitoring if they so choose.

Approaches for Monitoring Hardware Alarms

The following sections discuss ways in which you can monitor hardware using alarms:

Onsite Network Administrator Responds to Audible or Visual Alarms

An external element can be connected to a power supply using the DB-25 alarm connector on the power supply. The external element is a DC lightbulb for a visual alarm and a bell for an audible alarm.

If an alarm illuminates the CRIT, MIN, or MAJ LED on the Cisco ASR 1000 Series Route Processor (RP) faceplate, and a visual or audible alarm is wired, the alarm also activates an alarm relay in the power supply DB-25 connector (on the Cisco ASR 1006 Router and Cisco ASR 1004 Router). The bell rings or the lightbulb flashes.

Clearing Audible and Visual Alarms

To clear an audible alarm, do *one* of the following:

- Press the Audible Cut Off button on the RP faceplate.
- Enter the **clear facility-alarm** command.

To clear a visual alarm, you must resolve the alarm condition. The **clear facility-alarm** command does not clear an alarm LED on the RP faceplate or turn off the DC lightbulb. For example, if a critical alarm LED is illuminated because an active SPA was removed without a graceful deactivation of the SPA, the only way to resolve that alarm is to replace the SPA.

Network Administrator Checks Console or Syslog for Alarm Messages

The network administrator can monitor alarm messages by reviewing alarm messages sent to the system console or to a syslog. This section discusses the following topics:

Enabling the logging alarm Command

The **logging alarm** command must be enabled for the system to send alarm messages to a logging device, such as the console or a syslog. This command is not enabled by default.

You can specify the severity level of alarm to log. All alarms at and above the specified threshold generate alarm messages. For example, the following command sends only critical alarm messages to logging devices:

```
Router(config)# logging alarm critical
```

If alarm severity is not specified, alarm messages for all severity levels are sent to logging devices.

Examples of Alarm Messages

The following alarm messages are examples of alarm messages that are sent to the console when a SPA is removed without first doing a graceful deactivation of the SPA. The alarm is cleared when the SPA is re-inserted.

SPA REMOVED

*Aug 22 13:27:33.774: %ASR1000_OIR-6-REMSPA: SPA removed from subslot 1/1, interfaces disabled

*Aug 22 13:27:33.775: %SPA_OIR-6-OFFLINECARD: SPA (SPA-4XT-SERIAL) offline in subslot 1/1

SPA RE-INSERTED

*Aug 22 13:32:29.447: %ASR1000_OIR-6-INSSPA: SPA inserted in subslot 1/1

*Aug 22 13:32:34.916: %SPA_OIR-6-ONLINECARD: SPA (SPA-4XT-SERIAL) online in subslot 1/1

*Aug 22 13:32:35.523: %LINK-3-UPDOWN: SIP1/1: Interface EOBC1/1, changed state to up

ALARMS For Cisco ASR 1001 Router

To view the alarms on Cisco ASR 1001 router, use the **show facility-alarm status** command. The example shows a critical alarm for Power supply along with the description:

```
Router# show facility-alarm status
System Totals  Critical: 2  Major: 0  Minor: 1
Source              Severity      Description [Index]
------              --------      -------------------
Power Supply Bay 0         CRITICAL      Power Supply/FAN Module Missing [0]
xcvr container 0/0/0       CRITICAL      Transceiver Missing - Link Down [1]
xcvr container 0/1/0       INFO          Transceiver Missing [0]
xcvr container 0/1/1       INFO          Transceiver Missing [0]
xcvr container 0/2/0       INFO          Transceiver Missing [0]
xcvr container 0/2/1       INFO          Transceiver Missing [0]
xcvr container 0/2/2       INFO          Transceiver Missing [0]
xcvr container 0/2/3       INFO          Transceiver Missing [0]
Temp: Rear R0/26           MINOR         Temp Above Normal [4]
```

To view critical alarms specifically, use the **show facility-alarm status critical command:**

```
Router# show facility-alarm status critical
System Totals  Critical: 2  Major: 0  Minor: 1

Source               Severity      Description [Index]
------                     --------      -------------------
Power Supply Bay 0         CRITICAL      Power Supply/FAN Module Missing [0]
xcvr container 0/0/0       CRITICAL      Transceiver Missing - Link Down [1]
```

To view the operational state of the major hardware components on Cisco ASR 1001 Router, use the **show platform diag** command. This example shows the Power supply P0 has failed:

```
Router# show platform diag
Chassis type: ASR1001

Slot: 0, ASR1001
  Running state              : ok
  Internal state             : online
  Internal operational state  : ok
  Physical insert detect time : 00:00:51 (1d01h ago)
  Software declared up time   : 00:01:37 (1d01h ago)
  CPLD version               : 0902010A
  Firmware version           : 12.2(20090526:143323) [gschnorr-mcp_dev_1ru2 rel
ease 1.5 ]

Sub-slot: 0/0, 4XGE-BUILT-IN
  Operational status         : ok
  Internal state             : inserted
  Physical insert detect time : 00:01:39 (1d01h ago)
  Logical insert detect time  : 00:01:45 (1d01h ago)

Sub-slot: 0/1, SPA-2XOC12-POS
  Operational status         : ok
  Internal state             : inserted
  Physical insert detect time : 00:01:40 (1d01h ago)
  Logical insert detect time  : 00:01:47 (1d01h ago)

Sub-slot: 0/2, ASR1001-IDC-4XGE
  Operational status         : ok
  Internal state             : inserted
  Physical insert detect time : 00:01:41 (1d01h ago)
  Logical insert detect time  : 00:01:45 (1d01h ago)

Slot: R0, ASR1001
```

```
  Running state               : ok
  Internal state              : online
  Internal operational state  : ok
  Physical insert detect time : 00:00:51 (1d01h ago)
  Software declared up time   : 00:00:51 (1d01h ago)
  CPLD version                : 09020110
  Firmware version            : 12.2(20090526:143323) [gschnorr-mcp_dev_1ru2 rel
ease 1.5 ]

Sub-slot: R0/0,
  Running state               : ok, active
  Logical insert detect time  : 00:00:51 (1d01h ago)
  Became HA Active time       : 00:03:20 (1d01h ago)

Sub-slot: R0/1,
  Running state               : ok, standby
  Logical insert detect time  : 00:02:04 (1d01h ago)

Slot: F0, ASR1001
  Running state               : ok, active
  Internal state              : online
  Internal operational state  : ok
  Physical insert detect time : 00:00:51 (1d01h ago)
  Software declared up time   : 00:01:32 (1d01h ago)
  Hardware ready signal time  : 00:01:26 (1d01h ago)
  Packet ready signal time    : 00:01:37 (1d01h ago)
  CPLD version                : 0902010A
  Firmware version            : 12.2(20090526:143323) [gschnorr-mcp_dev_1ru2 rel
ease 1.5 ]

Slot: P0, Unknown
  State                       : ps, fail
  Physical insert detect time : 00:00:00 (never ago)

Slot: P1, ASR1001-PWR-AC
  State                       : ok
  Physical insert detect time : 00:01:18 (1d01h ago)

Slot: P2, ASR1001-FANTRAY
  State                       : ok
  Physical insert detect time : 00:01:17 (1d01h ago)
```

To view the operational state of the major hardware components on Cisco ASR 1013 Router, use the **show platform diag** command. This example shows the Power supply P0 has failed:

```
Router# show platform diag
Chassis type: ASR1013

Slot: 4, ASR1000-SIP10
  Running state               : ok
  Internal state              : online
  Internal operational state  : ok
  Physical insert detect time : 00:00:48 (02:20:23 ago)
  Software declared up time   : 00:01:42 (02:19:29 ago)
  CPLD version                : 09111601
  Firmware version            : 15.0(1r)S

Sub-slot: 4/2, SPA-2CHT3-CE-ATM
  Operational status          : ok
  Internal state              : inserted
  Physical insert detect time : 00:00:44 (02:20:27 ago)
  Logical insert detect time  : 00:02:23 (02:18:48 ago)

Slot: 5, ASR1000-SIP40
```

```
  Running state               : ok
  Internal state              : online
  Internal operational state  : ok
  Physical insert detect time : 00:00:48 (02:20:23 ago)
  Software declared up time   : 00:01:39 (02:19:32 ago)
  CPLD version                : 00200800
  Firmware version            : 15.0(1r)S

Sub-slot: 5/0, SPA-5X1GE-V2
  Operational status          : ok
  Internal state              : inserted
  Physical insert detect time : 00:00:43 (02:20:28 ago)
  Logical insert detect time  : 00:02:30 (02:18:41 ago)

Sub-slot: 5/1, SPA-8X1GE-V2
  Operational status          : ok
  Internal state              : inserted
  Physical insert detect time : 00:00:43 (02:20:28 ago)
  Logical insert detect time  : 00:02:24 (02:18:47 ago)

Sub-slot: 5/2, SPA-4XT3/E3
  Operational status          : ok
  Internal state              : inserted
  Physical insert detect time : 00:00:43 (02:20:28 ago)
  Logical insert detect time  : 00:02:30 (02:18:40 ago)

Slot: R0, ASR1000-RP2
  Running state               : ok, active
  Internal state              : online
  Internal operational state  : ok
  Physical insert detect time : 00:00:48 (02:20:23 ago)
  Software declared up time   : 00:00:48 (02:20:23 ago)
  Became HA Active time       : 00:05:05 (02:16:06 ago)
  CPLD version                : 10021901
  Firmware version            : 12.2(33r)XND

Slot: R1, ASR1000-RP2
  Running state               : ok, standby
  Internal state              : online
  Internal operational state  : ok
  Physical insert detect time : 00:00:48 (02:20:23 ago)
  Software declared up time   : 00:02:42 (02:18:29 ago)
  CPLD version                : 10021901
  Firmware version            : 12.2(33r)XND

Slot: F0, ASR1000-ESP40
  Running state               : ok, active
  Internal state              : online
  Internal operational state  : ok
  Physical insert detect time : 00:00:48 (02:20:23 ago)
  Software declared up time   : 00:05:30 (02:15:41 ago)
  Hardware ready signal time  : 00:04:22 (02:16:49 ago)
  Packet ready signal time    : 00:05:33 (02:15:37 ago)
  CPLD version                : 1003190E
  Firmware version            : 15.0(1r)S

Slot: F1, ASR1000-ESP40
  Running state               : init, standby
  Internal state              : online
  Internal operational state  : ok
  Physical insert detect time : 00:00:48 (02:20:23 ago)
  Software declared up time   : 01:35:45 (00:45:26 ago)
  Hardware ready signal time  : 01:34:35 (00:46:36 ago)
  Packet ready signal time    : 00:00:00 (never ago)
```

```
  CPLD version               : 1003190E
  Firmware version           : 15.0(1r)S

Slot: P0, Unknown
  State                      : ps,fail
  Physical insert detect time : 00:00:00 (never ago)

Slot: P1, ASR1013-PWR-AC
  State                      : ok
  Physical insert detect time : 00:01:35 (02:19:36 ago)

Slot: P2, ASR1013-PWR-AC
  State                      : ok
  Physical insert detect time : 00:01:35 (02:19:35 ago)

Slot: P3, ASR1013-PWR-AC
  State                      : ok
  Physical insert detect time : 00:01:36 (02:19:35 ago)
```

Reviewing and Analyzing Alarm Messages

To facilitate the review of alarm messages, you can write scripts to analyze alarm messages sent to the console or syslog. Scripts can provide reports on events such as alarms, security alerts, and interface status.

Syslog messages can also be accessed through Simple Network Management Protocol (SNMP) using the history table defined in the CISCO-SYSLOG-MIB.

Network Management System Alerts Network Administrator When an Alarm Is Reported Through SNMP

The Simple Network Management Protocol (SNMP) is an application-layer protocol that provides a standardized framework and a common language used for monitoring and managing devices in a network. Of all the approaches to monitor alarms, SNMP is the best approach for enterprise and service provider customers that have many routers to monitor.

SNMP provides notification of faults, alarms, and conditions that might affect services. SNMP allows a network administrator to access router information through a network management system (NMS) instead of by polling devices, reviewing logs, or reviewing log reports.

Note "Transceiver Missing - Link Down" alarm will be reported with a severity of "CRITICAL" in the output of **show facility-alarm status** command.

To use SNMP to get alarm notification, you must use the following MIBs:

- ENTITY-MIB, RFC 4133 (required for the CISCO-ENTITY-ALARM-MIB and CISCO-ENTITY-SENSOR-MIB to work)
- CISCO-ENTITY-ALARM-MIB
- CISCO-ENTITY-SENSOR-MIB (for SPA and transceiver environmental alarm information, which is not provided through the CISCO-ENTITY-ALARM-MIB)

For More Information

For more information about the topics discussed in this chapter, see the following documents:

Topic	Document
Command descriptions	Cisco IOS Master Command List, All Releases Command Lookup Tool (Requires Cisco.com user ID and password)
Configuring MIB support	*Cisco ASR 1000 Series Aggregation Services Routers MIB Specifications Guide*
Configuring SNMP	"SNMP Support" chapter in the *Cisco IOS XE Network Management Configuration Guide, Release 2*
Graceful Deactivation of a SIP or SPA: Online insertion and removal (OIR)	"Installing and Removing a SIP" chapter in the *Cisco ASR 1000 Series Aggregation Services Routers SIP and SPA Hardware Installation Guide*
MIBs supported on the Cisco ASR 1000 Series Aggregation Services Routers	*Cisco ASR 1000 Series Aggregation Services Routers MIB Specifications Guide*
Power supplies and the DB-25 alarm connector	"Cisco ASR 1000 Series Routers Components Overview" chapter in the *Cisco ASR 1000 Series Aggregations Services Routers Hardware Installation Guide*

CHAPTER 1

Configuring the Common Criteria Tcl Scripts

To monitor the packet drop event on the ASR 1000 Series Router, use the Common Criteria Tcl scripts.

This chapter includes the following sections:

Common Criteria Tcl Scripts Overview

Common Criteria (CC) is an international standard for evaluating IT product security and reliability. It is recognized by over 15 countries around the world including Australia, Canada, France, Germany, Greece, Italy, Japan, New Zealand, Spain, UK, South Korea and the United States. Many government customers around the world consider Common Criteria a mandatory requirement for purchasing network security products.

Common Criteria is a methodology for product evaluation. There are seven levels of evaluation and only levels 1 through 4 are mutually recognized by the participating countries. Products typically target EAL2 or EAL4, an evaluation conducted in any one of the participating countries is valid for the rest for the members. Cisco continues to be a global leader in completing and pursuing Common Criteria evaluations.

ASR1000 Series Routers support packet drop event monitoring as required by the Common Criteria standards. The Common Criteria features can be enabled using Tcl scripting. To find out more about Cisco IOS XE scripting using Tcl, see the “Cisco IOS XE Scripting with Tcl” chapter of the Cisco IOS XE Network Management Software Configuration Guide.

Common Criteria leverages the IOS XE Embedded Syslog Manager (ESM) and Embedded Event Manager (EEM) mechanisms for enabling periodic actions. The ESM feature provides a programmable framework that allows you to filter, escalate, correlate, route, and customize system logging messages prior to delivery by the Cisco IOS system message logger. For more information, see the Embedded Syslog Manager (ESM) Configuration Guide.

Table 1-1 lists the Common Criteria Tcl scripts.

Table 1-1 Common Criteria Tcl Scripts

Script Name	Description
timer.tcl	Supports the Timer events for other scripts.
alarms_db.tcl	Manages the alarms database.
em_ike_phase1_failure.tcl	Monitors the Internet Key Exchange (IKE) protocol Phase 1 negotiation failures.
em_ike_phase2_failure.tcl	IKEv1 Phase 2 negotiation failures watcher script.
em_login_failure.tcl	Monitors the user login failures.
em_monitor_violation.tcl	Monitors the information flow violations. ACL-based event monitors must be configured to trigger the violation monitor watcher.
em_monitor_vpn_event.tcl	Monitors the VPN encryption, decryption faults, and packet replay events.
monitor_ipsec.tcl	Configures the VPN event monitors.
syslog_exclude.tcl	Excludes the syslog messages containing the keywords from the syslog database.
syslog_include.tcl	Includes the syslog messages containing the keywords in the syslog database.
esm_conf_vty.tcl	Configures the syslog message output to the connected vty devices.

Installing the Common Criteria Tcl Scripts

Super administrator can copy the scripts from a portable device such as a USB flash drive on the hard disk, which is defined as a protected directory.

Example:

```
Copy bootflash:<folder name> <Tcl file name> harddisk:/cc_scripts
```

How to Configure the Common Criteria Tcl Scripts

To configure the Common Criteria Tcl scripts, complete the following steps:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **logging filter** [*script-url*] [**args** *filter-arguments*]
4. **end**
5. **show logging**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables the privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters the global configuration mode.
Step 3	**logging filter** [*script-url*] [**args** *filter-arguments*] **Example:** `Router(config)# logging filter bootflash:/escalate.tcl 1 args CONFIG_I 1`	Specifies one or more syslog filter modules to be applied to the generated system logging messages. To remove a module from the list of modules to be executed, use the **no** form of this command. **Note** Repeat this command for each syslog filter module that should be used. • (Optional) The *script-url* argument specifies the script URL. **Note** Provide a valid directory location, an incorrect location can trigger a router reload. • (Optional) The **args** *filter-arguments* syntax can be added to pass arguments to the specified filter. Multiple arguments can be specified. The number and type of arguments should be defined in the syslog filter module. For example, if the syslog filter module is designed to accept a specific e-mail address as an argument, you could pass the e-mail address using the **args user@host.com** syntax. Multiple arguments are typically delimited by spaces.
Step 4	**end** **Example:** `Router(config)# end`	Ends your current configuration session and returns the CLI to privileged EXEC mode.
Step 5	**show logging** **Example:** `Router# show logging`	(Optional) Displays the status of system logging, including the status of ESM-filtered logging: • If filtered logging to the buffer is enabled, this command also displays the data stored in the buffer. • The order in which the syslog filter modules are listed in the output of this command is the order in which the filter modules are executed.

Examples

This section provides the following configuration examples:

Alarm Confirmation Timer

This Common Criteria alarm confirmation timer watches for repetitive CC alarm confirmation requests. These requests are managed by the **timer.tcl** script:

```
logging filter <script-url>timer.tcl [args <interval>]
```

`interval`—interval between two successive CC alarm prompts. A default interval between two successive CC alarm prompts is 60 seconds:

```
logging filter bootflash:timer.tcl args 120
```

Alarm Database Manager

This Common Criteria alarm database manager maintains a repository of unconfirmed CC alarms. This request is managed by the **alarms_db.tcl** script:

```
logging filter <script-url>alarms_db.tcl [args <audible-property>]
```

`audible-property`—`alarm_audible` or `alarm_not_audible`

When the `alarm-property` is set to `alarm_audible`, it enables audio signals for every CC alarm confirmation prompt. By default, `audible-property` is set to `alarm_not_audible`:

```
logging filter bootflash:alarms_db.tcl args alarm_audible
```

IKEv1 Phase 1 and Phase 2 Failures Catcher

The IKEv1 failure catcher alert enables the monitoring of IKEv1 phase 1 and phase 2 negotiation failures. The commands for configuring the IKEv1 negotiation failure monitor are:

```
logging filter <script-url>em_ike_phase1_failure.tcl [args threshold [interval]]
logging filter <script-url>em_ike_phase2_failure.tcl [args threshold [interval]]
```

The argument values are as follows:

- `Threshold`—number of failures after which the CC alarm is raised. The default threshold value is 1.

- `Interval`—time interval during which the number of failures must reach a set threshold. On reaching the threshold, the alarms are triggered. The default value is indefinite.

  If the interval value is not set, the CC alarm is raised after the threshold pertaining to the number of failures is crossed.

  If the interval value is set, and the value is less than the threshold value, the failure counter is reset and the CC alarm is not raised.

Example:

```
logging filter bootflash:em_ike_phase1_failure.tcl args  3 300
```

This configuration raises a CC alarm after three IKEv1 Phase 1 failures occur during the 300-second interval.

If the number of failures are less than three within the 300-second interval, the CC alarm is not raised, and the failure counter is reset.

Syslog Filter

Syslog filter commands support both inclusive and exclusive filtering of syslog messages. The configured filters determine the order of syslog command execution. The number of syslog filters that can be configured depends on the device memory size.

The commands for configuring the syslog filters are:

- Inclusive filtering:

  ```
  logging filter <script-url>syslog_include.tcl [args <string>]
  ```

  The value of the `string` argument is an arbitrary character string.

  Example:

  ```
  logging filter bootflash:syslog_include.tcl args  ALARM
  logging filter bootflash:syslog_include.tcl args  LINK
  ```

  Syslog messages containing character strings such as ALARM or LINK are propagated to the configured auditable events repositories. Syslog messages that do not contain the configured character strings are dropped.

- Exclusive filtering

  ```
  logging filter <script-url>syslog_exclude.tcl [args <string>]
  ```

  The value of the string argument is an arbitrary character string.

  Example:

  ```
  logging filter bootflash:syslog_exclude.tcl args ALARM
  ```

  Any syslog message that contains the configured character string is dropped. Syslog messages that do not contain the configured character string are propagated to the auditable events repository.

Note Strings containing special characters should be enclosed within a pair of the escape characters such as single quotes (' '), double quotes (" "), or backslash (\ \).

Information Flow Violations Watcher

When an information flow violation occurs, the information flow violations watcher triggers a CC alarm. The command to configure the information flow violations watcher is:

```
logging filter <script-url>em_monitor_violation.tcl
```

IPsec Policy Violation Category Watcher

When an IPsec policy violation occurs, the IPsec policy violations watcher triggers a CC alarm. The command to configure IPsec policy violations watcher is

```
logging filter <script-url>monitor_ipsec.tcl args <esp> <category> <threshold>
```

The argument values are as follows:

- `esp`—Active or standby ASR1000 ESP on which IPsec policy violations are monitored.
- `category`—`decrypt-failed`, or `encrypt-failed`, or `replay`.

  Watches for decryption or encryption failures or IPsec packets replay events
- `threshold`—Number of events watched after which a cumulative event is reported. The threshold value must be greater than 0.

Note All command arguments are mandatory.

Multiple command lines can be configured for watching multiple categories of the IPsec policy violations.

Example:

```
logging filter bootflash:monitor_ipsec.tcl args active replay 100000
```

This command line configures a watcher for the IPsec packet replay violations. The watcher triggers an alarm after 100000 replayed IPsec packets are detected.

VPN Policy Violations Catcher

The VPN policy violations catcher triggers an alert if a violation occurs on the previously configured VPN policy:

```
logging filter <script-url> em_monotor_vpn_event.tcl
```

Replication Output of Syslog Messages

To replicate the syslog messages to all the connected terminal devices, use the following command:

```
logging filter <script-url>esm_conf_vty.tcl
```

Generating the Event Alarm Reports

CC Protection Profiles identify a number of events that generate alarms. The alarms must be acknowledged by the administrator.

For example, the following commands display the acknowledgement of alarms on the router:

```
000077:  *Apr 21 03:02:19.566: %CC-6-INFO: Please confirm alarm 000077
000077:  *Apr 21 02:54:23.001: %CC-6-ALARM: Login Authentication Failed for user eve 2
times in 11 seconds interval
```

Alarm confirmation on the router:

```
Router (config)#event manager environment confirm_alarm 000077
```

Based on the administrator-specified values, the syslog messages indicate alarm-inducing events.The reports that are generated for the event alarms include:

- Specified number of authentication failures—IOS supports logging of authentication events. To report authentication failures, administrators use the following commands:
  - `conf t`
  - `login on-failure log`
  - `end`
- Specified number of information flow policy violations by:
  - Individual source network identifiers, such as IP address, within a specified time.
  - Individual destination network identifiers, within user-specified time.
  - Individual destination subject service identifiers, such as TCP port, within user-specified time.
  - Individual or group rules within user-specified time.

Note The **monitor drop** command is used to configure event monitoring of the information flow policy violations.

- The VPN policy violation catcher includes:
  - Any detected replay of TSF data or security attributes
  - Security administrator-specified number of encryption failures
  - Security administrator-specified number of decryption failures

Note The **set platform hardware qfp feature ipsec event-monitor** command is used to configure VPN-specific event monitoring.
The **clear platform hardware qfp <mastership> feature ipsec event-monitor** command is used for removing the event monitors.

Configuration Examples of the Common Criteria Tcl Scripts

This section provides the following Tcl script examples:

- Example: Tcl Scripts for Common Criteria Alarms, page 1-8
- Example: Tcl Scripts for the IKEv1 Phase 1 Failure Catcher, page 1-12

Example: Tcl Scripts for Common Criteria Alarms

```
namespace eval ::common_criteria_alarms {
    # namespace variables
    array set unconfirmed_alarms_db {}
    array set logged_in_users_info {}
    array set alarms_linked_list {}
    variable first_alarm_id
    variable last_alarm_id

    array set msgs_to_watch {
        CC-6-ALARM    1
        CC-6-TIMER    1
        PARSER-5-CFGLOG_LOGGEDCMD  1
        SEC_LOGIN-5-LOGIN_SUCCESS 1
    }

    array set login_success_msg {
        SEC_LOGIN-5-LOGIN_SUCCESS 1
    }

    array set msg_to_log {
        CC-6-ALARM    1
    }

    array set msg_to_confirm {
        PARSER-5-CFGLOG_LOGGEDCMD  1
    }

    array set msg_timer {
        CC-6-TIMER    1
    }

    # Should I process this message ?
    proc query_category {cat} {
        variable msgs_to_watch

        if { [info exists msgs_to_watch($cat)] } {
            return $msgs_to_watch($cat)
        } else {
            return 0
        }
    }
```

```
# Should I log this message ?
proc query_log {cat} {
    variable msg_to_log

    if { [info exists msg_to_log($cat)] } {
        return $msg_to_log($cat)
    } else {
        return 0
    }
}

# Should I log this message ?
proc query_is_login_success {cat} {
    variable login_success_msg

    if { [info exists login_success_msg($cat)] } {
        return $login_success_msg($cat)
    } else {
        return 0
    }
}

# Should I confirm this message ?
proc query_confirm {cat} {
    variable msg_to_confirm

    if { [info exists msg_to_confirm($cat)] } {
        return $msg_to_confirm($cat)
    } else {
        return 0
    }
}

# is this timer syslog?
proc query_timer {cat} {
    variable msg_timer

    if { [info exists msg_timer($cat)] } {
        return $msg_timer($cat)
    } else {
        return 0
    }
}

# Accept alarm string and generate a syslog
proc generate_syslog {alarm_msg} {

# store all current syslog global params
    set prev_orig_msg       $::orig_msg
    set prev_timestamp      $::timestamp
    set prev_facility       $::facility
    set prev_mnemonic       $::mnemonic
    set prev_severity       $::severity
    set prev_stream         $::stream
    set prev_traceback      $::traceback
    set prev_pid            $::pid
    set prev_process        $::process
    set prev_format_string  $::format_string
    set prev_msg_args       $::msg_args

    # construct a new syslog with the details of the login failure
    # alarm

    set ::timestamp  [cisco_service_timestamp]
```

```
        set ::facility "CC"
        set ::mnemonic "INFO"
        set ::severity 6
        set ::stream 2
        set ::traceback "cc_internal_syslog"
        set ::pid ""
        set ::process ""
        set ::format_string ""
        set ::msg_args {}
        set ::orig_msg [format "%s %s: %s%s-%d-%s: %s" $::buginfseq $::timestamp "%"
$::facility $::severity $::mnemonic $alarm_msg]

        # Send a syslog to be catched by the script that handles the
        # alarms
        esm_errmsg 0

        # restore all syslog global params

        set ::orig_msg        $prev_orig_msg
        set ::timestamp       $prev_timestamp
        set ::facility        $prev_facility
        set ::mnemonic        $prev_mnemonic
        set ::severity        $prev_severity
        set ::stream          $prev_stream
        set ::traceback       $prev_traceback
        set ::pid             $prev_pid
        set ::process         $prev_process
        set ::format_string   $prev_format_string
        set ::msg_args        $prev_msg_args
    }


    # Process all alarm related syslogs (new alarm/timer/confirm)

    proc process_syslog {} {
        variable unconfirmed_alarms_db
        variable alarms_linked_list
        variable logged_in_users_info
        variable first_alarm_id
        variable last_alarm_id
        variable alarm_to_confirm

        # empty msg?
        if { [string length $::orig_msg] == 0} {
            return ""
        }

        set category "$::facility-$::severity-$::mnemonic"

        # Should I process this syslog?
        set need_to_process [query_category $category]
        if { $need_to_process == 0 } {
            return $::orig_msg
        }

        # Is this a login success syslog?
        set is_login_success [query_is_login_success $category]
        if { $is_login_success == 1} {
            # Save the ip address aln local port of the logged in user
            set user [lindex $::msg_args 0]
            set ip_address [lindex $::msg_args 1]
            set local_port [lindex $::msg_args 2]
            if { $ip_address == "0.0.0.0"} {
                set ip_address "console"
```

```
                set local_port "console"
            }
            set logged_in_users_info($user) [list $ip_address $local_port]
            return $::orig_msg
        }

        # Should I log this msg in unconfirmed alarms database?
        set need_to_log [query_log $category]
        if { $need_to_log == 1 } {
            set alarm_id [lindex [split $::buginfseq :]  0]
            if { $alarm_id == "" } {
                set alarm_db_syslog "Alarm Id is missing. Please configure 'service
sequence-numbers'"
                generate_syslog $alarm_db_syslog
                return ""
            }
            # Update the alarms linked list
            if { [info exists last_alarm_id] } {
                set alarms_linked_list($last_alarm_id) $alarm_id
                set last_alarm_id $alarm_id
                set alarms_linked_list($last_alarm_id) "void"
            } else {
                set first_alarm_id $alarm_id
                set last_alarm_id $alarm_id
                set alarms_linked_list($last_alarm_id) "void"
            }

            set unconfirmed_alarms_db($alarm_id) "$::orig_msg"
            return $::orig_msg
        }

        # Is this a confirmation syslog?
        set need_to_confirm [query_confirm $category]
        if { $need_to_confirm == 1 } {
            if { "event manager environment confirm_alarm" == [lrange [split [lindex
$::msg_args 1]] 0 3]} {
                set alarm_id_to_confirm [lindex [split [lindex $::msg_args end]] end]
                if { [info exists alarms_linked_list($alarm_id_to_confirm)] == 0} {
                    set alarm_db_syslog "ERROR: alarm id $alarm_id_to_confirm does not
exist"
                    generate_syslog $alarm_db_syslog
                    return $::orig_msg
                }

                if { $alarm_id_to_confirm != $first_alarm_id } {
                   set alarm_db_syslog "ERROR: Only the displayed alarm ($first_alarm_id)
can be confirmed"
                    generate_syslog $alarm_db_syslog
                    return $::orig_msg
                }

                set alarm_to_confirm unconfirmed_alarms_db($alarm_id_to_confirm)

                if { [string length $alarm_to_confirm] != 0} {
                    unset unconfirmed_alarms_db($alarm_id_to_confirm)

                    # Get the next alarm id to confirm
                    set first_alarm_id $alarms_linked_list($first_alarm_id)
                    unset alarms_linked_list($alarm_id_to_confirm)

                    if { $first_alarm_id == "void" } {
                        unset first_alarm_id
                        unset last_alarm_id
                    }
```

```
                    # Add user location info (ip adderss and local port
                    # to the orig msg
                    set user [lindex $::msg_args 0]
                    if { [info exists logged_in_users_info($user)] == 0 } {
                        set location_info {"unknown" "unknown"}
                    } else {
                        set location_info  $logged_in_users_info($user)
                    }
                    set ip_address [lindex $location_info 0]
                    set local_port [lindex $location_info 1]
                    set new_orig_msg "$::orig_msg \[source: $ip_address\] \[local_port:
$local_port\]"
                    set ::orig_msg $new_orig_msg
                    # Are there any unconfirmed alarms?
                    if { [info exists first_alarm_id] } {
                        set first_alarm_msg $unconfirmed_alarms_db($first_alarm_id)
                        set audible_sound ""
                        if { [lindex $::cli_args 0] == "alarm_audible" } {
                            set audible_sound "\a\t\a\t\a\t\a"
                        }
                        set alarm_db_syslog "Please confirm alarm $first_alarm_id \n
$first_alarm_msg $audible_sound"
                        generate_syslog $alarm_db_syslog
                    }
                    return $::orig_msg
                }
            } else {
                return $::orig_msg
            }
        }

        # Is this a cron/timer msgs
        set is_timer_msg [query_timer $category]
        if { $is_timer_msg == 1 } {
           # Are there any unconfirmed alarms?
            if { [info exists first_alarm_id] } {
                set first_alarm_msg $unconfirmed_alarms_db($first_alarm_id)
                set audible_sound ""
                if { [lindex $::cli_args 0] == "alarm_audible" } {
                    set audible_sound "\a\t\a\t\a\t\a"
                }
                set alarm_db_syslog "Please confirm alarm $first_alarm_id \n
$first_alarm_msg $audible_sound"
                generate_syslog $alarm_db_syslog
            }
        }

        return ""
    }

    # Process the message
    process_syslog

} ;# end namespace common_criteria_alarms
```

Example: Tcl Scripts for the IKEv1 Phase 1 Failure Catcher

```
namespace eval ::ike_auth_failures {
    # namespace variables
```

```
array set host_failed_ike_auth {}

# Should I process this message ?
proc query_category {cat} {
    variable msg_to_watch

    if { [info exists msg_to_watch($cat)] } {
        return 1
    } else {
        return 0
    }
}

# handle ike authentication failure for a given ip address
proc process_ike_auth_failure {ipaddress} {
    variable  host_failed_ike_auth

    set current_time    [clock seconds]

    #default values, change if get as cli args
    set alarm_threshold  1
    set time_gap         0

    # Get the list timestamps of previous ike authentication failures
    # of this ipaddress
    set time_list $host_failed_ike_auth($ipaddress)
    set list_len [llength $time_list]

    # First arg (if exists) is alarm threshlod
    if { [info exists ::cli_args] == 1 } {
        set alarm_threshold [lindex $::cli_args 0]

        # Second arg (if exists) is time gap
        if { [llength $::cli_args] > 1 } {
            set time_gap [lindex $::cli_args 1]
        }
    }

    if { $time_gap != 0 } {
        set i 0
        # run though the list and keep only the timestamps which are still
        # in the given time gap from current time
        while  { $i < $list_len } {
            if {[expr $current_time - [lindex $time_list $i]] <= $time_gap } {
                lappend new_time_list [lindex $time_list $i]
            }
            incr i
        }

        # update the timestamp list for the given ipaddress
        set host_failed_ike_auth($ipaddress) $new_time_list
    } else {
        set new_time_list $host_failed_ike_auth($ipaddress)
    }

    # does the updated timestamp list has more items than the threshold
    if { [llength $new_time_list] >= $alarm_threshold } {
        if { $time_gap != 0 } {
            # Need to send a new alarm.
            set ike_auth_fail_msg [format "Ike authentication failed with %s %d times
in %d seconds interval" $ipaddress [llength $new_time_list] [expr $current_time - [lindex
$new_time_list 0]]]
        } else {
```

```
                set ike_auth_fail_msg [format "Ike authentication failed with %s %d times"
$ipaddress [llength $new_time_list]]
            }
            # store all current syslog global params
            set prev_orig_msg       $::orig_msg
            set prev_timestamp      $::timestamp
            set prev_facility       $::facility
            set prev_mnemonic       $::mnemonic
            set prev_severity       $::severity
            set prev_stream         $::stream
            set prev_traceback      $::traceback
            set prev_pid            $::pid
            set prev_process        $::process
            set prev_format_string  $::format_string
            set prev_msg_args       $::msg_args

            # construct a new syslog with the details of the login failure
            # alarm

            set ::timestamp  [cisco_service_timestamp]
            set ::facility "CC"
            set ::mnemonic "ALARM"
            set ::severity 6
            set ::stream 2
            set ::traceback "cc_internal_syslog"
            set ::pid ""
            set ::process ""
            set ::format_string ""
            set ::msg_args {}
            set ::orig_msg [format "%s %s: %s%s-%d-%s: %s" $::buginfseq $::timestamp "%"
$::facility $::severity $::mnemonic $ike_auth_fail_msg]

            # Send a syslog to be catched by the script that handles the
            # alarms
            esm_errmsg 0

            # restore all syslog global params

            set ::orig_msg        $prev_orig_msg
            set ::timestamp       $prev_timestamp
            set ::facility        $prev_facility
            set ::mnemonic        $prev_mnemonic
            set ::severity        $prev_severity
            set ::stream          $prev_stream
            set ::traceback       $prev_traceback
            set ::pid             $prev_pid
            set ::process         $prev_process
            set ::format_string   $prev_format_string
            set ::msg_args        $prev_msg_args
        }
    }

    proc process_ike_syslog {} {
        variable host_failed_ike_auth
        variable msg_to_watch

        # empty msg?
        if { [string length $::orig_msg] == 0} {
            return ""
        }
        set category "$::facility-$::severity-$::mnemonic"

        # Should I process this syslog?
        set need_to_process [query_category $category]
```

```
        if { $need_to_process == 0 } {
            return $::orig_msg
        }

        # Extract isakmp mode of the failed negotiation
        if { $::mnemonic == "IKMP_MODE_FAILURE" } {
            set isakmp_mode [lindex $::msg_args 0]
            if { $isakmp_mode != "Main" && $isakmp_mode != "Aggressive" } {
                return $::orig_msg
            }
        }

        # Extract ip address of the failed authentication
        set ip_address [lindex $::msg_args $msg_to_watch($category)]
        if { [string length $ip_address] == 0 } {
            return $::orig_msg
        }

        # Get current time and add it to the given ip_address list
        set time [clock seconds]
        lappend host_failed_ike_auth($ip_address) $time

        process_ike_auth_failure $ip_address

        return $::orig_msg
    }

    array set msg_to_watch {
        CRYPTO-6-IKMP_AUTH_FAIL                   1
        CRYPTO-6-IKMP_MODE_FAILURE                1
        CRYPTO-4-IKE_DENY_SA_REQ                  1
        CRYPTO-6-IKMP_BAD_DOI_SA                  1
        CRYPTO-6-IKMP_CRYPT_FAILURE               0
        CRYPTO-6-IKMP_BAD_CERT_USE                0
        CRYPTO-5-IKMP_INVAL_CERT                  0
        CRYPTO-4-IKMP_NO_SA                       0
        CRYPTO-6-IKMP_NO_ID_CERT_DN_MATCH         0
        CRYPTO-6-IKMP_NO_ID_CERT_ADDR_MATCH       0
        CRYPTO-6-IKMP_NO_ID_CERT_FQDN_MATCH       0
        CRYPTO-6-IKMP_NO_ID_CERT_USER_FQDN_MATCH  0
        CRYPTO-6-IKMP_NOT_ENCRYPTED               0
        CRYPTO-6-IKMP_SA_NOT_AUTH                 0
        CRYPTO-4-IKMP_HASH_SIZE_EXCEEDED          0
    }

    # Process the message
    process_ike_syslog

} ;# end namespace ike_auth_failures
```

Example: Tcl Scripts for the IKEv1 Phase 2 Failure Catcher

```
namespace eval ::ike_quickmode_failures {
    # namespace variables
    array set failed_quickmode_peers_id {}

    # Should I process this message ?
    proc query_category {cat} {
        variable msg_to_watch
```

```
        if { [info exists msg_to_watch($cat)] } {
            return 1
        } else {
            return 0
        }
    }

    # handle ike quickmode failure for a given ip address
    proc process_ike_quickmode_failure {ipaddress} {
        variable  failed_quickmode_peers_id

        set current_time    [clock seconds]

        #default values, change if get as cli args
        set alarm_threshold  1
        set time_gap         0

        # Get the list timestamps of previous ike authentication failures
        # of this ipaddress
        set time_list $failed_quickmode_peers_id($ipaddress)
        set list_len [llength $time_list]

        # First arg (if exists) is alarm threshlod
        if { [info exists ::cli_args] == 1 } {
            set alarm_threshold [lindex $::cli_args 0]

            # Second arg (if exists) is time gap
            if { [llength $::cli_args] > 1 } {
                set time_gap [lindex $::cli_args 1]
            }
        }

        if { $time_gap != 0 } {
            set i 0
            # run though the list and keep only the timestamps which are still
            # in the given time gap from current time
            while  { $i < $list_len } {
                if {[expr $current_time - [lindex $time_list $i]] <= $time_gap } {
                    lappend new_time_list [lindex $time_list $i]
                }
                incr i
            }

            # update the timestamp list for the given ipaddress
            set failed_quickmode_peers_id($ipaddress) $new_time_list
        } else {
            set new_time_list $failed_quickmode_peers_id($ipaddress)
        }

        # does the updated timestamp list has more items than the threshold
        if { [llength $new_time_list] >= $alarm_threshold } {
            if { $time_gap != 0 } {
                # Need to send a new alarm.
                set ike_auth_fail_msg [format "Processing of quick mode failed with %s %d
times in %d seconds interval" $ipaddress [llength $new_time_list] [expr $current_time -
[lindex $new_time_list 0]]]
            } else {
                set ike_auth_fail_msg [format "Processing of quick mode failed with %s %d
times" $ipaddress [llength $new_time_list]]
            }
            # store all current syslog global params
            set prev_orig_msg       $::orig_msg
            set prev_timestamp      $::timestamp
            set prev_facility       $::facility
```

```
                set prev_mnemonic       $::mnemonic
                set prev_severity       $::severity
                set prev_stream         $::stream
                set prev_traceback      $::traceback
                set prev_pid            $::pid
                set prev_process        $::process
                set prev_format_string  $::format_string
                set prev_msg_args       $::msg_args

                # construct a new syslog with the details of the login failure
                # alarm

                set ::timestamp  [cisco_service_timestamp]
                set ::facility "CC"
                set ::mnemonic "ALARM"
                set ::severity 6
                set ::stream 2
                set ::traceback "cc_internal_syslog"
                set ::pid ""
                set ::process ""
                set ::format_string ""
                set ::msg_args {}
                set ::orig_msg [format "%s %s: %s%s-%d-%s: %s" $::buginfseq $::timestamp "%"
$::facility $::severity $::mnemonic $ike_auth_fail_msg]

                # Send a syslog to be catched by the script that handles the
                # alarms
                esm_errmsg 0

                # restore all syslog global params

                set ::orig_msg        $prev_orig_msg
                set ::timestamp       $prev_timestamp
                set ::facility        $prev_facility
                set ::mnemonic        $prev_mnemonic
                set ::severity        $prev_severity
                set ::stream          $prev_stream
                set ::traceback       $prev_traceback
                set ::pid             $prev_pid
                set ::process         $prev_process
                set ::format_string   $prev_format_string
                set ::msg_args        $prev_msg_args
            }
        }

        proc process_ike_quickmode_syslog {} {
            variable failed_quickmode_peers_id
            variable msg_to_watch

            # empty msg?
            if { [string length $::orig_msg] == 0} {
                return ""
            }
            set category "$::facility-$::severity-$::mnemonic"

            # Should I process this syslog?
            set need_to_process [query_category $category]
            if { $need_to_process == 0 } {
                return $::orig_msg
            }

            # Extract isakmp mode of the failed negotiation
            set isakmp_mode [lindex $::msg_args 0]
            if {$::mnemonic == "IKMP_MODE_FAILURE" && $isakmp_mode != "Quick"} {
```

```
            return $::orig_msg
        }

        # Extract ip address of the peer who failed quickmode
        set ip_address [lindex $::msg_args $msg_to_watch($category)]
        if { [string length $ip_address] == 0} {
            return $::orig_msg
        }

        # Get current time and add it to the given ip_address list
        set time [clock seconds]
        lappend failed_quickmode_peers_id($ip_address) $time

        process_ike_quickmode_failure $ip_address

        return $::orig_msg
    }

    array set msg_to_watch {
        CRYPTO-6-IKMP_MODE_FAILURE   1
        CRYPTO-6-IKMP_SA_NOT_OFFERED 0
        CRYPTO-6-IPSEC_TRANSFORM_NOT_SUPPORTED 0
    }

    # Process the message
    process_ike_quickmode_syslog

} ;# end namespace ike_quickmode_failures
```

Example: Tcl Scripts for User Login Failures

```
namespace eval ::login_failure {
    # namespace variables
    array set user_failed_logins {}
    array set init_variables {}
    array set global_variables {}

    # Should I process this message ?
    proc query_category {cat} {
        variable msg_to_watch

        if { [info exists msg_to_watch($cat)] } {
            return $msg_to_watch($cat)
        } else {
            return 0
        }
    }

    proc close_all_vty {} {

        #get all the vty IDs
        set vty_list [exec show ru | inc line vty]

               # split the contents on newlines
        set list_of_lines [split $vty_list "\n"]

        set first_vty -1
        set last_vty -1

        # loop through the lines
        foreach line $list_of_lines {
```

```
            set vty_location [lsearch -exact $line vty]
            if {$vty_location != -1} {
                set vty_id [lindex $line [expr $vty_location + 1]]
                if {$first_vty == -1} {
                    set first_vty $vty_id
                }
                set last_vty $vty_id
            }
        }

                if {$first_vty == $last_vty} {
                        ios_config "line vty $first_vty" "transport input none"
                } else {
                        ios_config "line vty $first_vty $last_vty" "transport input none"
                }

        # go over all the ssh connections and close them - one after the other -
        set ssh_list [exec show ssh | in IN]

        # split the contents on newlines
        set list_of_lines [split $ssh_list "\n"]

        # loop through the lines
        foreach line $list_of_lines {
            set key_word [lsearch -exact $line IN]
            if {$key_word != -1} {
                set ssh_id [lindex $line 0]
                exec disconnect ssh $ssh_id
            }
        }
    }

    proc generate_alarm {syslog_msg mnemonic_msg} {

    # store all current syslog global params
         set prev_orig_msg       $::orig_msg
         set prev_timestamp      $::timestamp
         set prev_facility       $::facility
         set prev_mnemonic       $::mnemonic
         set prev_severity       $::severity
         set prev_stream         $::stream
         set prev_traceback      $::traceback
         set prev_pid            $::pid
         set prev_process        $::process
         set prev_format_string  $::format_string
         set prev_msg_args       $::msg_args

         # construct a new syslog with the details of the login failure
         # alarm

         set ::timestamp  [cisco_service_timestamp]
         set ::facility "CC"
         set ::mnemonic $mnemonic_msg
         set ::severity 6
         set ::stream 2
         set ::traceback "cc_internal_syslog"
         set ::pid ""
         set ::process ""
         set ::format_string ""
         set ::msg_args {}
         set ::orig_msg [format "%s %s: %s%s-%d-%s: %s" $::buginfseq $::timestamp "%"
$::facility $::severity $::mnemonic $syslog_msg]

         # Send a syslog to be catched by the script that handles the
```

```
        # alarms
        esm_errmsg 0

        # restore all syslog global params

        set ::orig_msg        $prev_orig_msg
        set ::timestamp       $prev_timestamp
        set ::facility        $prev_facility
        set ::mnemonic        $prev_mnemonic
        set ::severity        $prev_severity
        set ::stream          $prev_stream
        set ::traceback       $prev_traceback
        set ::pid             $prev_pid
        set ::process         $prev_process
        set ::format_string   $prev_format_string
        set ::msg_args        $prev_msg_args

    }

    # handle login failure for a given user
    proc process_login_failure {user} {
        variable  user_failed_logins
        variable  prev_orig_msg
       variable  global_variables

        set current_time    [clock seconds]

        #default values, change if get as cli args
        set alarm_threshold  1
        set time_gap         0

        # Get the list timestamps of previous login failures of this user
        set time_list $user_failed_logins($user)
        set list_len [llength $time_list]

       set time_gap $global_variables("time_gap")
       set alarm_threshold $global_variables("alarm_threshold")
       set total_remain_fails $global_variables("remain_fails")
       set max_allow_fails $global_variables("max_fails")

        if { $max_allow_fails != 0 } {
           set total_remain_fails [expr $total_remain_fails - 1]
           set global_variables("remain_fails") $total_remain_fails

           if { $total_remain_fails == 0 } {
               set global_variables("remain_fails") $max_allow_fails
               set login_fail_msg "Total number of Login failure ($max_allow_fails) was
exceeded, shutting down all VTYs"
               generate_alarm $login_fail_msg "INFO"
               close_all_vty
           }
       }

        if { $time_gap != 0 } {
            set i 0
            # run though the list and keep only the timestamps which are still
            # in the given time gap from current time
            while  { $i < $list_len } {
                if {[expr $current_time - [lindex $time_list $i]] <= $time_gap } {
                    lappend new_time_list [lindex $time_list $i]
                }
                incr i
            }
            # update the timestamp list for the given user
```

```
            set user_failed_logins($user) $new_time_list
        } else {
            set new_time_list $user_failed_logins($user)
        }

        # does the updated timestamp list has more items than the threshold
        if { [llength $new_time_list] >= $alarm_threshold } {
            if { $time_gap != 0 } {
                # Need to send a new login failure alarm.
                set login_fail_msg [format "%s for user %s %d times in %d seconds
interval" [lindex $::msg_args 3] [lindex $::msg_args 0] [llength $new_time_list] [expr
$current_time - [lindex $new_time_list 0]]]
            } else {
                set login_fail_msg [format "%s for user %s %d times" [lindex $::msg_args
3] [lindex $::msg_args 0] [llength $new_time_list]]
            }
           generate_alarm $login_fail_msg "ALARM"
        }
    }

    proc process_syslog {} {
        variable  user_failed_logins

        # empty msg?
        if { [string length $::orig_msg] == 0} {
            return ""
        }

        set category "$::facility-$::severity-$::mnemonic"

        # Should I process this syslog?
        set need_to_process [query_category $category]
        if { $need_to_process == 0 } {
            return $::orig_msg
        }

        # Extract username of the failed login try
        set username "[lindex $::msg_args 0]"
        if { [string length $username] == 0} {
            return $::orig_msg
        }

        # Get current time and add it to the given user list
        set time [clock seconds]
        lappend user_failed_logins($username) $time

        process_login_failure $username
        return $::orig_msg
    }

    array set msg_to_watch {
        SEC_LOGIN-4-LOGIN_FAILED 1
    }

    if { [array size init_variables] == 0 } {
        set init_variables("vars") "init"
        set global_variables("alarm_threshold") "0"
        set global_variables("time_gap") "0"
        set global_variables("max_fails") "0"
        set global_variables("remain_fails") "0"

        # First arg (if exists) is alarm threshlod
        if { [info exists ::cli_args] == 1 } {
            set global_variables("alarm_threshold") [lindex $::cli_args 0]
```

```
            # Second arg (if exists) is time gap
            if { [llength $::cli_args] > 1 } {
                set global_variables("time_gap") [lindex $::cli_args 1]
            }
           # Second arg (if exists) is time gap
           if { [llength $::cli_args] > 2 } {
               set global_variables("max_fails") [lindex $::cli_args 2]
               set global_variables("remain_fails") [lindex $::cli_args 2]
           }
        }
    }

    # Process the message
    process_syslog

} ;# end namespace login_failure
```

Example: Tcl Scripts for Information Flow Violations

```
namespace eval ::monitor_violation {

    # Should I process this message ?
    proc query_category {cat} {
        variable msg_to_watch

        if { [info exists msg_to_watch($cat)] } {
            return $msg_to_watch($cat)
        } else {
            return 0
        }
    }

    proc process_monitor_violation {newMsg} {
        # Need to send a new login failure alarm.
        set monitor_violation_msg $newMsg
        # store all current syslog global params
        set prev_orig_msg       $::orig_msg
        set prev_timestamp      $::timestamp
        set prev_facility       $::facility
        set prev_mnemonic       $::mnemonic
        set prev_severity       $::severity
        set prev_stream         $::stream
        set prev_traceback      $::traceback
        set prev_pid            $::pid
        set prev_process        $::process
        set prev_format_string  $::format_string
        set prev_msg_args       $::msg_args

        # construct a new syslog with the details of the login failure
        # alarm

        set ::timestamp  [cisco_service_timestamp]
        set ::facility "CC"
        set ::mnemonic "ALARM"
        set ::severity 6
        set ::stream 2
        set ::traceback "cc_internal_syslog"
        set ::pid ""
        set ::process ""
```

```
        set ::format_string ""
        set ::msg_args {}
        set ::orig_msg [format "%s %s: %s%s-%d-%s: %s" $::buginfseq $::timestamp "%"
$::facility $::severity $::mnemonic $monitor_violation_msg]

        # Send a syslog to be catched by the script that handles the
        # alarms
        esm_errmsg 0

        # restore all syslog global params

        set ::orig_msg        $prev_orig_msg
        set ::timestamp       $prev_timestamp
        set ::facility        $prev_facility
        set ::mnemonic        $prev_mnemonic
        set ::severity        $prev_severity
        set ::stream          $prev_stream
        set ::traceback       $prev_traceback
        set ::pid             $prev_pid
        set ::process         $prev_process
        set ::format_string   $prev_format_string
        set ::msg_args        $prev_msg_args
    }


    proc process_syslog {} {

        # empty msg?
        if { [string length $::orig_msg] == 0} {
            return ""
        }

        set category "$::facility-$::severity-$::mnemonic"

        # Should I process this syslog?
        set need_to_process [query_category $category]
        if { $need_to_process == 0 } {
            return $::orig_msg
        }

        if { [llength $::msg_args] < 1} {
            return $::orig_msg
        }

       set list_of_args [split $::msg_args " "]

        # now check for MONITOR-6-VIOLATION inside the msg args
       set mon_event_str %MONITOR-6-VIOLATION:
       set vio_loc [lsearch -exact $list_of_args $mon_event_str ]

       if {$vio_loc != -1} {
           set arg1 [lindex $list_of_args [expr $vio_loc + 2]]
           set arg2 [lindex $list_of_args [expr $vio_loc + 4]]
           set arg3 [lindex $list_of_args [expr $vio_loc + 7]]
       } else {
            return $::orig_msg
       }

        set msg [format "Information Flow policy violation for ACL %s logged %d times in
%d seconds" $arg1 $arg2 $arg3]

       process_monitor_violation $msg
        return $::orig_msg
    }
```

```
    array set msg_to_watch {
        IOSXE-6-PLATFORM    1
    }

    # Process the message
    process_syslog

} ;# end namespace monitor_violation
```

Example: Tcl Scripts for VPN Events

```
namespace eval ::monitor_vpn_event {

    # Should I process this message ?
    proc query_category {cat} {
        variable msg_to_watch

        if { [info exists msg_to_watch($cat)] } {
            return $msg_to_watch($cat)
        } else {
            return 0
        }
    }

    proc process_monitor_vpn_event {newMsg} {
        set monitor_vpn_event_msg $newMsg

        # store all current syslog global params
        set prev_orig_msg       $::orig_msg
        set prev_timestamp      $::timestamp
        set prev_facility       $::facility
        set prev_mnemonic       $::mnemonic
        set prev_severity       $::severity
        set prev_stream         $::stream
        set prev_traceback      $::traceback
        set prev_pid            $::pid
        set prev_process        $::process
        set prev_format_string  $::format_string
        set prev_msg_args       $::msg_args

        # construct a new syslog with the details of the login failure
        # alarm

        set ::timestamp  [cisco_service_timestamp]
        set ::facility "CC"
        set ::mnemonic "ALARM"
        set ::severity 6
        set ::stream 2
        set ::traceback "cc_internal_syslog"
        set ::pid ""
        set ::process ""
        set ::format_string ""
        set ::msg_args {}
        set ::orig_msg [format "%s %s: %s%s-%d-%s: %s" $::buginfseq $::timestamp "%"
$::facility $::severity $::mnemonic $monitor_vpn_event_msg]

        # Send a syslog to be catched by the script that handles the
        # alarms
        esm_errmsg 0
```

```
    # restore all syslog global params

    set ::orig_msg        $prev_orig_msg
    set ::timestamp       $prev_timestamp
    set ::facility        $prev_facility
    set ::mnemonic        $prev_mnemonic
    set ::severity        $prev_severity
    set ::stream          $prev_stream
    set ::traceback       $prev_traceback
    set ::pid             $prev_pid
    set ::process         $prev_process
    set ::format_string   $prev_format_string
    set ::msg_args        $prev_msg_args
}


proc process_syslog {} {

    # empty msg?
    if { [string length $::orig_msg] == 0} {
        return ""
    }

    set category "$::facility-$::severity-$::mnemonic"

    # Should I process this syslog?
    set need_to_process [query_category $category]
    if { $need_to_process == 0 } {
        return $::orig_msg
    }

    if { [llength $::msg_args] < 1} {
        return $::orig_msg
    }

   set list_of_args [split $::msg_args " "]

   # now check for MONITOR-3-VPN_EVENT inside the msg args
   set vpn_event_str %MONITOR-3-VPN_EVENT:
   set vpn_loc [lsearch -exact $list_of_args $vpn_event_str ]

   if {$vpn_loc != -1} {
       set arg1 [lindex $list_of_args [expr $vpn_loc + 4]]
       set arg2 [lindex $list_of_args [expr $vpn_loc + 8]]
   } else {
        return $::orig_msg
   }

   set msg [format "Ipsec event type %s occured %d times" $arg1 $arg2]

    process_monitor_vpn_event $msg
    return $::orig_msg
}

array set msg_to_watch {
    IOSXE-3-PLATFORM 1
}

# Process the message
process_syslog

} ;# end namespace monitor_vpn_event
```

Example: Tcl Scripts for Configuring vty Devices

```
namespace eval ::CC_vty_monitor {

    # Should I process this message ?
    proc query_category {cat} {
        variable msg_to_watch

        if { [info exists msg_to_watch($cat)] } {
            return $msg_to_watch($cat)
        } else {
            return 0
        }
    }

    proc process_syslog {} {

        # empty msg?
        if { [string length $::orig_msg] == 0} {
            return ""
        }

        set category "$::facility-$::severity-$::mnemonic"

        # Should I process this syslog?
        set need_to_process [query_category $category]
        if { $need_to_process == 0 } {
            return $::orig_msg
        }

       # got success login - so now conf the VTYs
       set users_output [exec show users wide | inc vty]

       if {[llength $users_output] <= 1} {
            return $::orig_msg
       }


       # split the contents on newlines
       set list_of_lines [split $users_output "\n"]

       set first_vty -1
       set last_vty -1

       # loop through the lines
       foreach line $list_of_lines {
           set vty_location [lsearch -exact $line vty]
           if {$vty_location != -1} {
               set vty_id [lindex $line [expr $vty_location + 1]]
               if {$first_vty == -1} {
                   set first_vty $vty_id
               }
               set last_vty $vty_id
           }
       }
       if {$first_vty == $last_vty} {
           ios_config "line vty $first_vty" "monitor"
       } else {
           ios_config "line vty $first_vty $last_vty" "monitor"
       }

       return $::orig_msg
    }
```

```
    array set msg_to_watch {
        SEC_LOGIN-5-LOGIN_SUCCESS 1
    }

    # Process the message
    process_syslog
} ;
# end CC_vty_monitor
```

Example: Tcl Scripts for Periodic FIPS

```
namespace eval ::CC_periodic_fips {
    array set fips_periodic_started {}

    proc fips_periodic_run {} {
        if { [info exists ::CC_periodic_fips::fips_delta] } {
            set now [clock seconds]
            set tmp_val [expr $::CC_periodic_fips::curr_time +
$::CC_periodic_fips::fips_delta]
            if {$now > $tmp_val} {
                set ::CC_periodic_fips::curr_time $tmp_val
                exec "test crypto self-test"
            }
        }
    }
    # Initialize processes for alonTimer
    if { [array size fips_periodic_started] == 0 } {
        variable fips_periodic_started
        set fips_periodic_started("fips_periodic") "started"

        set curr_time [clock seconds]
        if { [info exists ::cli_args] } {
            set fips_delta $::cli_args
        } else {
            puts "bad cli argument, configure the script again"
        }
    }
    ::CC_periodic_fips::fips_periodic_run

    # just pass the message to next filter
    return $::orig_msg
} ;
# end CC_periodic_fips
```

Example: Tcl Scripts for the IPsec Policy Violation Category Watcher

```
namespace eval ::Ipsec_monitor {
    array set ipsec_monitor_started {}

    array set qfp_options {
        active  1
        standby 2
    }
```

```
array set type_options {
    decrypt-failed  1
    encrypt-failed  2
    replay          3
}

proc query_qfp {cat} {
    variable qfp_options

    if { [info exists qfp_options($cat)] } {
        return $qfp_options($cat)
    } else {
        return 0
    }
}

proc query_type {cat} {
    variable type_options

    if { [info exists type_options($cat)] } {
        return $type_options($cat)
    } else {
        return 0
    }
}

   proc generate_alarm {syslog_msg mnemonic_msg} {

# store all current syslog global params
    set prev_orig_msg       $::orig_msg
    set prev_timestamp      $::timestamp
    set prev_facility       $::facility
    set prev_mnemonic       $::mnemonic
    set prev_severity       $::severity
    set prev_stream         $::stream
    set prev_traceback      $::traceback
    set prev_pid            $::pid
    set prev_process        $::process
    set prev_format_string  $::format_string
    set prev_msg_args       $::msg_args

    # construct a new syslog with the details of the login failure
    # alarm

    set ::timestamp  [cisco_service_timestamp]
    set ::facility "CC"
    set ::mnemonic $mnemonic_msg
    set ::severity 6
    set ::stream 2
    set ::traceback "cc_internal_syslog"
    set ::pid ""
    set ::process ""
    set ::format_string ""
    set ::msg_args {}
    set ::orig_msg [format "%s %s: %s%s-%d-%s: %s" $::buginfseq $::timestamp "%"
$::facility $::severity $::mnemonic $syslog_msg]

    # Send a syslog to be catched by the script that handles the
    # alarms
    esm_errmsg 0

    # restore all syslog global params
```

```
            set ::orig_msg        $prev_orig_msg
            set ::timestamp       $prev_timestamp
            set ::facility        $prev_facility
            set ::mnemonic        $prev_mnemonic
            set ::severity        $prev_severity
            set ::stream          $prev_stream
            set ::traceback       $prev_traceback
            set ::pid             $prev_pid
            set ::process         $prev_process
            set ::format_string   $prev_format_string
            set ::msg_args        $prev_msg_args

        }


        # check if it was already excuted, if not read CLI args
        # verify that they are correct and configure the required command

        if { [array size ipsec_monitor_started] == 0 } {
            variable ipsec_monitor_started
            set ipsec_monitor_started("Ipsec") "started"

            # check if all the args are here
            if { [info exists ::cli_args] == 1 } {

                # check if all 3 args are here
                if { [llength $::cli_args] > 2 } {
                    set qfp   [lindex $::cli_args 0]
                    set type  [lindex $::cli_args 1]
                    set count [lindex $::cli_args 2]

                    set check_input [query_qfp $qfp]
                    if { $check_input == 0 } {
                        generate_alarm "Error: bad arg ($qfp)" "INFO"
                        return $::orig_msg
                    }

                    set check_input [query_type $type]
                    if { $check_input == 0 } {
                        generate_alarm "Error: bad arg ($type)" "INFO"
                        return $::orig_msg
                    }

                    if {$count < 0} {
                        generate_alarm "Error: bad arg ($count)" "INFO"
                        return $::orig_msg
                    }

                    exec set platform hardware qfp $qfp feature ipsec event-monitor type $type
count $count
                }

            }

            return $::orig_msg

        }

        # just pass the message to next filter
        return $::orig_msg
} ;
# end namespace Ipsec_monitor
```

Example: Tcl Scripts for the Exclude Syslog Messages with Keywords

```
namespace eval ::syslog_exclude {
    array set msg_to_confirm {
        PARSER-5-CFGLOG_LOGGEDCMD  1
    }

    proc query_confirm {cat} {
        variable msg_to_confirm

        if { [info exists msg_to_confirm($cat)] } {
            return $msg_to_confirm($cat)
        } else {
            return 0
        }
    }

    proc process_syslog {} {
        # empty msg?
        if { [string length $::orig_msg] == 0} {
            return ""
        }

        if { [info exists ::cli_args] == 0 } {
            return $::orig_msg
        }

        if { $::traceback == "cc_internal_syslog"} {
            # This is common creteria internal syslog
            return $::orig_msg
        }
        set category "$::facility-$::severity-$::mnemonic"
        # Is this a confirmation syslog?
        set need_to_confirm [query_confirm $category]
        if { $need_to_confirm == 1 } {
            if { "event manager environment confirm_alarm" == [lrange [split [lindex
$::msg_args 1]] 0 3]} {
                # This is common creteria internal syslog
                set ::stream  2
                return $::orig_msg
            }
        }
        set args_count [llength $::cli_args]
        set i 0
        while { $i < $args_count} {
            set result [regexp [lindex $::cli_args $i] $::orig_msg]
            if { $result == 1} {
                # found token in $::orig_msg
                return ""
            }
            incr i
        }
        return $::orig_msg
    }

    # Process the message
    process_syslog

} ;# end namespace syslog_exclude
```

Example: Tcl Scripts for the Include Syslog Messages with Keywords

```
namespace eval ::syslog_include {
    array set msg_to_confirm {
        PARSER-5-CFGLOG_LOGGEDCMD  1
    }

    proc query_confirm {cat} {
        variable msg_to_confirm

        if { [info exists msg_to_confirm($cat)] } {
            return $msg_to_confirm($cat)
        } else {
            return 0
        }
    }

    proc process_syslog {} {
        # empty msg?
        if { [string length $::orig_msg] == 0} {
            return ""
        }

        if { [info exists ::cli_args] == 0 } {
            return $::orig_msg
        }

        if {  $::traceback == "cc_internal_syslog"} {
            # This is common creteria internal syslog
            return $::orig_msg
        }
        set category "$::facility-$::severity-$::mnemonic"
        # Is this a confirmation syslog?
        set need_to_confirm [query_confirm $category]
        if { $need_to_confirm == 1 } {
            if { "event manager environment confirm_alarm" == [lrange [split [lindex
$::msg_args 1]] 0 3]} {
                # This is common creteria internal syslog
                set ::stream  2
                return $::orig_msg
            }
        }
        set args_count [llength $::cli_args]
        set i 0
        while { $i < $args_count} {
            set result [regexp [lindex $::cli_args $i] $::orig_msg]
            if { $result == 1} {
                # found token in $::orig_msg
                return $::orig_msg
            }
            incr i
        }
        return ""
    }

    # Process the message
    process_syslog

} ;# end namespace syslog_include
```

Example: Tcl Scripts for Timer Events

```
namespace eval ::Timer {
    array set timer_process_started {}

    proc timer_run {time_interval} {
        set curr_time [clock seconds]

        after $time_interval ::Timer::timer_run $time_interval

        set ::orig_msg "seconds $curr_time"
        set ::timestamp  [cisco_service_timestamp]
        set ::facility "CC"
        set ::mnemonic "TIMER"
        set ::severity 6
        set ::stream 2
        set ::traceback "cc_internal_syslog"
        set ::pid ""
        set ::process ""
        set ::format_string "seconds %d"
        set ::msg_args {$curr_time}
        esm_errmsg 0
    }

    # Initialize processes for Timer
    if { [array size timer_process_started] == 0 } {
        variable timer_process_started
        set timer_process_started("timer") "started"

        #default value 1 minute
        set time_interval 60000
        if { [info exists ::cli_args] == 1 } {
            set time_interval [expr [lindex $::cli_args 0] * 1000]
        }
        after 60000 ::Timer::timer_run $time_interval

       #set the debug flags we want - isakmp & ipsec
       exec debug crypto isakmp error
       exec debug crypto ipsec error
    }

    # just pass the message to next filter
    return $::orig_msg
} ;
# end namespace Timer
```

For More Information

The following sections provide references related to the Common Criteria Tcl Scripts feature.

Related Documents

Related Topic	Document Title
Embedded Syslog Manager	Embedded Syslog Manager module
Network Management Configurations	*Cisco IOS Network Management Configuration Guide*
Network Management commands (including Tcl and logging commands): complete command syntax, defaults, command mode, command history, usage guidelines, and examples	*Cisco IOS Network Management Command Reference*

CHAPTER 2

Monitoring the Control Plane

To verify the overall health of your system, monitor control plane resources on a regular basis.

This chapter includes the following sections:

Avoiding Problems Through Regular Monitoring

Monitoring system resources allows you to detect potential problems before they happen, thus avoiding outages. The following are show the advantages of regular monitoring:

- In a real-life example, customers installed new line cards. After the line cards were in operation for a few years, lack of memory on those line cards caused major outages in some cases. Monitoring memory usage would have identified a memory issue and avoided an outage.
- Regular monitoring establishes a baseline for a normal system load. You can use this information as a basis for comparison when you upgrade hardware or software—to see if the upgrade has affected resource usage.

Control Plane Overview

The following sections contain a high-level overview of the control plane:

Cisco ASR 1000 Series Routers Control Plane Architecture

The major components in the control plane are:

- Cisco ASR 1000 Series Route Processor (RP)—A general purpose CPU responsible for routing protocols, CLI, network management interfaces, code storage, logging, and chassis management. The Cisco ASR 1000 Series RPs process network control packets as well as protocols not supported by the Cisco ASR 1000 Series ESP.
- Cisco ASR 1000 Series Embedded Services Processor (ESP)—A forwarding processor that handles forwarding control plane traffic, and performs packet processing functions such as firewall inspection, ACLs, encryption, and QoS.
- Cisco ASR 1000 Series SPA Interface Processor (SIP)—An interface processor that provides the connection between the Route Processor and the shared port adapters (SPAs).

Distributed Control Plane Architecture

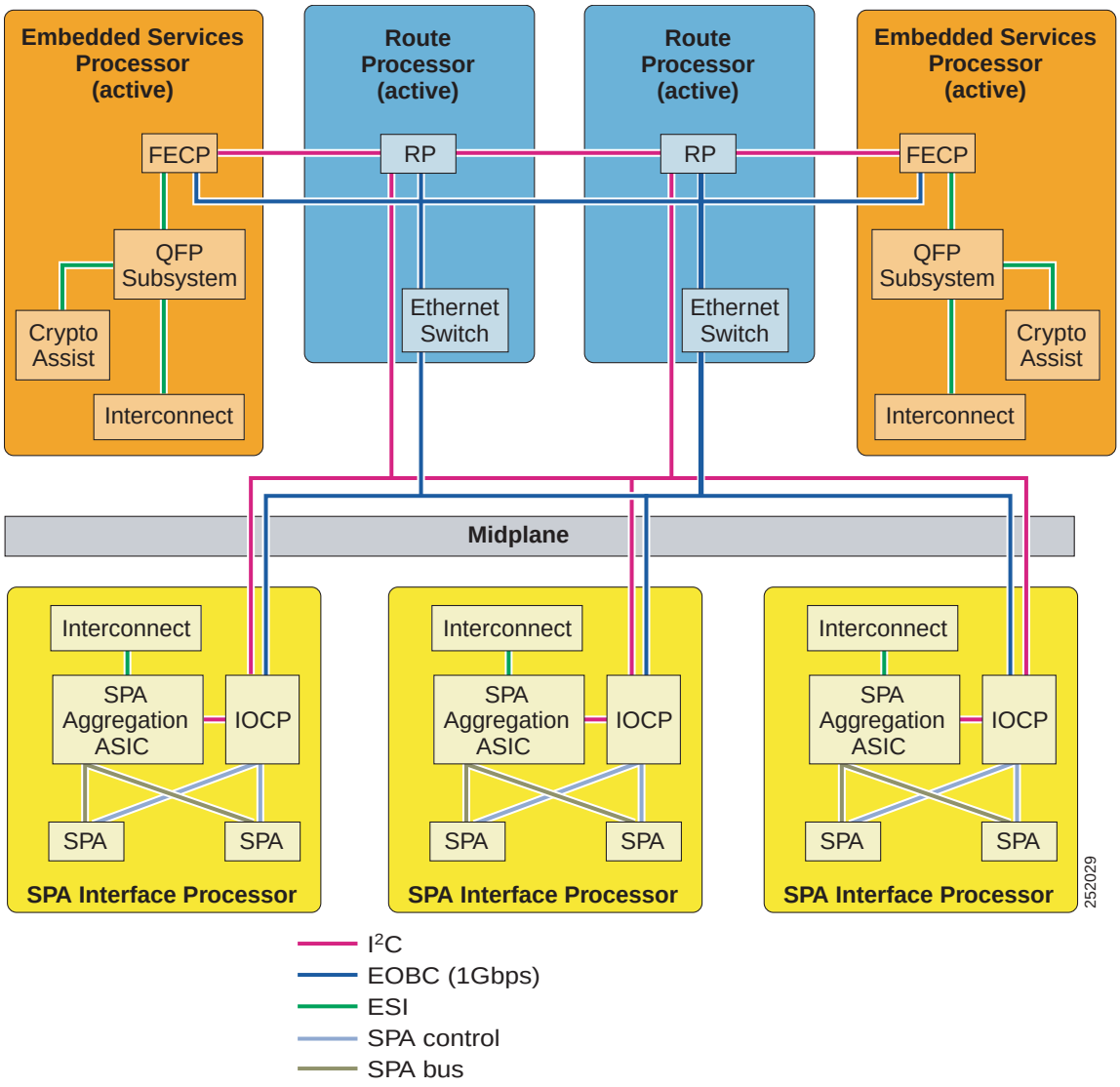
Cisco ASR 1000 Series Routers have a distributed control plane architecture. A separate control processor is embedded on each major component in the control plane, as shown in Figure 2-1:

- Route Processor (RP)
- Forwarding Engine Control Processor (FECP)
- I/O Control Processor (IOCP)

The RP manages and maintains the control plane using a dedicated Gigabit Ethernet out-of-band channel (EOBC). The internal EOBC is used to continuously exchange system state information among the different major components. For example, in the event of a failure condition, a switchover event occurs and the standby RP and ESP are immediately ready to assume the data forwarding functions or the control plane functions for the failed component.

The inter-integrated circuit (I^2C) monitors the health of hardware components. The Enhanced SerDes Interconnect (ESI) is a set of serial links that are the data path links on the midplane connecting the RP, SIPs, and standby ESPs to the active ESP.

Figure 2-1 Cisco ASR 1000 Series Routers Control Plane Architecture



The control plane processors perform the following functions:

RP

- Runs the router control plane (Cisco IOS), including processing network control packets, computing routes, and setting up connections.
- Monitors interface and environmental status, including management ports, LEDs, alarms, and SNMP network management.
- Downloads code to other components in the system.
- Selects the active RP and ESP and synchronizes the standby RP and ESP.
- Manages logging facilities, on-board failure logging (OBFL), and statistics aggregation.

FECP

- Provides direct CPU access to the forwarding engine subsystem—the Cisco QuantumFlowProcessor (QFP) subsystem—that is the forwarding processor chipset and also resides on the ESP.
- Manages the forwarding engine subsystem and its connection to I/O.
- Manages the forwarding processor chipset.

IOCP

- Provides direct CPU access to SPAs installed in a SIP.
- Manages the SPAs.
- Handles SPA online insertion and removal (OIR) events.
- Runs SPA drivers that initialize and configure SPAs.

Cisco IOS XE Software Architecture

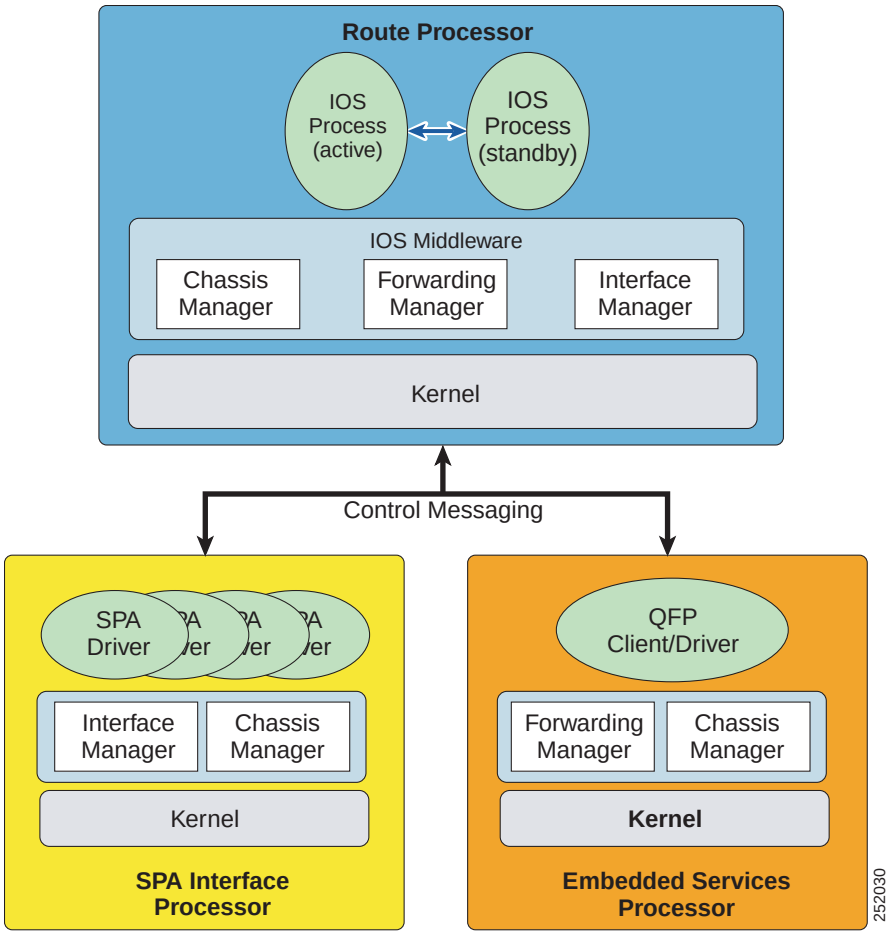
The control plane processors run Cisco IOS XE software, which is an operating system that consists of a Linux-based kernel and a common set of operating system-level utility programs. It is a distributed software architecture that moves many operating system responsibilities out of the IOS process.

In this architecture, IOS runs as one of many Linux processes while allowing other Linux processes to share responsibility for running the router. IOS runs as a user process on the RP. Hardware-specific components have been removed from the IOS process and are handled by separate middleware processes in Cisco IOS XE software. If a hardware-specific issue is discovered, the middleware process can be modified without touching the IOS process.

Figure 2-2 shows the main components of the Cisco IOS XE software architecture. This modular architecture increases network resiliency by distributing operating responsibility among separate processes. The architecture also allows for better allocation of memory so the router can run more efficiently.

All of the Cisco IOS XE software modules run in their own protective memory spaces, which facilitates fault containment. Any software outages of an individual software module are localized to that particular module. All other software processes continue to operate. For example, for each SPA, a separate driver process is executed on the SIP, even if multiple SPAs of the same type are present. Because each SPA driver runs in its own protective memory, failure or upgrade of an individual driver is localized to the affected SPA.

Figure 2-2 Cisco IOS XE Software Architecture



Using the Linux architecture, Cisco IOS XE provides the following benefits:

- The ability to integrate multi-core (multiple CPUs on a single piece of silicon) processors.
- The IOS process has no direct access to hardware components, thus providing a greater level of resiliency.
- The ability to run active and standby IOS processes on the non-hardware-redundant Cisco ASR 1004 Router and Cisco ASR 1006 Router.
- The IOS process operates as a virtual machine under the RP Linux kernel. Upon bootup, the RP Linux kernel allocates 50 percent of available memory to IOS processes as a one-time event. For systems that have a single IOS process, IOS is allocated approximately 45 percent of total RP memory. For redundant IOS process systems, each IOS process is allocated approximately 20 percent of total RP memory.
- Hardware components are managed through memory-protected middleware processes.
- SPA drivers run as unique processes allowing the ability to upgrade and restart individual SPAs.

Monitoring Control Plane Resources

The following sections discuss monitoring memory and CPU from the perspective of the IOS process and from the perspective of the overall control plane:

IOS Process Resources

For information about memory and CPU utilization from within the IOS process, use the **show memory** command and the **show process cpu** command. Note that these commands provide a representation of memory and CPU utilization from the perspective of the IOS process only; they do not include information for resources on the entire route processor. For example, **show memory** on an RP2 with 8 GB of RAM running a single IOS process shows the following memory usage:

```
Router# show memory

                Head       Total(b)     Used(b)     Free(b)     Lowest(b)   Largest(b)
Processor  2ABEA4316010   4489061884   314474916   4174586968   3580216380   3512323496
 lsmpi_io  2ABFAFF471A8     6295128     6294212         916          916          916
Critical   2ABEB7C72EB0     1024004          92     1023912      1023912      1023912
```

For the dual-core RP2, the **show process cpu** command reports a single IOS CPU utilization average using both processors:

```
Router# show process cpu

CPU utilization for five seconds: 0%/0%; one minute: 0%; five minutes: 0%
 PID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min TTY Process
   1         583     48054         12  0.00%  0.00%  0.00%   0 Chunk Manager
   2         991    176805          5  0.00%  0.00%  0.00%   0 Load Meter
   3           0         2          0  0.00%  0.00%  0.00%   0 IFCOM Msg Hdlr
   4           0        11          0  0.00%  0.00%  0.00%   0 Retransmission o
   5           0         3          0  0.00%  0.00%  0.00%   0 IPC ISSU Dispatc
   6      230385    119697       1924  0.00%  0.01%  0.00%   0 Check heaps
   7          49        28       1750  0.00%  0.00%  0.00%   0 Pool Manager
   8           0         2          0  0.00%  0.00%  0.00%   0 Timers
   9       17268    644656         26  0.00%  0.00%  0.00%   0 ARP Input
  10         197    922201          0  0.00%  0.00%  0.00%   0 ARP Background
  11           0         2          0  0.00%  0.00%  0.00%   0 ATM Idle Timer
  12           0         1          0  0.00%  0.00%  0.00%   0 ATM ASYNC PROC
  13           0         1          0  0.00%  0.00%  0.00%   0 AAA_SERVER_DEADT
  14           0         1          0  0.00%  0.00%  0.00%   0 Policy Manager
  15           0         2          0  0.00%  0.00%  0.00%   0 DDR Timers
  16           1        15         66  0.00%  0.00%  0.00%   0 Entity MIB API
  17          13      1195         10  0.00%  0.00%  0.00%   0 EEM ED Syslog
  18          93        46       2021  0.00%  0.00%  0.00%   0 PrstVbl
  19           0         1          0  0.00%  0.00%  0.00%   0 RO Notify Timers
```

Overall Control Plane Resources

For information about control plane memory and CPU utilization on each control processor, use the **show platform software status control-processor brief** command (summary view) or the **show platform software status control-processor** command (detailed view).

All control processors should show a status of Healthy. Other possible status values are Warning and Critical. Warning indicates that the router is operational but that the operating level should be reviewed. Critical implies that the router is near failure.

If you see a status of Warning or Critical, take the following actions:

- Reduce static and dynamic loads on the system by reducing the number of elements in the configuration or by limiting the capacity for dynamic services.
- Reduce the number of routes and adjacencies, limit the number of ACLs and other rules, reduce the number of VLANs, and so on.

The following sections describe the fields in **show platform software status control-processor** command output.

Load Average

Load average represents the process queue or process contention for CPU resources. For example, on a single-core processor, an instantaneous load of 7 would mean that seven processes are ready to run, one of which is currently running. On a dual-core processor, a load of 7 would represent seven processes are ready to run, two of which are currently running.

Memory Utilization

Memory utilization is represented by the following fields:

- Total—Total line card memory
- Used—Consumed memory
- Free—Available memory
- Committed—Virtual memory committed to processes

CPU Utilization

CPU utilization is an indication of the percentage of time the CPU is busy and is represented by the following fields:

- CPU—The allocated processor
- User—Non-Linux kernel processes
- System —Linux kernel process
- Nice—Low priority processes
- Idle—Percentage of time the CPU was inactive
- IRQ—Interrupts
- SIRQ—System Interrupts
- IOwait—Percentage of time CPU was waiting for I/O

The following are examples of the **show platform software status control-processor** command.

```
Router# show platform software status control-processor brief
Load Average
 Slot  Status  1-Min  5-Min 15-Min
  RP0 Healthy   0.25   0.30   0.44
  RP1 Healthy   0.31   0.19   0.12
 ESP0 Healthy   0.01   0.05   0.02
 ESP1 Healthy   0.03   0.05   0.01
 SIP1 Healthy   0.15   0.07   0.01
 SIP2 Healthy   0.03   0.03   0.00

Memory (kB)
 Slot  Status    Total     Used (Pct)     Free (Pct) Committed (Pct)
  RP0 Healthy  3722408  2514836 (60%)  1207572 (29%)   1891176 (45%)
  RP1 Healthy  3722408  2547488 (61%)  1174920 (28%)   1889976 (45%)
 ESP0 Healthy  2025468  1432088 (68%)   593380 (28%)   3136912 (149%)
 ESP1 Healthy  2025468  1377980 (65%)   647488 (30%)   3084412 (147%)
 SIP1 Healthy   480388   293084 (55%)   187304 (35%)    148532 (28%)
 SIP2 Healthy   480388   273992 (52%)   206396 (39%)     93188 (17%)

CPU Utilization
 Slot  CPU   User System   Nice   Idle    IRQ   SIRQ IOwait
  RP0    0  30.12   1.69   0.00  67.63   0.13   0.41   0.00
  RP1    0  21.98   1.13   0.00  76.54   0.04   0.12   0.16
 ESP0    0  13.37   4.77   0.00  81.58   0.07   0.19   0.00
 ESP1    0   5.76   3.56   0.00  90.58   0.03   0.05   0.00
 SIP1    0   3.79   0.13   0.00  96.04   0.00   0.02   0.00
 SIP2    0   3.50   0.12   0.00  96.34   0.00   0.02   0.00


Router# show platform software status control-processor
RP0: online, statistics updated 10 seconds ago
Load Average: healthy
  1-Min: 0.30, status: healthy, under 5.00
  5-Min: 0.31, status: healthy, under 5.00
  15-Min: 0.47, status: healthy, under 5.00
Memory (kb): healthy
  Total: 3722408
  Used: 2514776 (60%), status: healthy, under 90%
  Free: 1207632 (29%), status: healthy, over 10%
  Committed: 1891176 (45%), status: healthy, under 90%
Per-core Statistics
CPU0: CPU Utilization (percentage of time spent)
  User: 30.12, System:  1.69, Nice:  0.00, Idle: 67.63
  IRQ:  0.13, SIRQ:  0.41, IOwait:  0.00

RP1: online, statistics updated 5 seconds ago
Load Average: healthy
  1-Min: 0.14, status: healthy, under 5.00
  5-Min: 0.11, status: healthy, under 5.00
  15-Min: 0.09, status: healthy, under 5.00
Memory (kb): healthy
  Total: 3722408
  Used: 2547488 (61%), status: healthy, under 90%
  Free: 1174920 (28%), status: healthy, over 10%
  Committed: 1889976 (45%), status: healthy, under 90%
Per-core Statistics
CPU0: CPU Utilization (percentage of time spent)
  User: 21.98, System:  1.13, Nice:  0.00, Idle: 76.54
  IRQ:  0.04, SIRQ:  0.12, IOwait:  0.16

ESP0: online, statistics updated 5 seconds ago
Load Average: healthy
  1-Min: 0.06, status: healthy, under 5.00
  5-Min: 0.09, status: healthy, under 5.00
```

```
  15-Min: 0.03, status: healthy, under 5.00
Memory (kb): healthy
  Total: 2025468
  Used: 1432088 (68%), status: healthy, under 90%
  Free: 593380 (28%), status: healthy, over 10%
  Committed: 3136912 (149%), status: healthy, under 300%
Per-core Statistics
CPU0: CPU Utilization (percentage of time spent)
  User: 13.37, System:  4.77, Nice:  0.00, Idle: 81.58
  IRQ:  0.07, SIRQ:  0.19, IOwait:  0.00

ESP1: online, statistics updated 5 seconds ago
Load Average: healthy
  1-Min: 0.22, status: healthy, under 5.00
  5-Min: 0.08, status: healthy, under 5.00
  15-Min: 0.02, status: healthy, under 5.00
Memory (kb): healthy
  Total: 2025468
  Used: 1377980 (65%), status: healthy, under 90%
  Free: 647488 (30%), status: healthy, over 10%
  Committed: 3084412 (147%), status: healthy, under 300%
Per-core Statistics
CPU0: CPU Utilization (percentage of time spent)
  User:  5.76, System:  3.56, Nice:  0.00, Idle: 90.58
  IRQ:  0.03, SIRQ:  0.05, IOwait:  0.00

SIP1: online, statistics updated 6 seconds ago
Load Average: healthy
  1-Min: 0.05, status: healthy, under 5.00
  5-Min: 0.06, status: healthy, under 5.00
  15-Min: 0.00, status: healthy, under 5.00
Memory (kb): healthy
  Total: 480388
  Used: 293084 (55%), status: healthy, under 90%
  Free: 187304 (35%), status: healthy, over 10%
  Committed: 148532 (28%), status: healthy, under 90%
Per-core Statistics
CPU0: CPU Utilization (percentage of time spent)
  User:  3.79, System:  0.13, Nice:  0.00, Idle: 96.04
  IRQ:  0.00, SIRQ:  0.02, IOwait:  0.00

SIP2: online, statistics updated 8 seconds ago
Load Average: healthy
  1-Min: 0.03, status: healthy, under 5.00
  5-Min: 0.03, status: healthy, under 5.00
  15-Min: 0.00, status: healthy, under 5.00
Memory (kb): healthy
  Total: 480388
  Used: 273992 (52%), status: healthy, under 90%
  Free: 206396 (39%), status: healthy, over 10%
  Committed: 93188 (17%), status: healthy, under 90%
Per-core Statistics
CPU0: CPU Utilization (percentage of time spent)
  User:  3.50, System:  0.12, Nice:  0.00, Idle: 96.34
  IRQ:  0.00, SIRQ:  0.02, IOwait:  0.00
```

For More Information

For more information about the topics discussed in this chapter, see the following documents:

Topic	Document
Command descriptions	Cisco IOS Master Command List, All Releases Command Lookup Tool (Requires Cisco.com user ID and password)

CHAPTER 3

Performing File System Cleanups

This chapter describes the various file system cleanups performed on the Cisco ASR 1000 Series Aggregation Services Router.

This chapter includes the following sections:

Performing Core File and Trace File Cleanups

Core and trace files are automatically created and saved to the core and tracelogs directories on the harddisk: file system on all Cisco ASR 1000 Series Routers except the Cisco ASR 1001 Router, Cisco ASR 1002 Router and Cisco ASR 1002-F Router, which store core and trace files in the bootflash: file system.

Trace files are automatically created during normal operation of the router and are stored in the tracelogs directory. Normally, the router automatically purges the old tracelogs to provide space for new files.

In case of a process failure, core files may be generated in the core directory. If any core files are detected, contact Cisco TAC for assistance. Normally, the router will automatically purge the old core files to make space for new files.

The user has the option to delete unneeded core and tracelog files to make space for other content. However, such a removal may impact the debuggability of the system.

Note On an Cisco ASR 1001 Router equipped with the HDD IDC option, the ROMMON cannot see the hard disk and therefore the ROMMON has no access to the data stored on the HDD IDC. Though you can copy the Image to a harddisk on Cisco ASR 1001-HDD Router, you cannot boot from the hardisk. The ASR1001 HDD can be used for general storage but cannot be considered a complete filesystem (like bootflash or USB0:) and is only accessible when the box is running IOS. The true intent of the HDD is to support applications that require a harddisk. Therefore the HDD IDC on Cisco ASR 1001 Router should be used for application services such as call manager, etc.

To clean up the contents of the core and tracelogs directories, perform the following steps:

Step 1 Log in to the Cisco ASR 1000 Series Router using a Telnet or Secure Shell (SSH) connection.

Note The core and tracelogs directories can contain large volumes of output. Be sure to use a Telnet or SSH connection instead of the console port to avoid monopolizing the console port.

Step 2 Change to the core or tracelogs directory using the **cd** command.

```
Router# cd harddisk:/tracelogs
```

Step 3 Display the contents of the core or tracelogs directory using the **dir** command.

```
Router# dir
Directory of harddisk:/tracelogs/

753666  -rwx         164  Sep 14 2008 22:06:55 +01:00  inst_cleanup_R0-0.log.145
753667  -rwx         165  Sep 14 2008 21:01:41 +01:00  inst_cleanup_R0-0.log.221
753668  -rwx         165  Sep 14 2008 20:01:29 +01:00  inst_cleanup_R0-0.log.119
753669  -rwx         165  Sep 14 2008 20:06:30 +01:00  inst_cleanup_R0-0.log.110
753670  -rwx         165  Sep 14 2008 20:11:31 +01:00  inst_cleanup_R0-0.log.121
753671  -rwx         165  Sep 14 2008 20:16:32 +01:00  inst_cleanup_R0-0.log.132
753672  -rwx         165  Sep 14 2008 20:21:33 +01:00  inst_cleanup_R0-0.log.143
753673  -rwx         165  Sep 14 2008 20:26:34 +01:00  inst_cleanup_R0-0.log.154
753676  -rwx         165  Sep 14 2008 20:31:35 +01:00  inst_cleanup_R0-0.log.165
753677  -rwx         165  Sep 14 2008 20:36:36 +01:00  inst_cleanup_R0-0.log.176
753678  -rwx         165  Sep 14 2008 20:41:37 +01:00  inst_cleanup_R0-0.log.187
753679  -rwx         165  Sep 14 2008 20:46:38 +01:00  inst_cleanup_R0-0.log.198
753680  -rwx         165  Sep 14 2008 20:51:39 +01:00  inst_cleanup_R0-0.log.199
753681  -rwx         165  Sep 14 2008 20:56:40 +01:00  inst_cleanup_R0-0.log.200
753674  -rwx         165  Sep 14 2008 21:06:42 +01:00  inst_cleanup_R0-0.log.232
753675  -rwx         165  Sep 14 2008 21:11:43 +01:00  inst_cleanup_R0-0.log.233
753682  -rwx         165  Sep 14 2008 21:16:44 +01:00  inst_cleanup_R0-0.log.244
753683  -rwx         165  Sep 14 2008 21:21:45 +01:00  inst_cleanup_R0-0.log.255
753684  -rwx         165  Sep 14 2008 21:26:46 +01:00  inst_cleanup_R0-0.log.266

. . .

39313059840 bytes total (38428729344 bytes free)
```

Step 4 Remove files from the core or tracelogs directory using the **delete** command. Delete files based on their creation date; that is, delete older files first.

```
Router# delete inst_cleanup_R0-0*
```

Caution Core and trace files can be deleted; do not delete the core and tracelogs directories.

Step 5 Repeat Step 2 through Step 4 for all the core and tracelogs directories on the router as follows:

- For Cisco ASR 1006 Routers, perform the file cleanup on the harddisk: file system on both RPs.
- For Cisco ASR 1004 Routers, perform the file cleanup on the harddisk: file system on the single RP.
- For Cisco ASR 1002 Routers and Cisco ASR 1002-F Routers, perform the file cleanup on the bootflash: file system. (The harddisk: file system is not available.)

Performing Crashinfo File Cleanups

Crashinfo files are automatically created and saved to the bootflash: or harddisk: file systems on all Cisco ASR 1000 Series Routers. Delete unneeded crashinfo files at least once a week to maintain optimal router operation.

To delete crashinfo files, perform the following steps:

Step 1 Log in to the Cisco ASR 1000 Series Router using a Telnet or Secure Shell (SSH) connection.

Note Crashinfo files may generate large volumes of output. Be sure to use a Telnet or SSH connection instead of the console port to avoid monopolizing the console port.

Step 2 Change to the bootflash: or harddisk: directory using the **cd** command.

```
Router# cd harddisk:
```

Step 3 Display the contents of the directory using the **dir** command.

```
Router# dir
Directory of harddisk:/

   11  drwx       16384   Dec 4 2007 12:23:10 +00:00  lost+found
557057  drwx        4096   Aug 4 2008 23:10:46 +01:00  core
   12  -rw-           0   Dec 4 2007 12:24:35 +00:00  tracelogs.780
753665  drwx      167936  Sep 14 2008 22:27:00 +01:00  tracelogs
   13  -rw-      234250   Feb 1 2008 05:56:59 +00:00  crashinfo_SIP_01_00_20080C
   14  -rw-       46853  Apr 10 2008 00:50:12 +01:00  tech_support_ouput.tgz.tgz
   15  -rw-   225308932  Aug 13 2008 22:50:29 +01:00  2008-08-10_14.32.rp_supern
   16  -rw-   208904396  Aug 20 2008 21:20:33 +01:00  asr1000rp1-adventerprisekn

39313059840 bytes total (38428712960 bytes free)
```

Step 4 Delete crashinfo files using the **delete** command.

```
Router# delete crashinfo_SIP_01_00_20080C
```

Step 5 Repeat Step 2 through Step 4 for the other file system.
For Cisco ASR 1006 Routers, purge the crashinfo files on both RPs.

Performing Sub-Package File Cleanups

A consolidated package file can be stored in the bootflash: file system, on a USB Flash disk, or on any TFTP or other network server. Individual sub-package files and provisioning files must be stored in the bootflash: file system.

A sub-package file is no longer in use when it is no longer referenced by the booted or specified provisioning manager. Remove sub-package files and provisioning files that are no longer in use to maintain optimal router operation.

To delete sub-package files and provisioning files that are no longer in use, use the **request platform software package clean** command. This command checks to see which sub-package files and provisioning files are in use and deletes only those files that are *not* in use.

Example: Deleting All Unused Sub-Package Files and Provisioning Files From a Boot Directory

The following example shows how to delete all unused sub-package files and provisioning files from a boot directory:

```
Router# request platform software package clean
Cleaning up unnecessary package files
No path specified, will use booted path harddisk:packages.conf
Cleaning harddisk:
  Scanning boot directory for packages ... done.
  Preparing packages list to delete ...
    asr1000rp1-espbase.02.03.00.122-33.XNC.pkg
      File is in use, will not delete.
    asr1000rp1-rpaccess.02.03.00.122-33.XNC.pkg
      File is in use, will not delete.
    asr1000rp1-rpbase.02.03.00.122-33.XNC.pkg
      File is in use, will not delete.
    asr1000rp1-rpcontrol.02.03.00.122-33.XNC.pkg
      File is in use, will not delete.
    asr1000rp1-rpios-adventerprisek9.02.03.00.122-33.XNC.pkg
      File is in use, will not delete.
    asr1000rp1-sipbase.02.03.00.122-33.XNC.pkg
      File is in use, will not delete.
    asr1000rp1-sipspa.02.03.00.122-33.XNC.pkg
      File is in use, will not delete.
    packages.conf
      File is in use, will not delete.
  done.

Files that will be deleted:
  packages.con.00
  packages.conf.copy
  testing1,pkg
  testing1.pkg

Do you want to proceed? [confirm]y
  Deleting file harddisk:packages.con.00 ... done.
  Deleting file harddisk:packages.conf.copy ... done.
  Deleting file harddisk:testing1,pkg ... done.
  Deleting file harddisk:testing1.pkg ... done.
SUCCESS: Files deleted.
```

The following example shows all sub-package files and provisioning files in a boot directory. If they are in use, they cannot be deleted:

```
Router# request platform software package clean
Cleaning up unnecessary package files
No path specified, will use booted path harddisk:packages.conf
Cleaning harddisk:
  Scanning boot directory for packages ... done.
  Preparing packages list to delete ...
    asr1000rp1-espbase.02.03.00.122-33.XNC.pkg
      File is in use, will not delete.
    asr1000rp1-rpaccess.02.03.00.122-33.XNC.pkg
      File is in use, will not delete.
    asr1000rp1-rpbase.02.03.00.122-33.XNC.pkg
      File is in use, will not delete.
    asr1000rp1-rpcontrol.02.03.00.122-33.XNC.pkg
      File is in use, will not delete.
    asr1000rp1-rpios-adventerprisek9.02.03.00.122-33.XNC.pkg
      File is in use, will not delete.
    asr1000rp1-sipbase.02.03.00.122-33.XNC.pkg
      File is in use, will not delete.
    asr1000rp1-sipspa.02.03.00.122-33.XNC.pkg
```

```
      File is in use, will not delete.
    packages.conf
      File is in use, will not delete.
  done.

SUCCESS: No extra package or provisioning files found on media. Nothing to clean.
```

Example: Deleting a Specific Sub-Package File from a Boot Directory

The following example shows how to delete a specific sub-package file from a boot directory:

```
Router# request platform software package clean file harddisk:testing1.pkg
Cleaning up unnecessary package files
  Scanning boot directory for packages ... ^./testing1.pkg$ /harddisk/
done.
  Preparing packages list to delete ...
  done.

Files that will be deleted:
  testing1.pkg

Do you want to proceed? [confirm]y
  Deleting file harddisk:testing1.pkg ... done.
SUCCESS: Files deleted.
```

The following example shows that a specific sub-package file cannot be deleted if it is in use:

```
Router# request platform software package clean file harddisk:packages.conf
Cleaning up unnecessary package files
  Scanning boot directory for packages ... done.
  Preparing packages list to delete ...
    packages.conf
      File is in use, will not delete.
  done.

SUCCESS: No extra package or provisioning files found on media. Nothing to clean.
```

Example: Deleting a Duplicate Sub-Package File on Different Media

The following example shows how to delete a sub-package file that was copied and has the same name as the file that was used to boot, but the duplicate file is on different media:

```
Router# request platform software package clean file bootflash:packages.conf
Cleaning up unnecessary package files
  Scanning boot directory for packages ... done.
  Preparing packages list to delete ...
  done.

Files that will be deleted:
  packages.conf

Do you want to proceed? [confirm]y
  Deleting file bootflash:packages.conf ... done.
SUCCESS: Files deleted.
```

For More Information

For more information about the topics discussed in this chapter, see the following documents:

Topic	Document
Command descriptions	Cisco IOS Master Command List, All Releases Command Lookup Tool (Requires Cisco.com user ID and password)

CHAPTER 4

Upgrading System Software

The Cisco ASR 1000 Series Routers introduce a new software packaging model and architecture. The new software packaging model and upgrade processes are described in the the *Cisco ASR 1000 Series Aggregation Services Routers Software Configuration Guide*.

For information, see the following chapters in that guide:

Topic	Chapter
Software packaging model and architecture	"Software Packaging and Architecture"
Upgrading system software offline	"Consolidated Packages and Sub-Package Management"
Interoperability of Cisco IOS XE software releases	"Cisco IOS XE Software Package Compatibility for ISSU"
ISSU Upgrade Procedure	"Software Upgrade Process"
Using Sub-packages for Software Upgrade on a Cisco ASR 1001 Router	"Software Upgrade Process"

CHAPTER 1

Performing Factory Reset

This chapter describes Factory Reset feature and how it can be used to protect or restore a router to an earlier fully functional state.

Understanding How Factory Reset Works

The Factory Reset feature is used to remove all sensitive information from a router or restore the router to a fully functional state.

The factory reset process uses the **factory reset–all** command to take backup of existing configuration and then reset the router to an earlier fully functional state. In a high availability setup, the factory reset process is executed on the active Route Processor (RP) and is then synchronized to the standby RP. The duration of the factory reset process is dependent on the storage size of the router. It can extend between 30 minutes on an ASR1000 consolidated platform and up to 3 hours on a high availability setup.

Table 1 covers details of data erased or retained during the factory reset process:

Data Erased	Data Retained
Non-volatile random-access memory (NVRAM) data	Data from remote field-replaceable units (FRUs).
OBFL (Onboard Failure Logging) logs	Value of configuration register
Licenses	Contents of USB
User data, startup, and running configuration	Credentials (Secure Unique Device Identifier [SUDI] certificates, public key infrastructure (PKI) keys, and FIPS-related keys)

Data Erased	Data Retained
ROMMON variables	
All writable file systems and personal data. Note: The factory reset process takes a backup of the boot image if the system is booted from an image stored locally (bootflash or harddisk). If the current boot image is a remote image or stored on an USB, NIM-SSD or such, ensure that you take a backup of the image before starting the factory reset process	

After the factory reset process is complete, the router reboots to ROMMON mode. If you have the zero-touch provisioning (ZTP) capability setup, after the router completes the factory reset procedure, the router reboots with ZTP configuration.

Software and Hardware Support

- This feature is introduced starting from IOS XE Fuji 16.7.1 release.
- This feature is supported on all Cisco ASR 1000 platforms and Cisco ASR 1000 Series Route Processor 2 (RP2), and Cisco ASR 1000 Series Route Processor 3 (RP3)
- Factory reset process is supported on standalone routers and also on routers configured for high availability.

Prerequisites

- Ensure that all the software images, configurations and personal data is backed up before performing the factory reset operation.
- Ensure that there is uninterrupted power supply when the feature reset process is in progress.
- The factory reset process takes a backup of the boot image if the system is booted from an image stored locally (bootflash or harddisk). If the current boot image is a remote image or stored on an USB, NIM-SSD or such, ensure that you take a backup of the image before starting the factory reset process.
- Ensure that ISSU/ISSD (In- Service Software Upgrade or Downgrade) is not in progress before starting the factory reset process.

Restrictions

- Any software patches that are installed on the router will not be restored after the factory reset operation.
- If factory reset command is issued through a Virtual Teletype (VTY) session, the session is not restored after completion of factory reset process.

When to Use Factory Reset

- **Return Material Authorization (RMA):** If a router is returned back to Cisco for RMA, it is important that all sensitive information is removed.
- **Router is Compromised:** If the router data is compromised due to a malicious attack, the router must be reset to factory configuration and then reconfigured once again for further use**.**
- **Repurposing:** The router needs to be moved to a new topology or market from the existing site to a different site.

What Happens after Factory Reset

After factory reset is successfully completed, the router boots up. Before factory reset process is started, if the configuration register on the router is set to manually boot from ROMMON, then after factory reset the router will stop at ROMMON.

The factory reset process takes a backup of the boot image if the system is booted from an image stored locally (bootflash or harddisk). If the current boot image is a remote image or stored on an USB, NIM-SSD or such, ensure that you take a backup of the image before starting the factory reset process.