Cisco IOS Network Management Configuration Guide

Release 12.2SR

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

Cisco IOS Network Management Configuration Guide

About Cisco IOS Software Documentation

Last Updated: November 20, 2009

This document describes the objectives, audience, conventions, and organization used in Cisco IOS software documentation. Also included are resources for obtaining technical assistance, additional documentation, and other information from Cisco. This document is organized into the following sections:

Documentation Objectives

Cisco IOS documentation describes the tasks and commands available to configure and maintain Cisco networking devices.

Audience

The Cisco IOS documentation set is intended for users who configure and maintain Cisco networking devices (such as routers and switches) but who may not be familiar with the configuration and maintenance tasks, the relationship among tasks, or the Cisco IOS commands necessary to perform particular tasks. The Cisco IOS documentation set is also intended for those users experienced with Cisco IOS software who need to know about new features, new configuration options, and new software characteristics in the current Cisco IOS release.

Documentation Conventions

In Cisco IOS documentation, the term *router* may be used to refer to various Cisco products; for example, routers, access servers, and switches. These and other networking devices that support Cisco IOS software are shown interchangeably in examples and are used only for illustrative purposes. An example that shows one product does not necessarily mean that other products are not supported.

This section contains the following topics:

Typographic Conventions

Cisco IOS documentation uses the following typographic conventions:

Convention	Description
^ or Ctrl	Both the ^ symbol and Ctrl represent the Control (Ctrl) key on a keyboard. For example, the key combination **^D** or **Ctrl-D** means that you 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 a Simple Network Management Protocol (SNMP) community string to *public*, do not use quotation marks around the string; otherwise, the string will include the quotation marks.

Command Syntax Conventions

Cisco IOS documentation uses the following command syntax conventions:

Convention	Description
bold	Bold text indicates commands and keywords that you enter as shown.
italic	Italic text indicates arguments for which you supply values.
[x]	Square brackets enclose an optional keyword or argument.
...	An ellipsis (three consecutive nonbolded periods without spaces) after a syntax element indicates that the element can be repeated.
|	A vertical line, called a pipe, that is enclosed within braces or square brackets indicates a choice within a set of keywords or arguments.
[x | y]	Square brackets enclosing keywords or arguments separated by a pipe indicate an optional choice.
{x | y}	Braces enclosing keywords or arguments separated by a pipe indicate a required choice.
[x {y | z}]	Braces and a pipe within square brackets indicate a required choice within an optional element.

Software Conventions

Cisco IOS software uses the following program code conventions:

Convention	Description
`Courier font`	Courier font is used for information that is displayed on a PC or terminal screen.
`Bold Courier font`	Bold Courier font indicates text that the user must enter.
< >	Angle brackets enclose text that is not displayed, such as a password. Angle brackets also are used in contexts in which the italic font style is not supported; for example, ASCII text.
!	An exclamation point at the beginning of a line indicates that the text that follows is a comment, not a line of code. An exclamation point is also displayed by Cisco IOS software for certain processes.
[ ]	Square brackets enclose default responses to system prompts.

Reader Alert Conventions

Cisco IOS documentation uses the following conventions for reader alerts:

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 material not covered in the manual.

Timesaver Means *the described action saves time*. You can save time by performing the action described in the paragraph.

Documentation Organization

This section describes the Cisco IOS documentation set, how it is organized, and how to access it on Cisco.com. It also lists the configuration guides, command references, and supplementary references and resources that comprise the documentation set. It contains the following topics:

Cisco IOS Documentation Set

The Cisco IOS documentation set consists of the following:

- Release notes and caveats provide information about platform, technology, and feature support for a release and describe severity 1 (catastrophic), severity 2 (severe), and select severity 3 (moderate) defects in released Cisco IOS software. Review release notes before other documents to learn whether updates have been made to a feature.
- Sets of configuration guides and command references organized by technology and published for each standard Cisco IOS release.
  - Configuration guides—Compilations of documents that provide conceptual and task-oriented descriptions of Cisco IOS features.
  - Command references—Compilations of command pages in alphabetical order that provide detailed information about the commands used in the Cisco IOS features and the processes that comprise the related configuration guides. For each technology, there is a single command reference that supports all Cisco IOS releases and that is updated at each standard release.
- Lists of all the commands in a specific release and all commands that are new, modified, removed, or replaced in the release.
- Command reference book for **debug** commands. Command pages are listed in alphabetical order.
- Reference book for system messages for all Cisco IOS releases.

Cisco IOS Documentation on Cisco.com

The following sections describe the organization of the Cisco IOS documentation set and how to access various document types.

Use Cisco Feature Navigator to find information about platform support and Cisco IOS and Catalyst OS software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

New Features List

The New Features List for each release provides a list of all features in the release with hyperlinks to the feature guides in which they are documented.

Feature Guides

Cisco IOS features are documented in feature guides. Feature guides describe one feature or a group of related features that are supported on many different software releases and platforms. Your Cisco IOS software release or platform may not support all the features documented in a feature guide. See the Feature Information table at the end of the feature guide for information about which features in that guide are supported in your software release.

Configuration Guides

Configuration guides are provided by technology and release and comprise a set of individual feature guides relevant to the release and technology.

Command References

Command reference books contain descriptions of Cisco IOS commands that are supported in many different software releases and on many different platforms. The books are organized by technology. For information about all Cisco IOS commands, use the Command Lookup Tool at http://tools.cisco.com/Support/CLILookup or the *Cisco IOS Master Command List, All Releases*, at http://www.cisco.com/en/US/docs/ios/mcl/allreleasemcl/all_book.html.

Cisco IOS Supplementary Documents and Resources

Supplementary documents and resources are listed in Table 2 on page xi.

Configuration Guides, Command References, and Supplementary Resources

Table 1 lists, in alphabetical order, Cisco IOS software configuration guides and command references, including brief descriptions of the contents of the documents. The Cisco IOS command references contain commands for Cisco IOS software for all releases. The configuration guides and command references support many different software releases and platforms. Your Cisco IOS software release or platform may not support all these technologies.

Table 2 lists documents and resources that supplement the Cisco IOS software configuration guides and command references. These supplementary resources include release notes and caveats; master command lists; new, modified, removed, and replaced command lists; system messages; and the debug command reference.

For additional information about configuring and operating specific networking devices, and to access Cisco IOS documentation, go to the Product/Technologies Support area of Cisco.com at the following location:

http://www.cisco.com/go/techdocs

Table 1 Cisco IOS Configuration Guides and Command References

Configuration Guide and Command Reference Titles	Features/Protocols/Technologies
• *Cisco IOS AppleTalk Configuration Guide* • *Cisco IOS AppleTalk Command Reference*	AppleTalk protocol.
• *Cisco IOS Asynchronous Transfer Mode Configuration Guide* • *Cisco IOS Asynchronous Transfer Mode Command Reference*	LAN ATM, multiprotocol over ATM (MPoA), and WAN ATM.

Table 1 Cisco IOS Configuration Guides and Command References (continued)

Configuration Guide and Command Reference Titles	Features/Protocols/Technologies
• *Cisco IOS Bridging and IBM Networking Configuration Guide* • *Cisco IOS Bridging Command Reference* • *Cisco IOS IBM Networking Command Reference*	Transparent and source-route transparent (SRT) bridging, source-route bridging (SRB), Token Ring Inter-Switch Link (TRISL), and token ring route switch module (TRRSM). Data-link switching plus (DLSw+), serial tunnel (STUN), block serial tunnel (BSTUN); logical link control, type 2 (LLC2), synchronous data link control (SDLC); IBM Network Media Translation, including Synchronous Data Logical Link Control (SDLLC) and qualified LLC (QLLC); downstream physical unit (DSPU), Systems Network Architecture (SNA) service point, SNA frame relay access, advanced peer-to-peer networking (APPN), native client interface architecture (NCIA) client/server topologies, and IBM Channel Attach.
• *Cisco IOS Broadband Access Aggregation and DSL Configuration Guide* • *Cisco IOS Broadband Access Aggregation and DSL Command Reference*	PPP over ATM (PPPoA) and PPP over Ethernet (PPPoE).
• *Cisco IOS Carrier Ethernet Configuration Guide* • *Cisco IOS Carrier Ethernet Command Reference*	Operations, Administration, and Maintenance (OAM); Ethernet connectivity fault management (CFM); ITU-T Y.1731 fault management functions; Ethernet Local Management Interface (ELMI); MAC address support on service instances, bridge domains, and pseudowire; IEEE 802.3ad Link Bundling; Link Aggregation Control Protocol (LACP) support for Ethernet and Gigabit Ethernet links and EtherChannel bundles; LACP support for stateful switchover (SSO), in service software upgrade (ISSU), Cisco nonstop forwarding (NSF), and nonstop routing (NSR) on Gigabit EtherChannel bundles; and Link Layer Discovery Protocol (LLDP) and media endpoint discovery (MED).
• *Cisco IOS Configuration Fundamentals Configuration Guide* • *Cisco IOS Configuration Fundamentals Command Reference*	Autoinstall, Setup, Cisco IOS command-line interface (CLI), Cisco IOS file system (IFS), Cisco IOS web browser user interface (UI), basic file transfer services, and file management.
• *Cisco IOS DECnet Configuration Guide* • *Cisco IOS DECnet Command Reference*	DECnet protocol.
• *Cisco IOS Dial Technologies Configuration Guide* • *Cisco IOS Dial Technologies Command Reference*	Asynchronous communications, dial backup, dialer technology, dial-in terminal services and AppleTalk remote access (ARA), dial-on-demand routing, dial-out, ISDN, large scale dial-out, modem and resource pooling, Multilink PPP (MLP), PPP, and virtual private dialup network (VPDN).
• *Cisco IOS Flexible NetFlow Configuration Guide* • *Cisco IOS Flexible NetFlow Command Reference*	Flexible NetFlow.

Table 1 Cisco IOS Configuration Guides and Command References (continued)

Configuration Guide and Command Reference Titles	Features/Protocols/Technologies
• *Cisco IOS High Availability Configuration Guide* • *Cisco IOS High Availability Command Reference*	A variety of high availability (HA) features and technologies that are available for different network segments (from enterprise access to service provider core) to facilitate creation of end-to-end highly available networks. Cisco IOS HA features and technologies can be categorized in three key areas: system-level resiliency, network-level resiliency, and embedded management for resiliency.
• *Cisco IOS Integrated Session Border Controller Command Reference*	A VoIP-enabled device that is deployed at the edge of networks. An SBC is a toolkit of functions, such as signaling interworking, network hiding, security, and quality of service (QoS).
• *Cisco IOS Intelligent Services Gateway Configuration Guide* • *Cisco IOS Intelligent Services Gateway Command Reference*	Subscriber identification, service and policy determination, session creation, session policy enforcement, session life-cycle management, accounting for access and service usage, and session state monitoring.
• *Cisco IOS Interface and Hardware Component Configuration Guide* • *Cisco IOS Interface and Hardware Component Command Reference*	LAN interfaces, logical interfaces, serial interfaces, virtual interfaces, and interface configuration.
• *Cisco IOS IP Addressing Services Configuration Guide* • *Cisco IOS IP Addressing Services Command Reference*	Address Resolution Protocol (ARP), Network Address Translation (NAT), Domain Name System (DNS), Dynamic Host Configuration Protocol (DHCP), and Next Hop Address Resolution Protocol (NHRP).
• *Cisco IOS IP Application Services Configuration Guide* • *Cisco IOS IP Application Services Command Reference*	Enhanced Object Tracking (EOT), Gateway Load Balancing Protocol (GLBP), Hot Standby Router Protocol (HSRP), IP Services, Server Load Balancing (SLB), Stream Control Transmission Protocol (SCTP), TCP, Web Cache Communication Protocol (WCCP), User Datagram Protocol (UDP), and Virtual Router Redundancy Protocol (VRRP).
• *Cisco IOS IP Mobility Configuration Guide* • *Cisco IOS IP Mobility Command Reference*	Mobile ad hoc networks (MANet) and Cisco mobile networks.
• *Cisco IOS IP Multicast Configuration Guide* • *Cisco IOS IP Multicast Command Reference*	Protocol Independent Multicast (PIM) sparse mode (PIM-SM), bidirectional PIM (bidir-PIM), Source Specific Multicast (SSM), Multicast Source Discovery Protocol (MSDP), Internet Group Management Protocol (IGMP), and Multicast VPN (MVPN).
• *Cisco IOS IP Routing: BFD Configuration Guide*	Bidirectional forwarding detection (BFD).
• *Cisco IOS IP Routing: BGP Configuration Guide* • *Cisco IOS IP Routing: BGP Command Reference*	Border Gateway Protocol (BGP), multiprotocol BGP, multiprotocol BGP extensions for IP multicast.
• *Cisco IOS IP Routing: EIGRP Configuration Guide* • *Cisco IOS IP Routing: EIGRP Command Reference*	Enhanced Interior Gateway Routing Protocol (EIGRP).
• *Cisco IOS IP Routing: ISIS Configuration Guide* • *Cisco IOS IP Routing: ISIS Command Reference*	Intermediate System-to-Intermediate System (IS-IS).

Table 1 Cisco IOS Configuration Guides and Command References (continued)

Configuration Guide and Command Reference Titles	Features/Protocols/Technologies
• *Cisco IOS IP Routing: ODR Configuration Guide* • *Cisco IOS IP Routing: ODR Command Reference*	On-Demand Routing (ODR).
• *Cisco IOS IP Routing: OSPF Configuration Guide* • *Cisco IOS IP Routing: OSPF Command Reference*	Open Shortest Path First (OSPF).
• *Cisco IOS IP Routing: Protocol-Independent Configuration Guide* • *Cisco IOS IP Routing: Protocol-Independent Command Reference*	IP routing protocol-independent features and commands. Generic policy-based routing (PBR) features and commands are included.
• *Cisco IOS IP Routing: RIP Configuration Guide* • *Cisco IOS IP Routing: RIP Command Reference*	Routing Information Protocol (RIP).
• *Cisco IOS IP SLAs Configuration Guide* • *Cisco IOS IP SLAs Command Reference*	Cisco IOS IP Service Level Agreements (IP SLAs).
• *Cisco IOS IP Switching Configuration Guide* • *Cisco IOS IP Switching Command Reference*	Cisco Express Forwarding, fast switching, and Multicast Distributed Switching (MDS).
• *Cisco IOS IPv6 Configuration Guide* • *Cisco IOS IPv6 Command Reference*	For IPv6 features, protocols, and technologies, go to the IPv6 "Start Here" document.
• *Cisco IOS ISO CLNS Configuration Guide* • *Cisco IOS ISO CLNS Command Reference*	ISO Connectionless Network Service (CLNS).
• *Cisco IOS LAN Switching Configuration Guide* • *Cisco IOS LAN Switching Command Reference*	VLANs, Inter-Switch Link (ISL) encapsulation, IEEE 802.10 encapsulation, IEEE 802.1Q encapsulation, and multilayer switching (MLS).
• *Cisco IOS Mobile Wireless Gateway GPRS Support Node Configuration Guide* • *Cisco IOS Mobile Wireless Gateway GPRS Support Node Command Reference*	Cisco IOS Gateway GPRS Support Node (GGSN) in a 2.5-generation general packet radio service (GPRS) and 3-generation universal mobile telecommunication system (UMTS) network.
• *Cisco IOS Mobile Wireless Home Agent Configuration Guide* • *Cisco IOS Mobile Wireless Home Agent Command Reference*	Cisco Mobile Wireless Home Agent, an anchor point for mobile terminals for which mobile IP or proxy mobile IP services are provided.
• *Cisco IOS Mobile Wireless Packet Data Serving Node Configuration Guide* • *Cisco IOS Mobile Wireless Packet Data Serving Node Command Reference*	Cisco Packet Data Serving Node (PDSN), a wireless gateway that is between the mobile infrastructure and standard IP networks and that enables packet data services in a code division multiple access (CDMA) environment.
• *Cisco IOS Mobile Wireless Radio Access Networking Configuration Guide* • *Cisco IOS Mobile Wireless Radio Access Networking Command Reference*	Cisco IOS radio access network products.

Table 1 Cisco IOS Configuration Guides and Command References (continued)

Configuration Guide and Command Reference Titles	Features/Protocols/Technologies
• *Cisco IOS Multiprotocol Label Switching Configuration Guide* • *Cisco IOS Multiprotocol Label Switching Command Reference*	MPLS Label Distribution Protocol (LDP), MPLS Layer 2 VPNs, MPLS Layer 3 VPNs, MPLS traffic engineering (TE), and MPLS Embedded Management (EM) and MIBs.
• *Cisco IOS Multi-Topology Routing Configuration Guide* • *Cisco IOS Multi-Topology Routing Command Reference*	Unicast and multicast topology configurations, traffic classification, routing protocol support, and network management support.
• *Cisco IOS NetFlow Configuration Guide* • *Cisco IOS NetFlow Command Reference*	Network traffic data analysis, aggregation caches, and export features.
• *Cisco IOS Network Management Configuration Guide* • *Cisco IOS Network Management Command Reference*	Basic system management; system monitoring and logging; troubleshooting, logging, and fault management; Cisco Discovery Protocol; Cisco IOS Scripting with Tool Control Language (Tcl); Cisco networking services (CNS); DistributedDirector; Embedded Event Manager (EEM); Embedded Resource Manager (ERM); Embedded Syslog Manager (ESM); HTTP; Remote Monitoring (RMON); SNMP; and VPN Device Manager Client for Cisco IOS software (XSM Configuration).
• *Cisco IOS Novell IPX Configuration Guide* • *Cisco IOS Novell IPX Command Reference*	Novell Internetwork Packet Exchange (IPX) protocol.
• *Cisco IOS Optimized Edge Routing Configuration Guide* • *Cisco IOS Optimized Edge Routing Command Reference*	Optimized edge routing (OER) monitoring; Performance Routing (PfR); and automatic route optimization and load distribution for multiple connections between networks.
• *Cisco IOS Quality of Service Solutions Configuration Guide* • *Cisco IOS Quality of Service Solutions Command Reference*	Traffic queueing, traffic policing, traffic shaping, Modular QoS CLI (MQC), Network-Based Application Recognition (NBAR), Multilink PPP (MLP) for QoS, header compression, AutoQoS, Resource Reservation Protocol (RSVP), and weighted random early detection (WRED).
• *Cisco IOS Security Command Reference*	Access control lists (ACLs); authentication, authorization, and accounting (AAA); firewalls; IP security and encryption; neighbor router authentication; network access security; network data encryption with router authentication; public key infrastructure (PKI); RADIUS; TACACS+; terminal access security; and traffic filters.
• *Cisco IOS Security Configuration Guide: Securing the Data Plane*	Access Control Lists (ACLs); Firewalls: Context-Based Access Control (CBAC) and Zone-Based Firewall; Cisco IOS Intrusion Prevention System (IPS); Flexible Packet Matching; Unicast Reverse Path Forwarding (uRPF); Threat Information Distribution Protocol (TIDP) and TMS.
• *Cisco IOS Security Configuration Guide: Securing the Control Plane*	Control Plane Policing, Neighborhood Router Authentication.

Table 1 Cisco IOS Configuration Guides and Command References (continued)

Configuration Guide and Command Reference Titles	Features/Protocols/Technologies
• *Cisco IOS Security Configuration Guide: Securing User Services*	AAA (includes 802.1x authentication and Network Admission Control [NAC]); Security Server Protocols (RADIUS and TACACS+); Secure Shell (SSH); Secure Access for Networking Devices (includes Autosecure and Role-Based CLI access); Lawful Intercept.
• *Cisco IOS Security Configuration Guide: Secure Connectivity*	Internet Key Exchange (IKE) for IPsec VPNs; IPsec Data Plane features; IPsec Management features; Public Key Infrastructure (PKI); Dynamic Multipoint VPN (DMVPN); Easy VPN; Cisco Group Encrypted Transport VPN (GETVPN); SSL VPN.
• *Cisco IOS Service Advertisement Framework Configuration Guide* • *Cisco IOS Service Advertisement Framework Command Reference*	Cisco Service Advertisement Framework.
• *Cisco IOS Service Selection Gateway Configuration Guide* • *Cisco IOS Service Selection Gateway Command Reference*	Subscriber authentication, service access, and accounting.
• *Cisco IOS Software Activation Configuration Guide* • *Cisco IOS Software Activation Command Reference*	An orchestrated collection of processes and components to activate Cisco IOS software feature sets by obtaining and validating Cisco software licenses.
• *Cisco IOS Software Modularity Installation and Configuration Guide* • *Cisco IOS Software Modularity Command Reference*	Installation and basic configuration of software modularity images, including installations on single and dual route processors, installation rollbacks, software modularity binding, software modularity processes, and patches.
• *Cisco IOS Terminal Services Configuration Guide* • *Cisco IOS Terminal Services Command Reference*	DEC, local-area transport (LAT), and X.25 packet assembler/disassembler (PAD).
• *Cisco IOS Virtual Switch Command Reference*	Virtual switch redundancy, high availability, and packet handling; converting between standalone and virtual switch modes; virtual switch link (VSL); Virtual Switch Link Protocol (VSLP). **Note** For information about virtual switch configuration, see the product-specific software configuration information for the Cisco Catalyst 6500 series switch or for the Metro Ethernet 6500 series switch.
• *Cisco IOS Voice Configuration Library* • *Cisco IOS Voice Command Reference*	Cisco IOS support for voice call control protocols, interoperability, physical and virtual interface management, and troubleshooting. The library includes documentation for IP telephony applications.
• *Cisco IOS VPDN Configuration Guide* • *Cisco IOS VPDN Command Reference*	Layer 2 Tunneling Protocol (L2TP) dial-out load balancing and redundancy; L2TP extended failover; L2TP security VPDN; multihop by Dialed Number Identification Service (DNIS); timer and retry enhancements for L2TP and Layer 2 Forwarding (L2F); RADIUS Attribute 82 (tunnel assignment ID); shell-based authentication of VPDN users; tunnel authentication via RADIUS on tunnel terminator.

Table 1 Cisco IOS Configuration Guides and Command References (continued)

Configuration Guide and Command Reference Titles	Features/Protocols/Technologies
• *Cisco IOS Wide-Area Networking Configuration Guide* • *Cisco IOS Wide-Area Networking Command Reference*	Frame Relay; Layer 2 Tunnel Protocol Version 3 (L2TPv3); L2VPN Pseudowire Redundancy; L2VPN Interworking; Layer 2 Local Switching; Link Access Procedure, Balanced (LAPB); and X.25.
• *Cisco IOS Wireless LAN Configuration Guide* • *Cisco IOS Wireless LAN Command Reference*	Broadcast key rotation, IEEE 802.11x support, IEEE 802.1x authenticator, IEEE 802.1x local authentication service for Extensible Authentication Protocol-Flexible Authentication via Secure Tunneling (EAP-FAST), Multiple Basic Service Set ID (BSSID), Wi-Fi Multimedia (WMM) required elements, and Wi-Fi Protected Access (WPA).

Table 2 lists documents and resources that supplement the Cisco IOS software configuration guides and command references.

Table 2 Cisco IOS Supplementary Documents and Resources

Document Title or Resource	Description
Cisco IOS Master Command List, All Releases	Alphabetical list of all the commands documented in all Cisco IOS releases.
Cisco IOS New, Modified, Removed, and Replaced Commands	List of all the new, modified, removed, and replaced commands for a Cisco IOS release.
Cisco IOS System Message Guide	List of Cisco IOS system messages and descriptions. System messages may indicate problems with your system, may be informational only, or may help diagnose problems with communications lines, internal hardware, or system software.
Cisco IOS Debug Command Reference	Alphabetical list of **debug** commands including brief descriptions of use, command syntax, and usage guidelines.
Release Notes and Caveats	Information about new and changed features, system requirements, and other useful information about specific software releases; information about defects in specific Cisco IOS software releases.
MIBs	Files used for network monitoring. To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator.
RFCs	Standards documents maintained by the Internet Engineering Task Force (IETF) that Cisco IOS documentation references where applicable. The full text of referenced RFCs may be obtained at the following URL: http://www.rfc-editor.org/

Additional Resources and Documentation Feedback

What's New in Cisco Product Documentation is released monthly and describes all new and revised Cisco technical documentation. The *What's New in Cisco Product Documentation* publication also provides information about obtaining the following resources:

- Technical documentation
- Cisco product security overview
- Product alerts and field notices
- Technical assistance

Cisco IOS technical documentation includes embedded feedback forms where you can rate documents and provide suggestions for improvement. Your feedback helps us improve our documentation.

Using the Command-Line Interface in Cisco IOS Software

Last Updated: October 14, 2009

This document provides basic information about the command-line interface (CLI) in Cisco IOS software and how you can use some of the CLI features. This document contains the following sections:

For more information about using the CLI, see the "Using the Cisco IOS Command-Line Interface" section of the *Cisco IOS Configuration Fundamentals Configuration Guide*.

For information about the software documentation set, see the "About Cisco IOS Software Documentation" document.

Initially Configuring a Device

Initially configuring a device varies by platform. For information about performing an initial configuration, see the hardware installation documentation that is provided with the original packaging of the product or go to the Product/Technologies Support area of Cisco.com at http://www.cisco.com/go/techdocs.

After you have performed the initial configuration and connected the device to your network, you can configure the device by using the console port or a remote access method, such as Telnet or Secure Shell (SSH), to access the CLI or by using the configuration method provided on the device, such as Security Device Manager.

Changing the Default Settings for a Console or AUX Port

There are only two changes that you can make to a console port and an AUX port:

- Change the port speed with the **config-register 0x** command. Changing the port speed is not recommended. The well-known default speed is 9600.
- Change the behavior of the port; for example, by adding a password or changing the timeout value.

Note The AUX port on the Route Processor (RP) installed in a Cisco ASR 1000 series router does not serve any useful customer purpose and should be accessed only under the advisement of a customer support representative.

Using the CLI

This section describes the following topics:

Understanding Command Modes

The CLI command mode structure is hierarchical, and each mode supports a set of specific commands. This section describes the most common of the many modes that exist.

Table 1 lists common command modes with associated CLI prompts, access and exit methods, and a brief description of how each mode is used.

Table 1 CLI Command Modes

Command Mode	Access Method	Prompt	Exit Method	Mode Usage
User EXEC	Log in.	`Router>`	Issue the **logout** or **exit** command.	• Change terminal settings. • Perform basic tests. • Display device status.
Privileged EXEC	From user EXEC mode, issue the **enable** command.	`Router#`	Issue the **disable** command or the **exit** command to return to user EXEC mode.	• Issue **show** and **debug** commands. • Copy images to the device. • Reload the device. • Manage device configuration files. • Manage device file systems.
Global configuration	From privileged EXEC mode, issue the **configure terminal** command.	`Router(config)#`	Issue the **exit** command or the **end** command to return to privileged EXEC mode.	Configure the device.
Interface configuration	From global configuration mode, issue the **interface** command.	`Router(config-if)#`	Issue the **exit** command to return to global configuration mode or the **end** command to return to privileged EXEC mode.	Configure individual interfaces.
Line configuration	From global configuration mode, issue the **line vty** or **line console** command.	`Router(config-line)#`	Issue the **exit** command to return to global configuration mode or the **end** command to return to privileged EXEC mode.	Configure individual terminal lines.

Table 1 CLI Command Modes (continued)

Command Mode	Access Method	Prompt	Exit Method	Mode Usage
ROM monitor	From privileged EXEC mode, issue the **reload** command. Press the **Break** key during the first 60 seconds while the system is booting.	`rommon # >` The # symbol represents the line number and increments at each prompt.	Issue the **continue** command.	• Run as the default operating mode when a valid image cannot be loaded. • Access the fall-back procedure for loading an image when the device lacks a valid image and cannot be booted. • Perform password recovery when a Ctrl-Break sequence is issued within 60 seconds of a power-on or reload event.
Diagnostic (available only on Cisco ASR 1000 series routers)	The router boots or enters diagnostic mode in the following scenarios. When a Cisco IOS process or processes fail, in most scenarios the router will reload. • A user-configured access policy was configured using the **transport-map** command, which directed the user into diagnostic mode. • The router was accessed using an RP auxiliary port. • A break signal (**Ctrl-C**, **Ctrl-Shift-6**, or the **send break** command) was entered, and the router was configured to enter diagnostic mode when the break signal was received.	`Router(diag)#`	If a Cisco IOS process failure is the reason for entering diagnostic mode, the failure must be resolved and the router must be rebooted to exit diagnostic mode. If the router is in diagnostic mode because of a transport-map configuration, access the router through another port or use a method that is configured to connect to the Cisco IOS CLI. If the RP auxiliary port was used to access the router, use another port for access. Accessing the router through the auxiliary port is not useful for customer purposes.	• Inspect various states on the router, including the Cisco IOS state. • Replace or roll back the configuration. • Provide methods of restarting the Cisco IOS software or other processes. • Reboot hardware (such as the entire router, an RP, an ESP, a SIP, a SPA) or other hardware components. • Transfer files into or off of the router using remote access methods such as FTP, TFTP, and SCP.

EXEC commands are not saved when the software reboots. Commands that you issue in a configuration mode can be saved to the startup configuration. If you save the running configuration to the startup configuration, these commands will execute when the software is rebooted. Global configuration mode is the highest level of configuration mode. From global configuration mode, you can enter a variety of other configuration modes, including protocol-specific modes.

ROM monitor mode is a separate mode that is used when the software cannot load properly. If a valid software image is not found when the software boots or if the configuration file is corrupted at startup, the software might enter ROM monitor mode. Use the question symbol (?) to view the commands that you can use while the device is in ROM monitor mode.

```
rommon 1 > ?
alias               set and display aliases command
boot                boot up an external process
confreg             configuration register utility
cont                continue executing a downloaded image
context             display the context of a loaded image
cookie              display contents of cookie PROM in hex
.
.
.
rommon 2 >
```

The following example shows how the command prompt changes to indicate a different command mode:

```
Router> enable
Router# configure terminal
Router(config)# interface ethernet 1/1
Router(config-if)# ethernet
Router(config-line)# exit
Router(config)# end
Router#
```

Note A keyboard alternative to the **end** command is Ctrl-Z.

Using the Interactive Help Feature

The CLI includes an interactive Help feature. Table 2 describes the purpose of the CLI interactive Help commands.

Table 2 CLI Interactive Help Commands

Command	Purpose
help	Provides a brief description of the Help feature in any command mode.
?	Lists all commands available for a particular command mode.
*partial command***?**	Provides a list of commands that begin with the character string (no space between the command and the question mark).
partial command<**Tab**>	Completes a partial command name (no space between the command and <Tab>).
command **?**	Lists the keywords, arguments, or both associated with the command (space between the command and the question mark).
command keyword **?**	Lists the arguments that are associated with the keyword (space between the keyword and the question mark).

The following examples show how to use the help commands:

help

```
Router> help

Help may be requested at any point in a command by entering a question mark '?'. If nothing
matches, the help list will be empty and you must backup until entering a '?' shows the
available options.

Two styles of help are provided:

1. Full help is available when you are ready to enter a command argument (e.g. 'show ?')
and describes each possible argument.

2. Partial help is provided when an abbreviated argument is entered and you want to know
what arguments match the input (e.g. 'show pr?'.)
```

?

```
Router# ?
Exec commands:
  access-enable        Create a temporary access-List entry
  access-profile       Apply user-profile to interface
  access-template      Create a temporary access-List entry
  alps                 ALPS exec commands
  archive              manage archive files
<snip>
```

partial command?

```
Router(config)# zo?
zone  zone-pair
```

partial command<Tab>

```
Router(config)# we<Tab> webvpn
```

command ?

```
Router(config-if)# pppoe ?
  enable        Enable pppoe
  max-sessions  Maximum PPPOE sessions
```

command keyword ?

```
Router(config-if)# pppoe enable ?
  group  attach a BBA group
  <cr>
```

Understanding Command Syntax

Command syntax is the format in which a command should be entered in the CLI. Commands include the name of the command, keywords, and arguments. Keywords are alphanumeric strings that are used literally. Arguments are placeholders for values that a user must supply. Keywords and arguments may be required or optional.

Specific conventions convey information about syntax and command elements. Table 3 describes these conventions.

Table 3 CLI Syntax Conventions

Symbol/Text	Function	Notes
< > (angle brackets)	Indicate that the option is an argument.	Sometimes arguments are displayed without angle brackets.
A.B.C.D.	Indicates that you must enter a dotted decimal IP address.	Angle brackets (< >) are not always used to indicate that an IP address is an argument.
WORD (all capital letters)	Indicates that you must enter one word.	Angle brackets (< >) are not always used to indicate that a WORD is an argument.
LINE (all capital letters)	Indicates that you must enter more than one word.	Angle brackets (< >) are not always used to indicate that a LINE is an argument.
<cr> (carriage return)	Indicates the end of the list of available keywords and arguments, and also indicates when keywords and arguments are optional. When <cr> is the only option, you have reached the end of the branch or the end of the command if the command has only one branch.	—

The following examples show syntax conventions:

```
Router(config)# ethernet cfm domain ?
  WORD  domain name
Router(config)# ethernet cfm domain dname ?
  level
Router(config)# ethernet cfm domain dname level ?
  <0-7>  maintenance level number
Router(config)# ethernet cfm domain dname level 7 ?
  <cr>

Router(config)# snmp-server file-transfer access-group 10 ?
  protocol  protocol options
  <cr>

Router(config)# logging host ?
  Hostname or A.B.C.D  IP address of the syslog server
  ipv6                 Configure IPv6 syslog server
```

Understanding Enable and Enable Secret Passwords

Some privileged EXEC commands are used for actions that impact the system, and it is recommended that you set a password for these commands to prevent unauthorized use. Two types of passwords, enable (not encrypted) and enable secret (encrypted), can be set. The following commands set these passwords and are issued in global configuration mode:

- **enable** *password*
- **enable secret** *password*

Using an enable secret password is recommended because it is encrypted and more secure than the enable password. When you use an enable secret password, text is encrypted (unreadable) before it is written to the config.text file. When you use an enable password, the text is written as entered (readable) to the config.text file.

Each type of password is case sensitive, can contain from 1 to 25 uppercase and lowercase alphanumeric characters, and can start with a numeral. Spaces are also valid password characters; for example, "two words" is a valid password. Leading spaces are ignored, but trailing spaces are recognized.

Note Both password commands have numeric keywords that are single integer values. If you choose a numeral for the first character of your password followed by a space, the system will read the number as if it were the numeric keyword and not as part of your password.

When both passwords are set, the enable secret password takes precedence over the enable password.

To remove a password, use the **no** form of the commands: **no enable** *password* or **no enable secret** *password*.

For more information about password recovery procedures for Cisco products, see http://www.cisco.com/en/US/products/sw/iosswrel/ps1831/products_tech_note09186a00801746e6.shtml.

Using the Command History Feature

The command history feature saves, in a command history buffer, the commands that you enter during a session. The default number of saved commands is 10, but the number is configurable within the range of 0 to 256. This command history feature is particularly useful for recalling long or complex commands.

To change the number of commands saved in the history buffer for a terminal session, issue the **terminal history size** command:

```
Router# terminal history size num
```

A command history buffer is also available in line configuration mode with the same default and configuration options. To set the command history buffer size for a terminal session in line configuration mode, issue the **history** command:

```
Router(config-line)# history [size num]
```

To recall commands from the history buffer, use the following methods:

- Press Ctrl-P or the Up Arrow key—Recalls commands beginning with the most recent command. Repeat the key sequence to recall successively older commands.
- Press Ctrl-N or the Down Arrow key—Recalls the most recent commands in the history buffer after they have been recalled using Ctrl-P or the Up Arrow key. Repeat the key sequence to recall successively more recent commands.

  Note The arrow keys function only on ANSI-compatible terminals such as the VT100.

- Issue the **show history** command in user EXEC or privileged EXEC mode—Lists the most recent commands that you entered. The number of commands that are displayed is determined by the setting of the **terminal history size** and **history** commands.

The command history feature is enabled by default. To disable this feature for a terminal session, issue the **terminal no history** command in user EXEC or privileged EXEC mode or the **no history** command in line configuration mode.

Abbreviating Commands

Typing a complete command name is not always required for the command to execute. The CLI recognizes an abbreviated command when the abbreviation contains enough characters to uniquely identify the command. For example, the **show version** command can be abbreviated as **sh ver**. It cannot be abbreviated as **s ver** because **s** could mean **show**, **set**, or **systat**. The **sh v** abbreviation also is not valid because the **show** command has **vrrp** as a keyword in addition to **version**. (Command and keyword examples are from Cisco IOS Release 12.4(13)T.)

Using Aliases for CLI Commands

To save time and the repetition of entering the same command multiple times, you can use a command alias. An alias can be configured to do anything that can be done at the command line, but an alias cannot move between modes, type in passwords, or perform any interactive functions.

Table 4 shows the default command aliases.

Table 4 Default Command Aliases

Command Alias	Original Command
h	help
lo	logout
p	ping
s	show
u or **un**	undebug
w	where

To create a command alias, issue the **alias** command in global configuration mode. The syntax of the command is **alias** *mode command-alias original-command*. Following are some examples:

- Router(config)# **alias exec prt partition**—privileged EXEC mode
- Router(config)# **alias configure sb source-bridge**—global configuration mode
- Router(config)# **alias interface rl rate-limit**—interface configuration mode

To view both default and user-created aliases, issue the **show alias** command.

For more information about the **alias** command, see http://www.cisco.com/en/US/docs/ios/fundamentals/command/reference/cf_a1.html.

Using the no and default Forms of Commands

Most configuration commands have a **no** form that is used to reset a command to its default value or disable a feature or function. For example, the **ip routing** command is enabled by default. To disable this command, you would issue the **no ip routing** command. To re-enable IP routing, you would issue the **ip routing** command.

Configuration commands may also have a **default** form, which returns the command settings to their default values. For commands that are disabled by default, using the **default** form has the same effect as using the **no** form of the command. For commands that are enabled by default and have default settings, the **default** form enables the command and returns the settings to their default values.

The **no** form is documented in the command pages of command references. The **default** form is generally documented in the command pages only when the **default** form performs a different function than the plain and **no** forms of the command. To see what **default** commands are available on your system, enter **default ?** in the appropriate command mode.

Using the debug Command

A **debug** command produces extensive output that helps you troubleshoot problems in your network. These commands are available for many features and functions within Cisco IOS software. Some **debug** commands are **debug all**, **debug aaa accounting**, and **debug mpls packets**. To use **debug** commands during a Telnet session with a device, you must first enter the **terminal monitor** command. To turn off debugging completely, you must enter the **undebug all** command.

For more information about **debug** commands, see the *Cisco IOS Debug Command Reference* at http://www.cisco.com/en/US/docs/ios/debug/command/reference/db_book.html.

Caution Debugging is a high priority and high CPU utilization process that can render your device unusable. Use **debug** commands only to troubleshoot specific problems. The best times to run debugging are during periods of low network traffic and when few users are interacting with the network. Debugging during these periods decreases the likelihood that the **debug** command processing overhead will affect network performance or user access or response times.

Filtering Output Using Output Modifiers

Many commands produce lengthy output that may use several screens to display. Using output modifiers, you can filter this output to show only the information that you want to see.

The following three output modifiers are available:

- **begin** *regular-expression*—Displays the first line in which a match of the regular expression is found and all lines that follow.
- **include** *regular-expression*—Displays all lines in which a match of the regular expression is found.
- **exclude** *regular-expression*—Displays all lines except those in which a match of the regular expression is found.

To use one of these output modifiers, type the command followed by the pipe symbol (|), the modifier, and the regular expression that you want to search for or filter. A regular expression is a case-sensitive alphanumeric pattern. It can be a single character or number, a phrase, or a more complex string.

The following example illustrates how to filter output of the **show interface** command to display only lines that include the expression "protocol."

```
Router# show interface | include protocol

FastEthernet0/0 is up, line protocol is up
Serial4/0 is up, line protocol is up
Serial4/1 is up, line protocol is up
Serial4/2 is administratively down, line protocol is down
Serial4/3 is administratively down, line protocol is down
```

Understanding CLI Error Messages

You may encounter some error messages while using the CLI. Table 5 shows the common CLI error messages.

Table 5 Common CLI Error Messages

Error Message	Meaning	How to Get Help
% Ambiguous command: "show con"	You did not enter enough characters for the command to be recognized.	Reenter the command followed by a space and a question mark (?). The keywords that you are allowed to enter for the command appear.
% Incomplete command.	You did not enter all the keywords or values required by the command.	Reenter the command followed by a space and a question mark (?). The keywords that you are allowed to enter for the command appear.
% Invalid input detected at "^" marker.	You entered the command incorrectly. The caret (^) marks the point of the error.	Enter a question mark (?) to display all the commands that are available in this command mode. The keywords that you are allowed to enter for the command appear.

For more system error messages, see the following document:

- *Cisco IOS Release 12.4T System Message Guide*

Saving Changes to a Configuration

To save changes that you made to the configuration of a device, you must issue the **copy running-config startup-config** command or the **copy system:running-config nvram:startup-config** command. When you issue these commands, the configuration changes that you made are saved to the startup configuration and saved when the software reloads or power to the device is turned off or interrupted. The following example shows the syntax of the **copy running-config startup-config** command:

```
Router# copy running-config startup-config
Destination filename [startup-config]?
```

You press Enter to accept the startup-config filename (the default), or type a new filename and then press Enter to accept that name. The following output is displayed indicating that the configuration was saved.

```
Building configuration...
[OK]
Router#
```

On most platforms, the configuration is saved to NVRAM. On platforms with a Class A flash file system, the configuration is saved to the location specified by the CONFIG_FILE environment variable. The CONFIG_FILE variable defaults to NVRAM.

Additional Information

- "Using the Cisco IOS Command-Line Interface" section of the *Cisco IOS Configuration Fundamentals Configuration Guide*

  http://www.cisco.com/en/US/docs/ios/fundamentals/configuration/guide/cf_cli-basics.html
- Cisco Product/Technology Support

  http://www.cisco.com/go/techdocs
- Support area on Cisco.com (also search for documentation by task or product)

  http://www.cisco.com/en/US/support/index.html
- Software Download Center (downloads; tools; licensing, registration, advisory, and general information) (requires Cisco.com user ID and password)

  http://www.cisco.com/kobayashi/sw-center/
- Error Message Decoder, a tool to help you research and resolve error messages for Cisco IOS software

  http://www.cisco.com/pcgi-bin/Support/Errordecoder/index.cgi
- Command Lookup Tool, a tool to help you find detailed descriptions of Cisco IOS commands (requires Cisco.com user ID and password)

  http://tools.cisco.com/Support/CLILookup
- Output Interpreter, a troubleshooting tool that analyzes command output of supported **show** commands

  https://www.cisco.com/pcgi-bin/Support/OutputInterpreter/home.pl

---

Basic System Management

Performing Basic System Management

This chapter describes the basic tasks that you can perform to manage the general system features of the Cisco IOS software—those features that are generally not specific to a particular protocol.

This document applies to Cisco IOS Release 12.2.

For a complete description of the basic system management commands in this chapter, refer to the "Basic System Management Commands" chapter in the "Cisco IOS System Management Commands" part of the Release 12.2 *Cisco IOS Configuration Fundamentals Command Reference*. To locate documentation of other commands that appear in this chapter, use the *Cisco IOS Command Reference Master Index* or search online.

To identify hardware or software image support for a specific feature, use Feature Navigator on Cisco.com or refer to the software release notes for a specific release. For more information, see the "Identifying Platform Support for Cisco IOS Software Features" section in the "About Cisco IOS Software Documentation" chapter.

Basic System Management Task List

To customize the general functionality of your system, perform any of the tasks in the following sections. All tasks in this chapter are optional, though some, such as setting time and calendar services, are highly recommended.

- Configuring the System Name (Recommended)
- Customizing the CLI Prompt
- Creating and Displaying Command Aliases
- Controlling Minor Services (Recommended)
- Hiding Telnet Addresses
- Setting Time and Calendar Services (Recommended)
- Delaying EXEC Startup
- Handling an Idle Telnet Connection
- Setting the Interval for Load Data
- Limiting the Number of TCP Transactions

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

- Configuring Switching and Scheduling Priorities
- Modifying the System Buffer Size

See the end of this chapter for the "Basic System Management Examples" section.

Configuring the System Name

The most basic system management task is to assign a name to your system (router, access server, switch, and so on). The system name, also called the host name, is used to uniquely identify the system in your network. The system name is displayed at the CLI prompt. If no name is configured, the system default name is `Router`. To configure a name for your device, use the following command in global configuration mode:

Command	Purpose
`Router(config)# hostname name`	Sets the host name.

For an example of configuring a system name, see the section "System Configuration File Example" at the end of this chapter.

Customizing the CLI Prompt

By default, the CLI prompt consists of the system name followed by an angle bracket (>) for EXEC mode or a pound sign (#) for privileged EXEC mode. To customize the CLI prompt for your system, use either of the following commands in global configuration mode, as needed:

Command	Purpose
`Router(config)# prompt string`	Customizes the CLI prompt.
`Router(config)# no service prompt config`	Disables the display of the CLI prompt.

Creating and Displaying Command Aliases

Command aliases allow you to configure alternative syntax for commands. You may want to create aliases for commonly used or complex commands. For example, you could assign the alias **save config** to the **copy running-config startup-config** command to reduce the amount of typing you have to perform, or if your users might find a **save config** command easier to remember. Use word substitutions or abbreviations to tailor command syntax for you and your user community.

To create a command alias, use the following command in global configuration mode:

Command	Purpose
`Router(config)# alias mode alias-name alias-command-line`	Configures a command alias.

To display a list of command aliases currently configured on your system, and the original command syntax for those aliases, use the following command in EXEC mode:

Command	Purpose
`Router# show aliases [mode]`	Displays all command aliases and original command syntax, or displays the aliases for only a specified command mode.

Keep in mind that any aliases you configure will only be effective on your system, and that the original command syntax will appear in the configuration file.

Controlling Minor Services

The minor services are "small servers" that run on your routing device and are useful for basic system testing and for providing basic network functions. Minor services are useful for testing connections from another host on the network.

Cisco small servers are conceptually equivalent to daemons.

Small servers provided by Cisco IOS software-based devices include TCP, UDP, HTTP, BOOTP, and Finger. For information about the HTTP server, see the "Using the Cisco Web Browser User Interface" chapter in this book.

The TCP small server provides the following minor services:

- Echo—Echoes back whatever you type. To test this service, issue the **telnet** *a.b.c.d* **echo** command from a remote host.
- Chargen—Generates a stream of ASCII data. To test this service, issue the **telnet** *a.b.c.d* **chargen** command from a remote host.
- Discard—Discards whatever you type. To test this service, issue the **telnet** *a.b.c.d* **discard** command from a remote host.
- Daytime—Returns system date and time if you have configured NTP or have set the date and time manually. To test this service, issue the **telnet** *a.b.c.d* **daytime** command from a remote host.

The User Datagram Protocol (UDP) small server provides the following minor services:

- Echo—Echoes the payload of the datagram you send.
- Chargen—Discards the datagram you send and responds with a 72 character string of ASCII characters terminated with a CR+LF (carriage return and line feed).
- Discard—Silently discards the datagram you send.

To enable TCP or UDP services, use the following commands in global configuration mode, as needed:

Command	Purpose
`Router(config)# service tcp-small-servers`	Enables the minor TCP services echo, chargen, discard, and daytime.
`Router(config)# service udp-small-servers`	Enables the minor UDP services echo, chargen, and discard.

Because the minor services can be misused, these commands are disabled by default.

Caution Enabling minor services creates the potential for certain types of denial-of-service attacks, such as the UDP diagnostic port attack. Therefore, any network device that has UDP, TCP, BOOTP, or Finger services should be protected by a firewall or have the services disabled. For information on preventing UDP diagnostic port attacks, see the white paper titled *Defining Strategies to Protect Against UDP Diagnostic Port Denial of Service Attacks*, available on Cisco.com.

Note that the **no** form of the **service tcp-small-servers** and **service udp-small-servers** commands will appear in the configuration file to inform you when these basic services are disabled.

Controlling the BOOTP Server

You can enable or disable an async line Bootstrap Protocol (BOOTP) service on your routing device. This small server is enabled by default. Due to security considerations, this service should be disabled if you are not using it. To disable the BOOTP server on your platform, use the following command in global configuration mode:

Command	Purpose
`Router(config)# no ip bootp server`	Disables the BOOTP server.

Because Dynamic Host Configuration Protocol (DHCP) is based on the Bootstrap Protocol, both of these service share the "well-known" UDP server port of 67 (per the internet standards and RFCs). For more information about DHCP configuration in Cisco IOS software, see the *Cisco IOS IP Configuration Guide*. For more information about BOOTP, see RFC 951. Interoperation between BOOTP and DHCP is defined in RFC 1534. DHCP is defined in RFC 2131.

Controlling the Finger Protocol

The Finger protocol allows users throughout the network to get a list of the users currently using a particular routing device. The information displayed includes the processes running on the system, the line number, connection name, idle time, and terminal location. This information is provided through the Cisco IOS software **show users** EXEC command.

To enable a Cisco device to respond to Finger (port 79) requests, use the following command in global configuration mode:

Command	Purpose
`Router(config)# ip finger`	Enables the Finger protocol service, which allows the system to respond to finger requests.

To configure the finger protocol to be compliant with RFC 1288, use the following command in global configuration mode:

Command	Purpose
`Router(config)# ip finger rfc-compliant`	Configures the device to wait for "Return" or "/W" input when processing Finger requests.

The **rfc-compliant** form of this command should not be configured for devices with more than 20 simultaneous users (see caveat CSCds92731 on Cisco.com for details). The difference between the two forms of this command is as follows: when the **ip finger** command is configured, the router will respond to a **telnet** *a.b.c.d* **finger** command from a remote host by immediately displaying the output of the **show users** command and then closing the connection. When the **ip finger rfc-compliant** command is configured, the router will wait for input before displaying anything. The remote user can then press the Return key to display the output of the **show users** command, or enter **/W** to display the output of the **show users wide** command. After this information is displayed, the connection is closed.

Hiding Telnet Addresses

You can hide addresses while attempting to establish a Telnet session. To configure the router to suppress Telnet addresses, use the following command in global configuration mode:

Command	Purpose
`Router(config)# service hide-telnet-address`	Hides addresses while establishing a Telnet session.

The hide feature suppresses the display of the address and continues to display all other messages that normally would be displayed during a connection attempt, such as detailed error messages if the connection failed.

Use the **busy-message** line configuration command with the **service hide-telnet-address** command to customize the information displayed during Telnet connection attempts. If the connection attempt fails, the router suppresses the address and displays the message specified with the **busy-message** command.

Setting Time and Calendar Services

All Cisco routers provide an array of time-of-day services. These services allow the products to accurately keep track of the current time and date, to synchronize multiple devices to the same time, and to provide time services to other systems. The following sections describe the concepts and task associated with time and calendar services:

- Understanding Time Sources
- Configuring NTP
- Configuring SNTP
- Configuring VINES Time Service
- Configuring Time and Date Manually
- Using the Hardware Clock
- Monitoring Time and Calendar Services
- Configuring Time Ranges

Understanding Time Sources

Most Cisco routers have two clocks: a battery-powered hardware clock (referenced in CLI commands as the "calendar") and a software clock (referenced in CLI commands as the "clock"). These two clocks are managed separately.

The primary source for time data on your system is the software clock. This clock runs from the moment the system starts up and keeps track of the current date and time. The software clock can be set from a number of sources and in turn can be used to distribute the current time through various mechanisms to other systems. When a router with a hardware clock is initialized or rebooted, the software clock is initially set based on the time in the hardware clock. The software clock can then be updated from the following sources:

- Network Time Protocol (NTP)
- Simple Network Time Protocol (SNTP)
- VINES Time Service
- Manual configuration (using the hardware clock)

Because the software clock can be dynamically updated it has the potential to be more accurate than the hardware clock.

The software clock can provide time to the following services:

- Access lists
- NTP
- VINES time service
- User **show** commands
- Logging and debugging messages
- The hardware clock

Note The software clock cannot provide time to the NTP or VINES Time Service if it was set using SNTP.

The software clock keeps track of time internally based on Coordinated Universal Time (UTC), also known as Greenwich Mean Time (GMT). You can configure information about the local time zone and summer time (daylight savings time) so that the time is displayed correctly relative to the local time zone.

The software clock keeps track of whether the time is "authoritative" (that is, whether it has been set by a time source considered to be authoritative). If it is not authoritative, the time will be available only for display purposes and will not be redistributed.

Network Time Protocol

The Network Time Protocol (NTP) is a protocol designed to time-synchronize a network of machines. NTP runs over UDP, which in turn runs over IP. NTP Version 3 is documented in RFC 1305.

An NTP network usually gets its time from an authoritative time source, such as a radio clock or an atomic clock attached to a time server. NTP then distributes this time across the network. NTP is extremely efficient; no more than one packet per minute is necessary to synchronize two machines to the accuracy of within a millisecond of one another.

NTP uses the concept of a "stratum" to describe how many NTP "hops" away a machine is from an authoritative time source. A "stratum 1" time server typically has an authoritative time source (such as a radio or atomic clock, or a GPS time source) directly attached, a "stratum 2" time server receives its time via NTP from a "stratum 1" time server, and so on.

NTP avoids synchronizing to a machine whose time may not be accurate in two ways. First, NTP will never synchronize to a machine that is not in turn synchronized itself. Second, NTP will compare the time reported by several machines, and will not synchronize to a machine whose time is significantly different than the others, even if its stratum is lower. This strategy effectively builds a self-organizing tree of NTP servers.

The Cisco implementation of NTP does not support stratum 1 service; in other words, it is not possible to connect to a radio or atomic clock (for some specific platforms, however, you can connect a GPS time-source device). We recommend that time service for your network be derived from the public NTP servers available in the IP internet.

If the network is isolated from the internet, the Cisco implementation of NTP allows a machine to be configured so that it acts as though it is synchronized via NTP, when in fact it has determined the time using other means. Other machines can then synchronize to that machine via NTP.

A number of manufacturers include NTP software for their host systems, and a publicly available version for systems running UNIX and its various derivatives is also available. This software also allows UNIX-derivative servers to acquire the time directly from an atomic clock which would subsequently propagate time information along to Cisco routers.

The communications between machines running NTP (known as "associations") are usually statically configured; each machine is given the IP address of all machines with which it should form associations. Accurate timekeeping is made possible by exchanging NTP messages between each pair of machines with an association.

However, in a LAN environment, NTP can be configured to use IP broadcast messages instead. This alternative reduces configuration complexity because each machine can simply be configured to send or receive broadcast messages. However, the accuracy of timekeeping is marginally reduced because the information flow is one-way only.

The time kept on a machine is a critical resource, so we strongly recommend that you use the security features of NTP to avoid the accidental or malicious setting of incorrect time. Two mechanisms are available: an access list-based restriction scheme and an encrypted authentication mechanism.

When multiple sources of time (VINES, hardware clock, manual configuration) are available, NTP is always considered to be more authoritative. NTP time overrides the time set by any other method.

Simple Network Time Protocol

Simple Network Time Protocol (SNTP) is a simplified, client-only version of NTP for use on Cisco 1003, Cisco 1004, Cisco 1005, Cisco 1600, Cisco 1720, and Cisco 1750 routers. SNTP can receive only the time from NTP servers; it cannot be used to provide time services to other systems.

SNTP typically provides time within 100 milliseconds of the accurate time, but it does not provide the complex filtering and statistical mechanisms of NTP. In addition, SNTP does not authenticate traffic, although you can configure extended access lists to provide some protection. An SNTP client is more vulnerable to misbehaving servers than an NTP client and should be used only in situations where strong authentication is not required.

You can configure SNTP to request and accept packets from configured servers or to accept NTP broadcast packets from any source. When multiple sources are sending NTP packets, the server with the best stratum is selected. (See the "Network Time Protocol" section for a description of strata.) If multiple servers are at the same stratum, a configured server is preferred over a broadcast server. If multiple

servers pass both tests, the first one to send a time packet is selected. SNTP will choose a new server only if it stops receiving packets from the currently selected server, or if a better server (according to the above criteria) is discovered.

VINES Time Service

Time service is available when Banyan VINES is configured. This protocol is a standard part of VINES. The Cisco implementation allows the VINES time service to be used in two ways. First, if the system has learned the time from some other source, it can act as a VINES time server and provide time to other machines running VINES. Second, it can use the VINES time service to set the software clock if no other form of time service is available.

Note Support for Banyan VINES and XNS is removed from Cisco IOS software in Cisco IOS Release 12.2(13)T and later.

Hardware Clock

Some routers contain a battery-powered hardware clock that tracks the date and time across system restarts and power outages. The hardware clock is always used to initialize the software clock when the system is restarted.

Note Within the CLI command syntax, the hardware clock is referred to as the "system calendar."

If no other source is available, the hardware clock can be considered to be an authoritative source of time and be redistributed via NTP or VINES time service. If NTP is running, the hardware clock can be updated periodically from NTP, compensating for the inherent drift in the hardware clock.

Configuring NTP

NTP services are disabled on all interfaces by default. The following sections contain optional tasks that you can perform on your networking device:

- Configuring Poll-Based NTP Associations
- Configuring Broadcast-Based NTP Associations
- Configuring an NTP Access Group
- Configuring NTP Authentication
- Disabling NTP Services on a Specific Interface
- Configuring the Source IP Address for NTP Packets
- Configuring the System as an Authoritative NTP Server
- Updating the Hardware Clock
- Configuring an External Reference Clock

Configuring Poll-Based NTP Associations

Networking devices running NTP can be configured to operate in variety of association modes when synchronizing time with reference time sources. There are two ways that a networking device can obtain time information on a network: by polling host servers and by listening to NTP broadcasts. In this section, we will focus on the poll-based association modes. Broadcast-based NTP associations will be discussed in the next section.

The following are two most commonly used, poll-based association modes:

- Client mode
- Symmetric active mode

The *client* and the *symmetric active* modes should be used when NTP is required to provide a high level of time accuracy and reliability.

When a networking device is operating in the *client mode*, it polls its assigned time serving hosts for the current time. The networking device will then pick a host from all the polled time servers to synchronize with. Since the relationship that is established in this case is a client-host relationship, the host will not capture or use any time information sent by the local client device. This mode is most suited for file-server and workstation clients that are not required to provide any form of time synchronization to other local clients. Use the **ntp server** command to individually specify the time serving hosts that you want your networking device to consider synchronizing with and to set your networking device to operate in the *client mode*.

When a networking device is operating in the *symmetric active mode*, it polls its assigned time serving hosts for the current time and it responds to polls by its hosts. Since this is a peer-to-peer relationship, the host will also retain time-related information about the local networking device that it is communicating with. This mode should be used when there is a number of mutually redundant servers that are interconnected via diverse network paths. Most Stratum 1 and stratum 2 servers on the Internet today adopt this form of network setup. Use the **ntp peer** command to individually specify the time serving hosts that you want your networking device to consider synchronizing with and to set your networking device to operate in the *symmetric active mode*.

The specific mode that you should set each of your networking devices to depends primarily on the role that you want it to assume as a timekeeping device (server or client) and its proximity to a stratum 1 timekeeping server.

A networking device engages in polling when it is operating as a client or a host in the *client mode* or when it is acting as a peer in the *symmetric active mode*. Although polling does not usually exact a toll on memory and CPU resources such as bandwidth, an exceedingly large number of ongoing and simultaneous polls on a system can seriously impact the performance of a system or slow the performance of a given network. To avoid having an excessive number of ongoing polls on a network, you should limit the number of direct, peer-to-peer or client-to-server associations. Instead, you should consider using NTP broadcasts to propagate time information within a localized network.

Command	Purpose
`Router(config)# ntp peer ip-address [normal-sync] [version number] [key keyid] [source interface] [prefer]`	Forms a peer association with another system.
`Router(config)# ntp server ip-address [version number] [key keyid] [source interface] [prefer]`	Forms a server association with another system.

Note that only one end of an association needs to be configured; the other system will automatically establish the association.

Caution The **ntp clock-period** command is automatically generated to reflect the constantly changing *correction factor* when the **copy running-configuration startup-configuration** command is entered to save the configuration to NVRAM. Do not attempt to manually use the **ntp clock-period** command. Ensure that you remove this command line when copying configuration files to other devices.

For an example of configuring an NTP server-peer relationship, see the "Clock, Calendar, and NTP Configuration Examples" section at the end of this chapter.

Configuring Broadcast-Based NTP Associations

Broadcast-based NTP associations should be used when time accuracy and reliability requirements are modest and if your network is localized and has a large number of clients (more than 20). Broadcast-based NTP associations is also recommended for use on networks that have limited bandwidth, system memory, or CPU resources.

When a networking device is operating in the *broadcastclient mode*, it does not engage in any polling. Instead, it listens for NTP broadcast packets transmitted by broadcast time servers. Consequently, time accuracy can be marginally reduced since time information flows only one way.

Use the **ntp broadcast client** command to set your networking device to listen for NTP broadcast packets propagated through a network. In order for *broadcastclient mode* to work, the broadcast server and its clients must be located on the same subnet. The time server that is transmitting NTP broadcast packets will also have to be enabled on the interface of the given device using the **ntp broadcast** command.

To configure an interface to send NTP broadcasts, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# ntp broadcast [version number]`	Configures the specified interface to send NTP broadcast packets.

To configure an interface to receive NTP broadcasts, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# ntp broadcast client`	Configures the specified interface to receive NTP broadcast packets.

To manually set the estimated round-trip delay between the device and the NTP broadcast server, use the following command in global configuration mode:

Command	Purpose
`Router(config)# ntp broadcastdelay microseconds`	Adjusts the estimated round-trip delay for NTP broadcasts.

Caution The **ntp clock-period** command is automatically generated to reflect the constantly changing *correction factor* when the **copy running-configuration startup-configuration** command is entered to save the configuration to NVRAM. Do not attempt to manually use the **ntp clock-period** command. Ensure that you remove this command line when copying configuration files to other devices.

For an example of configuring broadcast-based NTP associations, see the "Clock, Calendar, and NTP Configuration Examples" section at the end of this chapter.

Configuring an NTP Access Group

The access list-based restriction scheme allows you to grant or deny certain access privileges to an entire network, a subnet within a network, or a host within a subnet. To define an NTP access group, use the following command in global configuration mode:

Command	Purpose
`Router(config)# ntp access-group {query-only | serve-only | serve | peer} access-list-number`	Creates an access group and applies a basic IP access list to it.

The access group options are scanned in the following order, from least restrictive to most restrictive:

1. **peer**—Allows time requests and NTP control queries and allows the system to synchronize itself to a system whose address passes the access list criteria.
2. **serve**—Allows time requests and NTP control queries, but does not allow the system to synchronize itself to a system whose address passes the access list criteria.
3. **serve-only**—Allows only time requests from a system whose address passes the access list criteria.
4. **query-only**—Allows only NTP control queries from a system whose address passes the access list criteria.

If the source IP address matches the access lists for more than one access type, the first type is granted. If no access groups are specified, all access types are granted to all systems. If any access groups are specified, only the specified access types will be granted.

For details on NTP control queries, see RFC 1305 (NTP version 3).

Configuring NTP Authentication

The encrypted NTP authentication scheme should be used when a reliable form of access control is required. Unlike the access list-based restriction scheme which is based on IP addresses, the encrypted authentication scheme uses authentication keys and an authentication process to determine if NTP synchronization packets sent by designated peers or servers on a local network are deemed as trusted before the time information that it carries along with it, is accepted.

The authentication process begins from the moment an NTP packet is created. Cryptographic checksum keys are generated using the MD5 Message Digest Algorithm and are embedded into the NTP synchronization packet that is sent to a receiving client. Once a packet is received by a client, its cryptographic checksum key is decrypted and checked against a list of trusted keys. If the packet contains a matching authenticator key, the timestamp information that is contained within it is accepted by the receiving client. NTP synchronization packets that do not contain a matching authenticator key will be ignored.

It is important to note that the encryption and decryption processes used in NTP authentication can be very CPU-intensive and can seriously degrade the accuracy of the time that is propagated within a network. If your network setup permits a more comprehensive model of access control, you should consider the use of the access list-based form of control instead.

After NTP authentication is properly configured, your networking device will only synchronize with and provide synchronization to trusted time sources. To enable your networking device to send and receive encrypted synchronization packets, use the following commands in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# ntp authenticate`	Enables the NTP authentication feature.
Step 2	`Router(config)# ntp authentication-key` *key-number* `md5` *value*	Defines the authentication keys. Each key has a key number, a type, and a value. Currently the only key type supported is **md5**.
Step 3	`Router(config)# ntp trusted-key` *key-number*	Defines trusted authentication keys. If a key is trusted, this system will be ready to synchronize to a system that uses this key in its NTP packets.
Step 4	`Router (config)# ntp server 172.16.0.6 key` *key-number*	Allows the software clock to be synchronized by a Network Time Protocol (NTP) time server.

Note In Cisco IOS software versions previous to release 12.0, the cryptotype value is displayed along with the ntp authentication key md5 value when the **show running-configuration** command is entered. Avoid copying and pasting the string cryptotype value that is displayed with the authentication-key as it will result in authentication failure.

Disabling NTP Services on a Specific Interface

NTP services are disabled on all interfaces by default.

NTP is enabled globally when any NTP commands are entered. you can selectively prevent NTP packets from being received through a specific interface by using the following command in interface configuration mode to turn off NTP on a given interface:

Command	Purpose
`Router(config-if)# ntp disable`	Disables NTP services on a specific interface.

Configuring the Source IP Address for NTP Packets

When the system sends an NTP packet, the source IP address is normally set to the address of the interface through which the NTP packet is sent. Use the following command in global configuration mode if you want to configure a specific interface from which the IP source address will be taken:

Command	Purpose
`Router(config)# ntp source interface`	Configures an interface from which the IP source address will be taken.

This interface will be used for the source address for all packets sent to all destinations. If a source address is to be used for a specific association, use the **source** parameter on the **ntp peer** or **ntp server** command shown earlier in this chapter.

Configuring the System as an Authoritative NTP Server

Use the following command in global configuration mode if you want the system to be an authoritative NTP server, even if the system is not synchronized to an outside time source:

Command	Purpose
`Router(config)# ntp master [stratum]`	Makes the system an authoritative NTP server.

Note Use the **ntp master** command with caution. It is very easy to override valid time sources using this command, especially if a low stratum number is configured. Configuring multiple machines in the same network with the **ntp master** command can cause instability in timekeeping if the machines do not agree on the time.

For an example of configuring an authoritative NTP server, see the "Clock, Calendar, and NTP Configuration Examples" section at the end of this chapter.

Updating the Hardware Clock

On devices that have hardware clocks (system calendars), you can configure the hardware clock to be periodically updated from the software clock. This is advisable for any device using NTP, because the time and date on the software clock (set using NTP) will be more accurate than the hardware clock, because the time setting on the hardware clock has the potential to drift slightly over time.

Use the following command in global configuration mode if a routing device is synchronized to an outside time source via NTP and you want the hardware clock to be synchronized to NTP time:

Command	Purpose
`Router(config)# ntp update-calendar`	Configures the system to update its hardware clock from the software clock at periodic intervals.

For an example of configuring NTP to update the calendar, see the section "Clock, Calendar, and NTP Configuration Examples" at the end of this chapter.

Configuring an External Reference Clock

Because Cisco's implementation of NTP does not support stratum 1 service, it is not possible to connect to a radio or atomic clock (for some specific platforms however, you can connect a GPS timesource device). However, certain Cisco devices allow you to connect a external GPS-based time-source device for the purposes of distributing a time signal to your network using NTP.

For example, the Trimble Palisade NTP Synchronization Kit can be connected to the auxiliary port of a Cisco 7200 Series router. Also, selected platforms support the use of GPS clocks from Symmetricom (formerly Telecom-Solutions). The refclock (reference clock) drivers provided on these platforms provides the ability to receive an RTS time-stamp signal on the auxiliary port of your routing device.

To configure a Trimble Palisade GPS product connected to the auxiliary port of a Cisco 7200 series router as the NTP reference clock, use the following commands, beginning in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# line aux 0`	Enters line configuration mode for the auxiliary port 0.
Step 2	`Router(config-line)# ntp refclock trimble pps none stratum 1`	Enables the driver that allows the Trimble Palisade NTP Synchronization Kit to be used as the NTP reference clock source (Cisco 7200 series routers only).

To configure a Symmetricom GPS product connected to the auxiliary port of a supported router or switch as the NTP reference clock, use the following commands beginning in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# line aux 0`	Enters line configuration mode for the auxiliary port zero.
Step 2	`Router(config-line)# ntp refclock telecom-solutions pps cts stratum 1`	Enables the driver that allows the Symmetricom GPS product to be used as the NTP reference clock source.

To configure a PPS signal as the source for NTP synchronization, use the following form of the **ntp refclock** command in line configuration mode:

Command	Purpose
`Router(config-line)# ntp refclock pps {cts | ri} [inverted] [pps-offset number] [stratum number] [timestamp-offset number]`	Configures a PPS signal as the source for NTP synchronization.

Verifying the Status of the External Reference Clock

To verify the status of NTP components, use the following commands in privileged EXEC mode:

Command	Purpose
`Router# show ntp associations`	Displays the status of NTP associations, including the status of the GPS reference clock.

Command	Purpose
`Router# show ntp status`	Displays the status of NTP.
`Router# debug ntp refclock`	Allows advanced monitoring of reference clock activities for the purposes of debugging.

Configuring SNTP

SNTP generally is supported on those platforms that do not provide support for NTP, such as the Cisco 1000 series, 1600 series, and 1700 series platforms. SNTP is disabled by default. In order to enable SNTP, use one or both of the following commands in global configuration mode:

Command	Purpose
`Router(config)# sntp server {address | hostname} [version number]`	Configures SNTP to request NTP packets from an NTP server.
`Router(config)# sntp broadcast client`	Configures SNTP to accept NTP packets from any NTP broadcast server.

Enter the **sntp server** command once for each NTP server. The NTP servers must be configured to respond to the SNTP messages from the router.

If you enter both the **sntp server** command and the **sntp broadcast client** command, the router will accept time from a broadcast server but prefer time from a configured server, assuming that the strata are equal. To display information about SNTP, use the **show sntp** EXEC command.

Configuring VINES Time Service

Note Support for Banyan VINES and XNS has been removed from Cisco IOS software, beginning in Cisco IOS Release 12.2(13)T. The following VINES commands are not available in releases derived from 12.2(13)T, such as the 12.3 mainline release.

To distribute the system time and date to other devices on the network using VINES time services, use the following command in global configuration mode:

Command	Purpose
`Router(config)# vines time use-system`	Distributes the system software clock time to other VINES systems.

To set the system time and date from received VINES time services, use the following command in global configuration mode:

Command	Purpose
`Router(config)# vines time set-system`	Sets the software clock system time from received VINES time services.

Configuring Time and Date Manually

If no other source of time is available, you can manually configure the current time and date after the system is restarted. The time will remain accurate until the next system restart. We recommend that you use manual configuration only as a last resort.

To set up time services, complete the tasks in the following sections as needed. If you have an outside source to which the router can synchronize, you do not need to manually set the software clock.

- Configuring the Time Zone
- Configuring Summer Time (Daylight Savings Time)
- Manually Setting the Software Clock
- Using the Hardware Clock

Configuring the Time Zone

To manually configure the time zone used by the Cisco IOS software, use the following command in global configuration mode:

Command	Purpose
`Router(config)# clock timezone zone hours-offset [minutes-offset]`	Sets the time zone. The *zone* argument is the name of the time zone (typically a standard acronym). The *hours-offset* argument is the number of hours the time zone is different from UTC. The *minutes-offset* argument is the number of minutes the time zone is different from UTC.

Tip The *minutes-offset* argument of the **clock timezone** command is available for those cases where a local time zone is a percentage of an hour different from UTC/GMT. For example, the time zone for some sections of Atlantic Canada (AST) is UTC -3.5. In this case, the necessary command would be **clock timezone AST -3 30**.

For an example of configuring the time zone, see the section "Clock, Calendar, and NTP Configuration Examples" at the end of this chapter.

Configuring Summer Time (Daylight Savings Time)

To configure summer time (daylight savings time) in areas where it starts and ends on a particular day of the week each year, use the following command in global configuration mode:

Command	Purpose
`Router(config)# clock summer-time zone recurring [week day month hh:mm week day month hh:mm [offset]]`	Configures a recurring summer time start and end date. The *offset* argument is used to indicate the number of minutes to add to the clock during summer time.

If summer time in your area does not follow this pattern, you can configure the exact date and time of the next summer time event by using one of the following commands in global configuration mode:

Command	Purpose

`Router(config)# clock summer-time zone date month date year hh:mm month date year hh:mm [offset]`
or
`Router(config)# clock summer-time zone date date month year hh:mm date month year hh:mm [offset]`
	Configures a specific summer time start and end date. The *offset* argument is used to indicate the number of minutes to add to the clock during summer time.

For an example of configuring summer time, see the section "Clock, Calendar, and NTP Configuration Examples" at the end of this chapter.

Manually Setting the Software Clock

Generally, if the system is synchronized by a valid outside timing mechanism, such as an NTP or VINES clock source, or if you have a router with a hardware clock, you need not set the software clock. Use this command if no other time sources are available. The time specified in this command is relative to the configured time zone. To set the software clock manually, use the following command in privileged EXEC mode:

Command	Purpose

`Router# clock set hh:mm:ss date month year`
or
`Router# clock set hh:mm:ss month date year`
	Sets the software clock.

Using the Hardware Clock

Most Cisco devices have a separate hardware-based clock in addition to the software-based clock. The hardware clock is a chip with a rechargeable backup battery that can retain the time and date information across reboots of the device.

To maintain the most accurate time update from an authoritative time source on the network, the software clock should receive time updates from an authoritative time on the network. The hardware clock should in turn be updated at regular intervals from the software clock while the system is running.

To customize the use of the hardware clock on your system, perform any of the following optional tasks:

- Setting the Hardware Clock
- Configuring the Router as a Network Time Source
- Setting the Software Clock from the Hardware Clock
- Setting the Hardware Clock from the Software Clock

Setting the Hardware Clock

The hardware clock (system calendar) maintains time separately from the software clock. The hardware clock continues to run when the system is restarted or when the power is turned off. Typically, the hardware clock needs to be manually set only once, when the system is first installed.

You should avoid setting the hardware clock manually if you have access to a reliable external time source. Time synchronization should instead be established using NTP.

If you do not have access to an external time source, use one of the forms of the following command in EXEC mode to set the hardware clock:

Command	Purpose

`Router> calendar set hh:mm:ss day month year`
or
`Router> calendar set hh:mm:ss month day year`
	Sets the hardware clock manually.

Configuring the Router as a Network Time Source

By default, the time maintained on the software clock is not considered to be authoritative and will not be redistributed with NTP or VINES Time Service. To classify the hardware clock as authoritative, use the following command in global configuration mode:

Command	Purpose
`Router(config)# clock calendar-valid`	Enables the router to act as a valid time source to which network peers can synchronize.

For an example of making the hardware clock authoritative, see the "Clock, Calendar, and NTP Configuration Examples" section at the end of this chapter.

Setting the Software Clock from the Hardware Clock

To set the software clock to the new hardware clock setting, use the following command in EXEC mode:

Command	Purpose
`Router# clock read-calendar`	Sets the software clock from the hardware clock.

Setting the Hardware Clock from the Software Clock

To update the hardware clock with a new software clock setting, use the following command in EXEC mode:

Command	Purpose
`Router# clock update-calendar`	Sets the hardware clock from the software clock.

Monitoring Time and Calendar Services

To monitor clock, calendar, and NTP EXEC services, use the following commands in EXEC mode, as needed:

Command	Purpose
`Router# show calendar`	Displays the current hardware clock time.
`Router# show clock [detail]`	Displays the current software clock time.
`Router# show ntp associations [detail]`	Displays the status of NTP associations.
`Router# show ntp status`	Displays the status of NTP.
`Router# show sntp`	Displays information about SNTP (Cisco 1003, Cisco 1004, Cisco 1005, Cisco 1600, Cisco 1720, or Cisco 1750 routers only).

Configuring Time Ranges

Cisco IOS allows implementation of features based on the time of day. The **time-range** global configuration command defines specific times of the day and week, which then can be referenced by a function, so that those time restrictions are imposed on the function itself.

In Cisco IOS Release 12.2, IP and IPX extended access lists are the only functions that can use time ranges. The time range allows the network administrator to define when the **permit** or **deny** statements in the access list are in effect. Prior to the introduction of this feature, access list statements were always in effect once they were applied. Both named or numbered access lists can reference a time range.

Benefits of time ranges include the following:

- The network administrator has more control over permitting or denying a user access to resources. These resources could be an application (identified by an IP address/mask pair and a port number), policy routing, or an on-demand link (identified as interesting traffic to the dialer).
- Network administrators can set a time-based security policy, including the following:
  - Perimeter security using the Cisco IOS Firewall feature set or access lists
  - Data confidentiality with Cisco Encryption Technology or IPSec
- Policy-based routing and queueing functions are enhanced.
- When provider access rates vary by time of day, it is possible to automatically reroute traffic cost effectively.
- Service providers can dynamically change a committed access rate (CAR) configuration to support the quality of service (QoS) service level agreements (SLAs) that are negotiated for certain times of day.
- Network administrators can control logging messages. Access list entries can log traffic at certain times of the day, but not constantly. Therefore, administrators can simply deny access without needing to analyze many logs generated during peak hours.

Defining a Time Range

Note The time range relies on the system's software clock. For the time range feature to work the way you intend, you need a reliable clock source. We recommend that you use NTP to synchronize the system's software clock.

To define a time range, use the following commands beginning in global configuration mode.

	Command	Purpose
Step 1	`Router(config)# time-range` *time-range-name*	Assigns a name to the time range to be configured and enters time-range configuration mode.

Step 2	`Router(config-time-range)# absolute [start` *time date*`] [end` *time date*`]`
	or
	`Router(config-time-range)# periodic` *days-of-the-week hh:mm* `to [`*days-of-the-week*`] hh:mm`
		Specifies when the time range will be in effect. Use some combination of these commands; multiple **periodic** statements are allowed; only one **absolute** statement is allowed.

Repeat these tasks if you have multiple items you want in effect at different times. For example, repeat the steps to include multiple **permit** or **deny** statements in an access list in effect at different times. For more information about these commands, refer to the "Basic System Management Commands" chapter in the "Cisco IOS System Management Commands" part of the Release 12.2 *Cisco IOS Configuration Fundamentals Command Reference*.

Referencing the Time Range

In order for a time range to be applied, you must reference it by name in a feature that can implement time ranges. You can reference the time range in the following Cisco IOS software features:

- IP Extended Access Lists
  - Refer to the "Configuring IP Services" chapter of the Release 12.2 *Cisco IOS IP Configuration Guide* for instructions on creating an IP Extended Access List and referencing a time range.
- IPX Extended Access Lists
  - Refer to the "Configuring Novell IPX" chapter of the Release 12.2 *Cisco IOS AppleTalk and Novell IPX Configuration Guide* for instructions on creating an IPX Extended Access List and referencing a time range.

Delaying EXEC Startup

To delay the startup of the EXEC process on noisy lines until the line has been idle for 3 seconds, use the following command in global configuration mode:

Command	Purpose
`Router(config)# service exec-wait`	Delays startup of the EXEC.

This command is useful on noisy modem lines or when a modem attached to the line is configured to ignore MNP or V.42 negotiations, and MNP or V.42 modems may be dialing in. In these cases, noise or MNP/V.42 packets might be interpreted as usernames and passwords, causing authentication failure before the user can type a username or password. The command is not useful on nonmodem lines or lines without some kind of login configured.

Handling an Idle Telnet Connection

To configure the Cisco IOS software to set the TCP window to zero (0) when the Telnet connection is idle, use the following command in global configuration mode:

Command	Purpose
`Router(config)# service telnet-zero-idle`	Sets the TCP window to zero when the Telnet connection is idle.

Normally, data sent to noncurrent Telnet connections is accepted and discarded. When **service telnet-zero-idle** is enabled, if a session is suspended (that is, some other connection is made active), the TCP window is set to zero. This action prevents the remote host from sending any more data until the connection is resumed. Use this command when it is important that all messages sent by the host be seen by the users and the users are likely to use multiple sessions. Do not use this command if your host will eventually time out and log out a TCP user whose window is zero.

Setting the Interval for Load Data

You can change the period of time over which a set of data is used for computing load statistics. Decisions, such as for dial backup, depend on these statistics. If you decrease the load interval, the average statistics are computed over a shorter period of time and are more responsive to bursts of traffic.

To change the length of time for which a set of data is used to compute load statistics, use the following command in interface configuration mode:

Command	Purpose
`Router(config-if)# load-interval` *seconds*	Sets the length of time for which data is used for load calculations.

Limiting the Number of TCP Transactions

When using a standard TCP implementation to send keystrokes between machines, TCP tends to send one packet for each keystroke typed, which can use up bandwidth and contribute to congestion on larger networks.

John Nagle's algorithm (RFC 896) helps alleviate the small-packet problem in TCP. The first character typed after connection establishment is sent in a single packet, but TCP holds any additional characters typed until the receiver acknowledges the previous packet. Then the second, larger packet is sent, and

additional typed characters are saved until the acknowledgment comes back. The effect is to accumulate characters into larger chunks, and pace their transmission to the network at a rate matching the round-trip time of the given connection. This method is usually preferable for all TCP-based traffic.

By default, the Nagle algorithm is not enabled. To enable the Nagle algorithm and thereby reduce the number of TCP transactions, use the following command in global configuration mode:

Command	Purpose
`Router(config)# service nagle`	Enables the Nagle slow packet avoidance algorithm.

Configuring Switching and Scheduling Priorities

The normal operation of the network server allows the switching operations to use as much of the central processor as is required. If the network is running unusually heavy loads that do not allow the processor the time to handle the routing protocols, you may need to give priority to the system process scheduler. To do so, use the following command in global configuration mode:

Command	Purpose
`Router(config)# scheduler interval` *milliseconds*	Defines the maximum amount of time that can elapse without running the lowest-priority system processes.

To change the amount of time that the CPU spends on fast-switching and process-level operations on the Cisco 7200 series and Cisco 7500 series routers, use the following command in global configuration mode:

Command	Purpose
`Router(config)# scheduler allocate` *network-microseconds process-microseconds*	For the Cisco 7200 series and Cisco 7500 series routers, changes the default time the CPU spends on process tasks and fast switching.

Caution We recommend that you do not change the default values of the **scheduler allocate** command.

To configure the characteristics for a looping process, use the following command in global configuration mode:

Command	Purpose
`Router(config)# scheduler process-watchdog {hang | normal | reload | terminate}`	Configures an action for a looping process.

Modifying the System Buffer Size

You can adjust initial buffer pool settings and the limits at which temporary buffers are created and destroyed. To do so, use the following commands in global configuration mode, as needed:

Command	Purpose
`Router(config)# buffers {small | middle | big | verybig | large | huge | type number} {permanent | max-free | min-free | initial} number`	Adjusts the system buffer sizes.
`Router(config)# buffers huge size number`	Dynamically resizes all huge buffers to the value that you supply.

Caution Normally you need not adjust these parameters; do so only after consulting with technical support personnel. Improper settings can adversely impact system performance.

During normal system operation, there are two sets of buffer pools: public and interface. They behave as follows:

- The buffers in the public pools grow and shrink based upon demand. Some public pools are temporary and are created and destroyed as needed. Other public pools are permanently allocated and cannot be destroyed. Public buffer pools are labeled as small, middle, big, large, very big, and huge.
- Interface pools are static—that is, they are all permanent. One interface pool exists for each interface. For example, a Cisco 4000 1E 4T configuration has one Ethernet buffer pool and four serial buffer pools. In the **buffers** EXEC command, the *type* and *number* arguments allow the user to tune the interface pools.

See the section "Buffer Modification Examples" at the end of this chapter for more information.

The server has one pool of queueing elements and six public pools of packet buffers of different sizes. For each pool, the server keeps count of the number of buffers outstanding, the number of buffers in the free list, and the maximum number of buffers allowed in the free list. To display statistics about the buffer pool on the system, use the following commands in EXEC mode, as needed:

Command	Purpose
`Router> show buffers`	Displays all public pool information.
`Router> show buffers address hex-addr`	Displays buffer information for an address.
`Router> show buffers all [dump | header | packet]`	Displays all public and interface pool information.
`Router> show buffers assigned [dump | header | packet]`	Displays a listing of all buffers in use.
`Router> show buffers failures [dump | header | packet]`	Displays buffer allocation failures.
`Router> show buffers free [dump | header | packet]`	Displays buffers available for use.
`Router> show buffers old [dump | header | packet]`	Displays buffers older than one minute.
`Router> show buffers input-interface interface-type identifier`	Displays buffer information for an input interface.
`Router> show buffers pool pool name`	Displays all interface pool information.

Basic System Management Examples

This section provides the following system management examples:

System Configuration File Example

The following is an example of a typical system configuration file:

```
! Define line password
line 0 4
 password secret
 login
!
! Define privileged-level password
enable-password Secret Word
!
! Define a system hostname
hostname TIP
! Specify a configuration file to load at system startup
boot host host1-confg 192.168.1.111
boot host host2-confg 192.168.1.111
! Specify the system image to boot at startup
boot system sys1-system 192.168.13.111
boot system sys2-system 192.168.1.111
boot system rom
!
! Enable SNMP
snmp-server community red
snmp-server enable traps snmp authentication
snmp-server host 192.168.1.27 public
snmp-server host 192.168.1.111 public
snmp-server host 192.168.2.63 public
!
! Define TACACS server hosts
tacacs-server host 192.168.1.27
tacacs-server host 192.168.13.33
tacacs-server host 192.168.1.33
!
! Define a message-of-the-day banner
banner motd ^C
The Information Place welcomes you

Please call 1-800-555-2222 for a login account, or enter
your password at the prompt.
^C
```

Clock, Calendar, and NTP Configuration Examples

In the following example, a router with a hardware clock has server associations with two other systems, sends broadcast NTP packets, periodically updates the hardware clock, and redistributes time into VINES:

```
clock timezone PST -8
clock summer-time PDT recurring
ntp update-calendar
ntp server 192.168.13.57
ntp server 192.168.11.58
interface Ethernet 0/0
```

```
 ntp broadcast
vines time use-system
```

In the following example, a router with a hardware clock has no outside time source, so it uses the hardware clock as an authoritative time source and distributes the time via NTP broadcast packets:

```
clock timezone MET 2
clock calendar-valid
ntp master
interface fddi 0/0
 ntp broadcast
```

Buffer Modification Examples

The following example instructs the system to keep at least 50 small buffers free:

```
Router> buffers small min-free 50
```

The following example instructs the system to keep no more than 200 middle buffers free:

```
Router> buffers middle max-free 200
```

The following example instructs the system to create one large temporary extra buffer, just after a reload:

```
Router> buffers large initial 1
```

The following example instructs the system to create one permanent huge buffer:

```
Router> buffers huge permanent 1
```

System Monitoring and Logging

Configuring CPU Threshold Notifications

The CPU Thresholding Notification feature notifies users when a predefined threshold of CPU usage is crossed by generating a Simple Network Management Protocol (SNMP) trap message for the top users of the CPU.

Feature History for the CPU Thresholding Notification Feature

Release	Modification
12.0(26)S	This feature was introduced.
12.3(4)T	This feature was integrated into Cisco IOS Release 12.3(4)T.
12.2(25)S	This feature was integrated into Cisco IOS Release 12.2(25)S.

Finding Support Information for Platforms and Cisco IOS Software Images

Use Cisco Feature Navigator to find information about platform support and Cisco IOS software image support. Access Cisco Feature Navigator at http://www.cisco.com/go/fn. You must have an account on Cisco.com. If you do not have an account or have forgotten your username or password, click **Cancel** at the login dialog box and follow the instructions that appear.

Contents

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

Restrictions for CPU Thresholding Notification

CPU utilization averages are computed by Cisco IOS software using a 4-millisecond Network-to-Management Interface (NMI) tick. In the unlikely event where the traffic rate is a multiple of this tick rate over a prolonged period of time, the CPU Thresholding Notification feature may not accurately measure the CPU load.

Information About CPU Thresholding Notification

The CPU Thresholding Notification feature allows you to configure CPU utilization thresholds that, when crossed, trigger a notification. Two types of CPU utilization threshold are supported:

- Rising Threshold, page 2
- Falling Threshold, page 2

Rising Threshold

A rising CPU utilization threshold specifies the percentage of CPU resources that, when exceeded for a configured period of time, triggers a CPU threshold notification.

Falling Threshold

A falling CPU utilization threshold specifies the percentage of CPU resources that, when CPU usage falls below this level for a configured period of time, triggers a CPU threshold notification.

How to Configure CPU Thresholding Notification

This section contains the following procedures:

- Enabling CPU Thresholding Notification, page 2
- Defining CPU Thresholding Notification, page 3
- Setting the Entry Limit and Size of CPU Utilization Statistics, page 4

Enabling CPU Thresholding Notification

To specify the recipient of SNMP notification operations and enable CPU thresholding notification, perform these steps:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp-server enable traps cpu threshold**

4. **snmp-server host** *host-address* [**traps | informs**] [**version** {**1 | 2c | 3 [auth | noauth | priv]**}] *community-string* [**udp-port** *port*] **cpu** [*notification-type*] [**vrf** *vrf-name*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enables global configuration mode.
Step 3	**snmp-server enable traps cpu threshold** **Example:** `Router(config)# snmp-server enable traps cpu threshold`	Enables CPU thresholding violation notification as traps and inform requests.
Step 4	**snmp-server host** *host-address* [**traps** | **informs**] [**version** {**1** | **2c** | **3** [**auth** | **noauth** | **priv**]}] *community-string* [**udp-port** *port*] **cpu** [*notification-type*] [**vrf** *vrf-name*] **Example:** `Router(config)# snmp-server host 192.168.0.0 traps public cpu`	Sends CPU traps to the specified address.

Defining CPU Thresholding Notification

To define a rising and a falling CPU threshold notification, perform these steps:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **process cpu threshold type** {**total | process | interrupt**} **rising** *percentage* **interval** *seconds* [**falling** *percentage* **interval** *seconds*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**process cpu threshold type** {**total** | **process** | **interrupt**} **rising** *percentage* **interval** *seconds* [**falling** *percentage* **interval** *seconds*] **Example:** `Router(config)# process cpu threshold type total rising 80 interval 5 falling 20 interval 5`	Sets the CPU thresholding notifications types and values. • In this example, the CPU utilization threshold is set to 80 percent for a rising threshold notification and 20 percent for a falling threshold notification, with a 5-second polling interval.

Setting the Entry Limit and Size of CPU Utilization Statistics

To set the process entry limit and the size of the history table for CPU utilization statistics, perform these steps:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **process cpu statistics limit entry-percentage** *number* [**size** *seconds*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**process cpu statistics limit entry-percentage** *number* [**size** *seconds*] **Example:** `Router(config)# process cpu statistics limit entry-percentage 40 size 300`	Sets the process entry limit and the size of the history table for CPU utilization statistics. • In this example, to generate an entry in the history table, a process must exceed 40 percent CPU utilization. • In this example, the duration of time for which the most recent history is saved in the history table is 300 seconds.

Configuration Examples for CPU Thresholding Notification

The following examples show how to set a rising and a falling CPU thresholding notification:

- Setting a Rising CPU Thresholding Notification: Example, page 5
- Setting a Falling CPU Thresholding Notification: Example, page 5

Setting a Rising CPU Thresholding Notification: Example

The following example shows how to set a rising CPU thresholding notification for total CPU utilization. When total CPU utilization exceeds 80 percent for a period of 5 seconds or longer, a rising threshold notification is sent.

```
Router(config)# process cpu threshold type total rising 80 interval 5
```

Note When the optional **falling** arguments (*percentage* and *seconds*) are not specified, they take on the same values as the **rising** arguments (*percentage* and *seconds*).

Setting a Falling CPU Thresholding Notification: Example

The following example shows how to set a falling CPU thresholding notification for total CPU utilization. When total CPU utilization, which at one point had risen above 80 percent and triggered a rising threshold notification, falls below 70 percent for a period of 5 seconds or longer, a falling threshold notification is sent.

```
Router(config)# process cpu threshold type total rising 80 interval 5 falling 70
interval 5
```

Note When the optional **falling** arguments (*percentage* and *seconds*) are not specified, they take on the same values as the **rising** arguments (*percentage* and *seconds*).

Additional References

For additional information related to the CPU Thresholding Notification feature, refer to the following references:

Related Documents

Related Topic	Document Title
SNMP traps	Configuration Fundamentals Command Reference

Standards

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

MIBs

MIBs	MIBs Link
CISCO-PROCESS-MIB	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

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

Technical Assistance

Description	Link
Technical Assistance Center (TAC) home page, containing 30,000 pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/public/support/tac/home.shtml

Command Reference

The following new and modified commands are pertinent to this feature. To see the command pages for these commands and other commands used with this feature, go to the *Cisco IOS Master Commands List*, Release 12.4, at http://www.cisco.com/en/US/docs/ios/mcl/124mainlinemcl/124_book.html.

- **process cpu statistics limit entry-percentage**
- **process cpu threshold type**
- **snmp-server enable traps cpu**
- **snmp-server host**

Defining Memory Threshold Notifications

First Published: August 21, 2003
Last Updated: May 2, 2008

The Memory Threshold Notifications feature allows you to reserve memory for critical notifications and to configure a router to issue notifications when available memory falls below a specified threshold.

Feature History for the Memory Threshold Notifications Feature

Release	Modification
12.2(18)S	This feature was introduced.
12.0(26)S	This feature was integrated into Cisco IOS Release 12.0(26) S.
12.3(4)T	This feature was integrated into Cisco IOS Release 12.3(4)T.

Finding Support Information for Platforms and Cisco IOS Software Images

Use Cisco Feature Navigator to find information about platform support and Cisco IOS software image support. Access Cisco Feature Navigator at http://www.cisco.com/go/fn. You must have an account on Cisco.com. If you do not have an account or have forgotten your username or password, click **Cancel** at the login dialog box and follow the instructions that appear.

Contents

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

Information About Memory Threshold Notifications

The Memory Threshold Notifications feature provides two ways to mitigate low-memory conditions on a router: notifications can be sent to indicate that free memory has fallen below a configured threshold, and memory can be reserved to ensure that sufficient memory is available to issue critical notifications. To implement the Memory Threshold Notifications feature, you should understand the following concepts:

Memory Threshold Notifications

Notifications are messages issued by the router. When you specify a memory threshold using the **memory free low-watermark** command, for example, the router issues a notification when available free memory falls below the specified threshold, and again once available free memory rises to 5 percent above the specified threshold. The following are examples of memory threshold notifications:

Available Free Memory Less Than the Specified Threshold

```
000029: *Aug 12 22:31:19.559: %SYS-4-FREEMEMLOW: Free Memory has dropped below 2000k
Pool: Processor  Free: 66814056  freemem_lwm: 204800000
```

Available Free Memory Recovered to More Than the Specified Threshold

```
000032: *Aug 12 22:33:29.411: %SYS-5-FREEMEMRECOVER: Free Memory has recovered 2000k
Pool: Processor  Free: 66813960  freemem_lwm: 0
```

Memory Reservation

Memory reservation for critical operations ensures that management processes, such as event logging, continue to function even when router memory is exhausted.

How to Define Memory Threshold Notifications

This section contains the following procedures:

Setting a Low Free Memory Threshold

To set a low free memory threshold, perform the following steps:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **memory free low-watermark** {**processor** *threshold* | **io** *threshold*}

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.

Step 3	**memory free low-watermark processor** *threshold*
	or
	memory free low-watermark io *threshold*
	
	Example:
	`Router(config)# memory free low-watermark processor 20000`
	or
	
	Example:
	`Router(config)# memory free low-watermark io 20000`
		Specifies a threshold in kilobytes of free processor or input/output (I/O) memory. To view acceptable values for the memory threshold, enter the following command:
		• **memory free low-watermark processor ?**
		or
		• **memory free low-watermark io ?**

Reserving Memory for Critical Notifications

When a router is overloaded by processes, the amount of available memory might fall to levels insufficient for it to issue critical notifications. To reserve a region of memory to be used by the router for the issuing of critical notifications, perform the following steps:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **memory reserve critical** *kilobytes*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**memory reserve critical** *kilobytes* **Example:** `Router(config)# memory reserve critical 1000`	Reserves the specified amount of memory in kilobytes so that the router can issue critical notifications. • The amount of memory reserved for critical notifications cannot exceed 25 percent of total available memory.

Configuration Examples for Memory Threshold Notifications

The following examples show how to configure a router to issue notifications when available memory falls below a specified threshold and how to reserve memory for critical notifications:

- Setting a Low Free Memory Threshold: Examples, page 4
- Reserving Memory for Critical Notifications: Example, page 5

Setting a Low Free Memory Threshold: Examples

The following example specifies a threshold of 20000 KB of free processor memory before the router issues notifications:

Threshold for Free Processor Memory

```
Router(config)# memory free low-watermark processor 20000
```

The following example specifies a threshold of 20000 KB of free I/O memory before the router issues notifications:

Threshold for Free IO Memory

```
Router(config)# memory free low-watermark io 20000
```

If available free memory falls below the specified threshold, the router sends a notification message like this one:

```
000029: *Aug 12 22:31:19.559: %SYS-4-FREEMEMLOW: Free Memory has dropped below 20000k
Pool: Processor  Free: 66814056  freemem_lwm: 204800000
```

Once available free memory rises to above 5 percent of the threshold, another notification message like this is sent:

```
000032: *Aug 12 22:33:29.411: %SYS-5-FREEMEMRECOVER: Free Memory has recovered 20000k
Pool: Processor  Free: 66813960  freemem_lwm: 0
```

Reserving Memory for Critical Notifications: Example

The following example reserves 1000 KB of memory for critical notifications:

```
Router# memory reserved critical 1000
```

Note The amount of memory reserved for critical notifications cannot exceed 25 percent of total available memory.

Additional References

The following sections provide references related to the Memory Threshold Notifications feature:

Related Documents

Related Topic	Document Title
Logging system messages	Troubleshooting and Fault Management module

Standards

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

MIBs

MIBs	MIBs Link
No new or modified MIBs are supported by this feature and support for existing MIBs has not been modified by this feature.	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

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

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/techsupport

Command Reference

The following commands are introduced or modified in the feature or features documented in this module. For information about these commands, see the *Cisco IOS Network Management Command Reference* at http://www.cisco.com/en/US/docs/ios/netmgmt/command/reference/nm_book.html. For information about all Cisco IOS commands, go to the Command Lookup Tool at http://tools.cisco.com/Support/CLILookup or to the Cisco IOS Master Commands List.

- **memory free low-watermark**
- **memory reserve critical**

Troubleshooting, Fault Management, and Logging

Troubleshooting and Fault Management

This chapter describes basic tasks that you can perform to troubleshoot your system and the network. For detailed troubleshooting procedures and scenarios, refer to the *Internetwork Troubleshooting Guide*. For complete details on all **debug** commands, refer to the *Cisco IOS Debug Command Reference*.

For a complete description of the troubleshooting commands in this chapter, refer to the "Troubleshooting and Fault Management Commands" chapter in "Cisco IOS System Management Commands" part of the Release 12.2 *Cisco IOS Configuration Fundamentals Command Reference*. To locate documentation of other commands that appear in this chapter, use the *Cisco IOS Command Reference Master Index* or search online.

Troubleshooting and Fault Management Task List

To manage network faults, you need to discover, isolate, and correct problems. You can discover problems with the system monitoring commands, isolate problems with the system test commands, and resolve problems with other commands, including **debug** commands.

To perform general fault management, perform the tasks described in the following sections:

In addition to the material presented in this chapter, many chapters in the Cisco IOS software configuration guides include fault management tasks specific to certain technologies and features. You can find these tasks in the "Monitoring and Maintaining" sections.

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

Displaying System Information Using show Commands

To provide information about system processes, the Cisco IOS software includes an extensive list of EXEC commands that begin with the word **show**, which, when executed, display detailed tables of system information. Following is a partial list of system management **show** commands. To display the information described, use the following commands in EXEC mode, as needed:

Command	Purpose
Router# **show c2600**	Displays information about the Cisco 2600 platform, including interrupts, IOS Priority Masks, and IDMA status, for troubleshooting.
Router# **show c7200**	Displays information about the CPU and midplane for the Cisco 7200 series routers.
Router# **show context**	Displays information stored in NVRAM when the router crashes. This command is only useful to your technical support representative. This command is supported on the Cisco 2600 and 7000 series routers.
Router# **show controllers**	Displays information specific to the hardware on a line card.
Router# **show controllers logging**	Displays logging information about a line card.
Router# **show controllers tech-support**	Displays general information about a line for use when reporting a problem.
Router# **show controllers vip** *slot-number* **tech-support**	Displays information about the Versatile Interface Processor (VIP) card for use when reporting a problem
Router# **show diag**	Displays hardware information (including DRAM and static RAM details) for line cards.
Router# **show environment** [**all** | **last** | **table**]	Displays a message indicating whether an environmental warning condition currently exists, the temperature and voltage information, the last measured value from each of the six test points stored in nonvolatile memory, or environmental specifications. Examples of systems that support this command include the Cisco 7000 and the Cisco 12000 series routers.
Router# **show gsr**	Displays hardware information on the Cisco 12000 series Gigabit Switch Router (GSR).
Router# **show gt64010**	Displays all GT64010 internal registers and interrupt status on the Cisco 7200 series routers.
Router# **show memory** [*memory-type*] [**free**] [**summary**]	Displays memory pool statistics including summary information about the activities of the system memory allocator and a block-by-block listing of memory use.
Router# **show pci** {**hardware** | **bridge** [*register*]}	Displays information about the peripheral component interconnect (PCI) hardware registers or bridge registers for the Cisco 2600 and 7000 series routers.
Router# **show processes** [**cpu**]	Displays information about all active processes.
Router# **show processes memory**	Displays information about memory usage.
Router# **show protocols**	Displays the configured protocols.

Command	Purpose
Router# **show stacks**	Displays stack usage of processes and interrupt routines, including the reason for the last system reboot. This command is only useful to your technical support representative.
Router# **show subsys** [**class** *class* | **name** *name*]	Displays subsystem information.
Router# **show tcp** [*line-number*]	Displays the status of TCP connections.
Router# **show tcp brief** [**all**]	Displays a concise description of TCP connection endpoints.
Router# **show tdm connections** [**motherboard** | **slot** *number*]	Displays a snapshot of the time-division multiplexing (TDM) bus connection or data memory in a Cisco AS5200 access server.
Router# **show tech-support** [**page**] [**password**]	Displays information about the system for use when reporting a problem.

Refer to specific **show** commands in the tables of configuration commands found throughout the chapters in Cisco IOS software configuration guides. Refer to the Cisco IOS software command reference publications for detailed descriptions of the commands.

Testing Network Connectivity

To test basic network connectivity, perform the tasks described in the following sections:

- Configuring the TCP Keepalive Packet Service, page 3
- Testing Connections with the ping Command, page 4
- Tracing Packet Routes, page 4

Configuring the TCP Keepalive Packet Service

The TCP keepalive capability allows a router to detect when the host with which it is communicating experiences a system failure, even if data stops being sent (in either direction). This capability is most useful on incoming connections. For example, if a host failure occurs while the router is communicating with a printer, the router might never notice, because the printer does not generate any traffic in the opposite direction. If keepalives are enabled, they are sent once every minute on otherwise idle connections. If 5 minutes pass and no keepalives are detected, the connection is closed. The connection is also closed if the host replies to a keepalive packet with a reset packet. This will happen if the host crashes and comes back up again.

To generate the TCP keepalive packet service, use the following command in global configuration mode:

Command	Purposes
Router(config)# **service** {**tcp-keepalives-in** | **tcp-keepalives-out**}	Generates TCP keepalive packets on idle network connections, either incoming connections initiated by a remote host, or outgoing connections initiated by a user.

Testing Connections with the ping Command

As an aid to diagnosing basic network connectivity, many network protocols support an echo protocol. The protocol involves sending a special datagram to the destination host, then waiting for a reply datagram from that host. Results from this echo protocol can help in evaluating the path-to-host reliability, delays over the path, and whether the host can be reached or is functioning.

To invoke the echo protocol, use the following command in either user or privileged EXEC mode:

Command	Purposes
`Router# ping [protocol] {host | address}`	Invokes a diagnostic tool for testing connectivity.

Refer to specific **ping** commands in the tables of configuration commands found throughout the chapters in Cisco IOS software configuration guides. Refer to the Cisco IOS software command reference publications for detailed descriptions of the command.

Tracing Packet Routes

To trace the routes that packets will actually take when traveling to their destinations, use the following command in either user or privileged EXEC mode:

Command	Purposes
`Router# trace [protocol] [destination]`	Traces packet routes through the network (privileged level).

Logging System Messages

By default, routers send logging messages (including debug command output) a logging process. The logging process controls the distribution of logging messages to various destinations, such as the logging buffer, terminal lines, or a UNIX syslog server, depending on your configuration. The process also sends messages to the console. When the logging process is on, the messages are displayed on the console after the process that generated them has finished.

When the logging process is disabled, messages are sent only to the console. The messages are sent as they are generated, so error and debug output will be interspersed with prompts or output from the command.

You can set the severity level of the messages to control the type of messages displayed for the console and each destination. You can time-stamp log messages or set the syslog source address to enhance real-time debugging and management.

System logging messages are traditionally referred to as System Error Messages. Refer to the *Cisco IOS Software System Error Messages* publication for detailed information on specific system logging messages.

Enabling System Message Logging

System message logging is enabled by default. It must be enabled in order to send messages to any destination other than the console.

To disable message logging, use the **no logging on** command. Note that disabling the logging process can slow down the router because a process cannot continue until the messages are written to the console.

To reenable message logging after it has been disabled, use the following command in global configuration mode:

Command	Purposes
`Router(config)# logging on`	Enables message logging.

Enabling Message Logging for a Slave Card

To enable slave VIP cards to log status messages to the console (print the messages to the screen), use the following command in global configuration mode:

Command	Purposes
`Router(config)# service slave-log`	Enables slave message logging.

Setting the Syslog Destination

If message logging is enabled, you can send messages to specified locations, in addition to the console.

To set the locations that receive messages, use the following commands in global configuration mode, as needed:

Command	Purposes
`Router(config)# logging buffered` [*size*]	Logs messages to an internal buffer.
`Router(config)# terminal monitor`	Logs messages to a nonconsole terminal.
`Router(config)# logging` *host*	Logs messages to a syslog server host.

The **logging buffered** command copies logging messages to an internal buffer. The buffer is circular, so newer messages overwrite older messages after the buffer is full. To display the messages that are logged in the buffer, use the **show logging** EXEC command. The first message displayed is the oldest message in the buffer. To clear the current contents of the buffer, use the **clear logging** privileged EXEC command.

The **terminal monitor** EXEC command locally accomplishes the task of displaying the system logging messages to a terminal.

The **logging** command identifies a syslog server host to receive logging messages. The *host* argument is the name or IP address of the host. By issuing this command more than once, you build a list of syslog servers that receive logging messages. The **no logging** command deletes the syslog server with the specified address from the list of syslogs.

Configuring Synchronization of Logging Messages

You can configure the system to synchronize unsolicited messages and **debug** command output with solicited device output and prompts for a specific line. You can identify the types of messages to be output asynchronously based on the level of severity. You can also determine the maximum number of buffers for storing asynchronous messages for the terminal after which messages are dropped.

When synchronous logging of unsolicited messages and **debug** command output is turned on, unsolicited device output is displayed on the console or printed after solicited device output is displayed or printed. Unsolicited messages and **debug** command output is displayed on the console after the prompt for user input is returned. Therefore, unsolicited messages and **debug** command output are not interspersed with solicited device output and prompts. After the unsolicited messages are displayed, the console displays the user prompt again.

To configure for synchronous logging of unsolicited messages and **debug** command output with solicited device output and prompts, use the following commands beginning in global configuration mode:

	Command	Purposes
Step 1	`Router(config)# line [aux | console | vty] beginning-line-number [ending-line-number]`	Specifies the line to be configured for synchronous logging of messages.
Step 2	`Router(config-line)# logging synchronous [level severity-level | all] [limit number-of-buffers]`	Enables synchronous logging of messages.

Enabling Time-Stamps on Log Messages

By default, log messages are not time-stamped. To enable time-stamping of log messages, use either of the following commands in global configuration mode:

Command	Purposes

`Router(config)# service timestamps log uptime`
or
`Router(config)# service timestamps log datetime [msec] [localtime] [show-timezone]`
	Enables log time stamps.

Limiting the Error Message Severity Level and Facilities

You can limit the number of messages displayed to the selected device by specifying the severity level of the error message (see Table 1 for level descriptions). To do so, use the following commands in global configuration mode, as needed:

Command	Purposes
`Router(config)# logging console level`	Limits the number of messages logged to the console.
`Router(config)# logging monitor level`	Limits the number of messages logged to the terminal lines.
`Router(config)# logging trap level`	Limits the number of messages logged to the syslog servers.

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

Messages are stored in the history table because SNMP traps are not guaranteed to reach their destination. By default, one message of the level warning and above (see Table 1) is stored in the history table even if syslog traps are not enabled.

To change level and table size defaults, use the following commands in global configuration mode:

	Command	Purposes
Step 1	`Router(config)# logging history level`	Changes the default level of syslog messages stored in the history file and sent to the SNMP server.
Step 2	`Router(config)# logging history size number`	Changes the number of syslog messages that can be stored in the history table.

Note Table 1 lists the level keywords and severity level. For SNMP usage, the severity level values use +1. For example, **emergency** equals 1 not 0 and **critical** equals 3 not 2.

The **logging console** command limits the logging messages displayed on the console terminal to messages with a level number at or below the specified severity level, which is specified by the *level* argument. Table 1 lists the error message *level* keywords and corresponding UNIX syslog definitions in order from the most severe level to the least severe level.

Table 1 System Logging Message Severity Levels

Level Keyword	Level	Description	Syslog Definition
emergencies	0	System unusable	LOG_EMERG
alerts	1	Immediate action needed	LOG_ALERT
critical	2	Critical conditions	LOG_CRIT
errors	3	Error conditions	LOG_ERR
warnings	4	Warning conditions	LOG_WARNING
notifications	5	Normal but significant condition	LOG_NOTICE
informational	6	Informational messages only	LOG_INFO
debugging	7	Debugging messages	LOG_DEBUG

The **no logging console** command disables logging to the console terminal.

The default is to log messages to the console at the **debugging** level and those level numbers that are lower, which means all levels. The **logging monitor** command defaults to **debugging** also. The **logging trap** command defaults to the **informational** level.

To display logging messages on a terminal, use the **terminal monitor** EXEC command.

Current software generates the following four categories of error messages:

- Error messages about software or hardware malfunctions, displayed at levels **warnings** through **emergencies**

- Output from the **debug** commands, displayed at the **debugging** level
- Interface up/down transitions and system restart messages, displayed at the **notifications** level
- Reload requests and low-process stack messages, displayed at the **informational** level

Defining the UNIX System Logging Facility

You can log messages produced by UNIX system utilities. To do this, enable this type logging and define the UNIX system facility from which you want to log messages. Table 2 lists the UNIX system facilities supported by the Cisco IOS software. Consult the operator manual for your UNIX operating system for more information about these UNIX system facilities. The syslog format is compatible with Berkeley Standard Distribution (BSD) UNIX version 4.3.

To define UNIX system facility message logging, use the following command in global configuration mode:

Command	Purposes
`Router(config)# logging facility facility-type`	Configures system log facilities.

Table 2 Logging Facility Type Keywords

Facility Type Keyword	Description
auth	Indicates the authorization system.
cron	Indicates the cron facility.
daemon	Indicates the system daemon.
kern	Indicates the Kernel.
local0–7	Reserved for locally defined messages.
lpr	Indicates line printer system.
mail	Indicates mail system.
news	Indicates USENET news.
sys9	Indicates system use.
sys10	Indicates system use.
sys11	Indicates system use.
sys12	Indicates system use.
sys13	Indicates system use.
sys14	Indicates system use.
syslog	Indicates the system log.
user	Indicates user process.
uucp	Indicates UNIX-to-UNIX copy system.

Displaying Logging Information

To display logging information, use the following commands in EXEC mode, as needed:

Command	Purposes
`Router# show logging`	Displays the state of syslog error and event logging, including host addresses, whether console logging is enabled, and other logging statistics.
`Router# show controllers vip slot-number logging`	Displays the state of syslog error and event logging of a VIP card, including host addresses, whether console logging is enabled, and other logging statistics.
`Router# show logging history`	Displays information in the syslog history table such as the table size, the status of messages, and the text of the messages stored in the table.

Logging Errors to a UNIX Syslog Daemon

To configure the syslog daemon on a 4.3 BSD UNIX system, include a line such as the following in the /etc/syslog.conf file:

```
local7.debugging /usr/adm/logs/cisco.log
```

The **debugging** keyword specifies the syslog level; see Table 1 for a general description of other keywords. The **local7** keyword specifies the logging facility to be used; see Table 2 for a general description of other keywords.

The syslog daemon sends messages at this level or at a more severe level to the file specified in the next field. The file must already exist, and the syslog daemon must have permission to write to it.

Setting the Syslog Source Address

By default, a syslog message contains the IP address of the interface it uses to leave the router. To set all syslog messages to contain the same IP address, regardless of which interface they use, use the following command in global configuration mode:

Command	Purposes
`Router(config)# logging source-interface type number`	Sets the syslog source address.

Using Field Diagnostics on Line Cards

Each line card on the Cisco 12000 series routers can perform field diagnostic testing to isolate faulty hardware without disrupting normal operation of the system. However, performing field diagnostic testing on a line card does halt all activity on the line card for the duration of the testing. After successful completion of the field diagnostic testing, the Cisco IOS software is automatically reloaded on the line card.

Note The field diagnostic **diag** command must be executed from the Gigabit Route Processor (GRP) main console port.

To perform field diagnostic testing on a line card, use the following command in privileged EXEC mode:

Command	Purposes
`Router# diag slot-number [previous | post | verbose | wait]`	Specifies the line card on which you want to perform diagnostic testing. Optionally, specifies that previous test results are displayed, that only extended power-on self-tests (POST) be performed, that the maximum messages are displayed, or that the Cisco IOS software not be reloaded on the line card after successful completion of the tests. The following prompt is displayed: `Running Diags will halt ALL activity on the requested slot. [confirm]` At the prompt, press **Return** to confirm that you want to perform field diagnostic testing on the specified line card, or type **no** to stop the testing.

To stop field diagnostic testing on a line card, use either of the following commands in privileged EXEC mode:

Command	Purpose

`Router# diag slot-number halt`
or
`Router# no diag slot-number`
	Specifies the line card on which you want to stop diagnostic testing.

Note When you stop the field diagnostic test, the line card remains down (that is, in an unbooted state). In most cases, you stopped the testing because you need to remove the line card or replace the line card. If that is not the case and you want to bring the line card back up (that is, online), you must use the **microcode reload** global configuration command or power cycle the line card.

Troubleshooting Specific Line Cards

Cisco IOS provides the **execute-on** command to allow you to issue Cisco IOS commands (such as **show** commands) to a specific line card for monitoring and maintenance. For example, you could show which Cisco IOS image is loaded on the card in slot 3 of a Cisco 12012 Gigabit Switch Router (GSR) by issuing the **execute-on slot 3 show version** command. You can also use this command for troubleshooting cards in the dial shelf of Cisco access servers. For complete documentation of this command, refer to the "Troubleshooting" chapter of the Release 12.2 *Cisco IOS Configuration Fundamentals Command Reference*.

Storing Line Card Crash Information

This section explains how to enable storing of crash information for a line card and optionally specify the type and amount of information stored. Technical support representatives need to be able to look at the crash information from the line card to troubleshoot serious problems on the line card. The crash information contains all the line card memory information, including the main memory and transmit and receive buffer information.

Caution Use the **exception linecard** global configuration command only when directed by a technical support representative, and only enable options that the technical support representative requests you to enable.

To enable and configure the crash information options for a line card, use the following command in global configuration mode:

Command	Purpose
`Router(config)# exception linecard {all | slot slot-number} [corefile filename | main-memory size [k | m] | queue-ram size [k | m] | rx-buffer size [k | m] | sqe-register-rx | sqe-register-tx | tx-buffer size [k | m]]`	Specifies the line card for which you want crash information when a line card resets. Optionally, specify the type and amount of memory to be stored.

Creating Core Dumps for System Exceptions

"System exceptions" are any unexpected system shutdowns or reboots (most frequently caused by a system failure, commonly referred to as a "system crash"). When an exception occurs, it is sometimes useful to obtain a full copy of the memory image (called a core dump) to identify the cause of the unexpected shutdown. Not all exception types will produce a core dump.

Core dumps are generally useful only to your technical support representative. The core dump file, which is a very large binary file, can be transferred to a Trivial File Transfer Protocol (TFTP), File Transfer Protocol (FTP), or Remote Copy Protocol (RCP) server, or (on limited platforms) saved to the flash disk, and subsequently interpreted by technical personnel who have access to source code and detailed memory maps.

Caution Use the **exception** commands only under the direction of a technical support representative. Creating a core dump while the router is functioning in a network can disrupt network operation.

Specifying the Destination for the Core Dump File

To configure the router to generate a core dump, you must enable exception dumps and configure a destination for the core dump file, as described in the following sections:

- Using TFTP for Core Dumps, page 12
- Using FTP for Core Dumps, page 12
- Using rcp for Core Dumps, page 13
- Using a Flash Disk for Core Dumps, page 13

Using TFTP for Core Dumps

Due to a limitation of most TFTP applications, the router will dump only the first 16 MB of the core file. Therefore, if your router's main memory is larger than 16 MB, do not use TFTP.

To configure a router for a core dump using TFTP, use the following commands in global configuration mode:

	Command or Action	Purpose
Step 1	**exception protocol tftp**	(Optional) Explicitly specifies TFTP as the protocol to be used for router exceptions (core dumps for unexpected system shutdowns). **Note** Because TFTP is the default exception protocol, the **exception protocol tftp** command does not need to be used unless the protocol has been previously changed to ftp or rcp in your system's configuration. To determine if the exception protocol has been changed, use the **show running-config** command in EXEC mode.
Step 2	**exception dump** *ip-address*	Configures the router to dump a core file to the specified server if the router crashes.
Step 3	**exception core-file** [*filepath/*]*filename*	(Optional) Specifies the name to be used for the core dump file. The file usually must pre-exist on the TFTP server, and be writable.

For example, the following command configures a router to send a core file to the server at the IP address 172.17.92.2. As the exception protocol is not specified, the default protocol of TFTP will be used.

```
Router(config)# exception dump 172.17.92.2
```

The core dump is written to a file named "*hostname*-core" on the TFTP server, where *hostname* is the name of the route (in the example above, the file would be named Router-core). You can change the name of the core file by adding the **exception core-file** *filename* configuration command.

Depending on the TFTP server application used, it may be necessary to create, on the TFTP server, the empty target file to which the router can write the core. Also, make sure there is enough memory on your TFTP server to hold the complete core dump.

Using FTP for Core Dumps

To configure the router for a core dump using FTP, use the following commands in global configuration mode:

	Command	Purposes
Step 1	Router(config)# **ip ftp username** *username*	(Optional) Configures the user name for FTP connections.
Step 2	Router(config)# **ip ftp password** [***type***] *password*	(Optional) Specifies the password to be used for FTP connections.
Step 3	Router(config)# **exception protocol ftp**	Specifies that FTP should be used for core dump file transfers.

	Command	Purposes
Step 4	`Router(config)# exception dump ip-address`	Configures the router to dump a core file to a particular server if the router crashes.
Step 5	`Router(config)# exception core-file filename`	(Optional) Specifies the name to be used for the core dump file.

The following example configures a router to use FTP to dump a core file named "dumpfile" to the FTP server at 172.17.92.2 when it crashes.

```
ip ftp username red
ip ftp password blue
exception protocol ftp
exception dump 172.17.92.2
exception core-file dumpfile
```

Using rcp for Core Dumps

The remote copy protocol can also be used to send a core dump file. To configure the router to send core dump files using rcp, use the following commands:

	Command or Action	Purpose
Step 1	`ip rcmd remote-username username`	(Optional) Specifies the username sent by the router to the remote server with an rcp copy/write request. The remote rcp server must configured to grant write access to the specified username (in other words, an account must be defined on the network server for the username).
Step 2	`exception protocol rcp`	Configures the rcp as the protocol to use for sending core dump files.
Step 3	`exception dump ip-address`	Configures the router to dump a core file to the specified server if the router crashes.
Step 4	`exception core-file filename`	(Optional) Specifies the name to be used for the core dump file.

When an rcp username is not configured through the **ip rcmd remote-username** command, the rcp username defaults to the username associated with the current terminal (tty) connection. For example, if the user is connected to the router through Telnet and was authenticated through the username command, the router software sends the Telnet username as the rcp username. If the terminal username is not available, the router hostname will be used as the rcp username.

Using a Flash Disk for Core Dumps

Some router platforms support the Flash disk as an alternative to the linear Flash memory or PCMCIA Flash card. The large storage capacity of these Flash disks makes them good candidates for another means of capturing a core dump. To configure a router for a core dump using a Flash disk, use the following command in global configuration mode:

Command	Purpose
`Router(config)# exception flash [procmem |iomem | all] device-name[:partition-number] [erase | no_erase]`	Configures the router for a core dump using a flash disk.
`Router(config)# exception core-file filename`	(Optional) Specifies the name to be used for the core dump file.

The **show flash all** EXEC command will list the devices you can use for the **exception flash** command.

Creating an Exception Memory Core Dump

To cause the router to create a core dump and reboot when certain memory size parameters are violated during the debugging process, use the following commands in global configuration mode:

As a debugging procedure, you can cause the router to create a core dump and reboot when certain memory size parameters are violated. The following **exception memory** commands are used to trigger a core dump:

Command	Purpose
`Router(config)# exception memory minimum bytes`	Triggers a core dump and system reload when the amount of free memory falls below the specified number of bytes. • Do not specify too low a memory value, as the router needs some amount of free memory to provide the core dump. • If you enter a size that is greater than the free memory (and the **exception dump** command has been configured), a core dump and router reload is generated after 60 seconds.
`Router(config)# memory check-interval seconds`	(Optional) Increases the interval at which memory will be checked. The default is 60 seconds, but much can happen in 60 seconds to mask the cause of corruption. Reducing the interval will increase CPU utilization (by around 12%) which will be acceptable in most cases, but will also increase the chance of getting a usable core. To make sure CPU utilization doesn't hit 100%, you should gradually decrease the interval on busy routers. The ideal interval is as low as possible without causing other system problems.
`Router(config)# exception memory fragment bytes`	Triggers a core dump and system reload when the amount of contiguous (non-fragmented) free memory falls below the specified number of bytes.
`Router(config)# exception core-file filename`	(Optional) Specifies the name to be used for the core dump file. The file usually must exist on the TFTP server, and be writable. Note that the file will be the same size as the amount of processor memory on the router.

Note that the **exception memory minimum** command is primarily useful if you anticipate running out of memory before a core dump can be triggered or other debugging can be performed (rapid memory leak); if the memory leak is gradual (slow drift), you have generally have time to perform debugging before the system runs out of memory and must be reloaded.

By default, the number of free memory bytes is checked every 60 seconds when these commands are configured. The frequency of this checking can be increased using the **memory check-interval** *seconds* command.

The **exception dump** *ip-address* command must be configured with these commands. If the **exception dump** command is not configured, the router reloads without triggering a core dump.

The following example configures the router to monitor the free memory. If the memory falls below 250000 bytes, the core dump is created and the router reloads.

```
exception dump 172.18.92.2
exception core-file memory.overrun
exception memory minimum 250000
```

Setting a Spurious Interrupt Core Dump

During the debugging process, you can configure the router to create a spurious interrupt core dump and reboot when a specified number of interrupts have occurred.

Caution Use the **exception spurious-interrupt** global configuration command only when directed by a technical support representative and only enable options requested by the technical support representative.

To enable and configure the crash information for spurious interrupts, use the following commands in global configuration mode:

Command	Purpose
`Router(config)# exception spurious-interrupt` *number*	Sets the maximum number of spurious interrupts to include in the core dump before reloading.
`Router(config)# exception dump` *ip-address* or `Router(config)# exception flash`	Specifies the destination for the core dump file.

The following example configures a router to create a core dump with a limit of two spurious interrupts:

```
exception spurious-interrupt 2
exception dump 209.165.200.225
```

Enabling Debug Operations

Your router includes hardware and software to aid in troubleshooting internal problems and problems with other hosts on the network. The **debug** privileged EXEC mode commands start the console display of several classes of network events. The following commands describe in general the system debug message feature. Refer to the *Cisco IOS Debug Command Reference* for all information regarding **debug** commands. Also refer to the *Internetwork Troubleshooting Guide* publication for additional information.

To enable debugging operations, use the following commands:

Command	Purposes
`Router# show debugging`	Displays the state of each debugging option.
`Router# debug ?`	Displays a list and brief description of all the **debug** command options.
`Router# debug` *command*	Begins message logging for the specified **debug** command.
`Router# no debug` *command*	Turns message logging off for the specified **debug** command.

Caution The system gives high priority to debugging output. For this reason, debugging commands should be turned on only for troubleshooting specific problems or during troubleshooting sessions with technical support personnel. Excessive debugging output can render the system inoperable.

You can configure time-stamping of system **debug** messages. Time-stamping enhances real-time debugging by providing the relative timing of logged events. This information is especially useful when customers send debugging output to your technical support personnel for assistance. To enable time-stamping of system **debug** messages, use either of the following commands in global configuration mode:

Command	Purposes

`Router(config)# service timestamps debug uptime`
 	
or
 	
`Router(config)# service timestamps debug datetime [msec] [localtime] [show-timezone]`
	Enables time-stamping of system **debug** messages.

Normally, the messages are displayed only on the console terminal. Refer to the section "Setting the Syslog Destination, page 5" earlier in this chapter to change the output device.

Enabling Conditionally Triggered Debugging

When the Conditionally Triggered Debugging feature is enabled, the router generates debugging messages for packets entering or leaving the router on a specified interface; the router will not generate debugging output for packets entering or leaving through a different interface. You can specify the interfaces explicitly. For example, you may only want to see debugging messages for one interface or subinterface. You can also turn on debugging for all interfaces that meet specified condition. This feature is useful on dial access servers, which have a large number of ports.

Normally, the router will generate debugging messages for every interface, resulting in a large number of messages. The large number of messages consumes system resources, and can affect your ability to find the specific information you need. By limiting the number of debugging messages, you can receive messages related to only the ports you wish to troubleshoot.

Conditionally Triggered Debugging controls the output from the following protocol-specific **debug** commands:

- **debug aaa** {**accounting** | **authorization** | **authentication**}

- **debug dialer** {**events** | **packets**}
- **debug isdn** {**q921** | **q931**}
- **debug modem** {**oob** | **trace**}
- **debug ppp** {**all** | **authentication** | **chap** | **error** | **negotiation** | **multilink events** | **packet**}

Although this feature limits the output of the commands listed, it does not automatically enable the generation of debugging output from these commands. Debugging messages are generated only when the protocol-specific **debug** command is enabled. The **debug** command output is controlled through two processes:

- The protocol-specific **debug** commands specify which protocols are being debugged. For example, the **debug dialer events** command generates debugging output related to dialer events.
- The **debug condition** commands limit these debugging messages to those related to a particular interface. For example, the **debug condition username bob** command generates debugging output only for interfaces with packets that specify a username of bob.

To configure Conditionally Triggered Debugging, perform the tasks described in the following sections:

- Enabling Protocol-Specific debug Commands, page 17
- Enabling Conditional Debugging Commands, page 17
- Specifying Multiple Debugging Conditions, page 19

Enabling Protocol-Specific debug Commands

In order to generate any debugging output, the protocol-specific **debug** command for the desired output must be enabled. Use the **show debugging** command to determine which types of debugging are enabled. To display the current debug conditions, use the **show debug condition** command. To enable the desired protocol-specific **debug** commands, use the following commands in privileged EXEC mode:

Command	Purpose
`Router# show debugging`	Determines which types of debugging are enabled.
`Router# show debug condition [condition-id]`	Displays the current **debug** conditions.
`Router# debug protocol`	Enables the desired debugging commands.
`Router# no debug protocol`	Disables the debugging commands that are not desired.

If you do not want output, disable all the protocol-specific **debug** commands.

Enabling Conditional Debugging Commands

If no **debug condition** commands are enabled, all debugging output, regardless of the interface, will be displayed for the enabled protocol-specific **debug** commands.

The first **debug condition** command you enter enables conditional debugging. The router will display only messages for interfaces that meet one of the specified conditions. If multiple conditions are specified, the interface must meet at least one of the conditions in order for messages to be displayed.

To enable messages for interfaces specified explicitly or for interfaces that meet certain conditions, perform the tasks described in the following sections:

Displaying Messages for One Interface

To disable debugging messages for all interfaces except one, use the following command in privileged EXEC mode:

Command	Purpose
`Router# debug condition interface interface`	Enables debugging output for only the specified interface.

To reenable debugging output for all interfaces, use the **no debug interface** command.

Displaying Messages for Multiple Interfaces

To enable debugging messages for multiple interfaces, use the following commands in privileged EXEC mode:

	Command	Purposes
Step 1	`Router# debug condition interface interface`	Enables debugging output for only the specified interface
Step 2	`Router# debug condition interface interface`	Enable debugging messages for additional interfaces. Repeat this task until debugging messages are enabled for all desired interfaces.

If you specify more than one interface by entering this command multiple times, debugging output will be displayed for all of the specified interfaces. To turn off debugging on a particular interface, use the **no debug interface** command. If you use the **no debug interface all** command or remove the last **debug interface** command, debugging output will be reenabled for all interfaces.

Limiting the Number of Messages Based on Conditions

The router can monitor interfaces to learn if any packets contain the specified value for one of the following conditions:

- username
- calling party number
- called party number

If you enter a condition, such as calling number, debug output will be stopped for all interfaces. The router will then monitor every interface to learn if a packet with the specified calling party number is sent or received on any interfaces. If the condition is met on an interface or subinterface, **debug** command output will be displayed for that interface. The debugging output for an interface is “triggered” when the condition has been met. The debugging output continues to be disabled for the other interfaces. If, at some later time, the condition is met for another interface, the debug output also will become enabled for that interface.

Once debugging output has been triggered on an interface, the output will continue until the interface goes down. However, the session for that interface might change, resulting in a new username, called party number, or calling party number. Use the **no debug interface** command to reset the debug trigger mechanism for a particular interface. The debugging output for that interface will be disabled until the interface meets one of the specified conditions.

To limit the number of debugging messages based on a specified condition, use the following command in privileged EXEC mode:

Command	Purpose
`Router# debug condition {username username | called dial-string | caller dial-string}`	Enables conditional debugging. The router will display only messages for interfaces that meet this condition.

To reenable the debugging output for all interfaces, enter the **no debug condition all** command.

Specifying Multiple Debugging Conditions

To limit the number of debugging messages based on more than one condition, use the following commands in privileged EXEC mode:

	Command	Purposes
Step 1	`Router# debug condition {username username | called dial-string | caller dial-string}`	Enables conditional debugging, and specifies the first condition.
Step 2	`Router# debug condition {username username | called dial-string | caller dial-string}`	Specifies the second condition. Repeat this task until all conditions are specified.

If you enter multiple **debug condition** commands, debugging output will be generated if an interface meets at least one of the conditions. If you remove one of the conditions using the **no debug condition** command, interfaces that meet only that condition no longer will produce debugging output. However, interfaces that meet a condition other than the removed condition will continue to generate output. Only if no active conditions are met for an interface will the output for that interface be disabled.

Conditionally Triggered Debugging Configuration Examples

In this example, four conditions have been set by the following commands:

- **debug condition interface serial 0**
- **debug condition interface serial 1**
- **debug condition interface virtual-template 1**
- **debug condition username fred**

The first three conditions have been met by one interface. The fourth condition has not yet been met:

```
Router# show debug condition

Condition 1: interface Se0 (1 flags triggered)
        Flags: Se0
Condition 2: interface Se1 (1 flags triggered)
        Flags: Se1
```

```
Condition 3: interface Vt1 (1 flags triggered)
        Flags: Vt1
Condition 4: username fred (0 flags triggered)
```

When any **debug condition** command is entered, debugging messages for conditional debugging are enabled. The following debugging messages show conditions being met on different interfaces as the serial 0 and serial 1 interfaces come up. For example, the second line of output indicates that serial interface 0 meets the username fred condition.

```
*Mar  1 00:04:41.647: %LINK-3-UPDOWN: Interface Serial0, changed state to up
*Mar  1 00:04:41.715: Se0 Debug: Condition 4, username fred triggered, count 2
*Mar  1 00:04:42.963: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0, changed
state to up
*Mar  1 00:04:43.271: Vi1 Debug: Condition 3, interface Vt1 triggered, count 1
*Mar  1 00:04:43.271: %LINK-3-UPDOWN: Interface Virtual-Access1, changed state to up
*Mar  1 00:04:43.279: Vi1 Debug: Condition 4, username fred triggered, count 2
*Mar  1 00:04:43.283: Vi1 Debug: Condition 1, interface Se0 triggered, count 3
*Mar  1 00:04:44.039: %IP-4-DUPADDR: Duplicate address 172.27.32.114 on Ethernet 0,
sourced by 00e0.1e3e.2d41
*Mar  1 00:04:44.283: %LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access1,
changed state to up
*Mar  1 00:04:54.667: %LINK-3-UPDOWN: Interface Serial1, changed state to up
*Mar  1 00:04:54.731: Se1 Debug: Condition 4, username fred triggered, count 2
*Mar  1 00:04:54.735: Vi1 Debug: Condition 2, interface Se1 triggered, count 4
*Mar  1 00:04:55.735: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial1, changed
state to up
```

After a period of time, the **show debug condition** command displays the revised list of conditions:

```
Router# show debug condition

Condition 1: interface Se0 (2 flags triggered)
        Flags: Se0 Vi1
Condition 2: interface Se1 (2 flags triggered)
        Flags: Se1 Vi1
Condition 3: interface Vt1 (2 flags triggered)
        Flags: Vt1 Vi1
Condition 4: username fred (3 flags triggered)
        Flags: Se0 Vi1 Se1
```

Next, the serial 1 and serial 0 interfaces go down. When an interface goes down, conditions for that interface are cleared.

```
*Mar  1 00:05:51.443: %LINK-3-UPDOWN: Interface Serial1, changed state to down
*Mar  1 00:05:51.471: Se1 Debug: Condition 4, username fred cleared, count 1
*Mar  1 00:05:51.479: Vi1 Debug: Condition 2, interface Se1 cleared, count 3
*Mar  1 00:05:52.443: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial1, changed
state to down
*Mar  1 00:05:56.859: %LINK-3-UPDOWN: Interface Serial0, changed state to down
*Mar  1 00:05:56.887: Se0 Debug: Condition 4, username fred cleared, count 1
*Mar  1 00:05:56.895: Vi1 Debug: Condition 1, interface Se0 cleared, count 2
*Mar  1 00:05:56.899: Vi1 Debug: Condition 3, interface Vt1 cleared, count 1
*Mar  1 00:05:56.899: Vi1 Debug: Condition 4, username fred cleared, count 0
*Mar  1 00:05:56.903: %LINK-3-UPDOWN: Interface Virtual-Access1, changed state to down
*Mar  1 00:05:57.907: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0, changed
state to down
*Mar  1 00:05:57.907: %LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access1,
changed state to down
```

The final **show debug condition** output is the same as the output before the interfaces came up:

```
Router# show debug condition

Condition 1: interface Se0 (1 flags triggered)
```

```
        Flags: Se0
Condition 2: interface Se1 (1 flags triggered)
        Flags: Se1
Condition 3: interface Vt1 (1 flags triggered)
        Flags: Vt1
Condition 4: username fred (0 flags triggered)
```

Using the Environmental Monitor

Some routers and access servers have an environmental monitor that monitors the physical condition of the router. If a measurement exceeds acceptable margins, a warning message is printed to the system console. The system software collects measurements once every 60 seconds, but warnings for a given test point are printed at most once every 4 hours. If the temperature measurements are out of specification more than the shutdown, the software shuts the router down (the fan will remain on). The router must be manually turned off and on after such a shutdown. You can query the environmental monitor using the **show environment** command at any time to determine whether a measurement is out of tolerance. Refer to the *Cisco IOS System Error Messages* publication for a description of environmental monitor warning messages.

On routers with an environmental monitor, if the software detects that any of its temperature test points have exceeded maximum margins, it performs the following steps:

1. Saves the last measured values from each of the six test points to internal nonvolatile memory.
2. Interrupts the system software and causes a shutdown message to be printed on the system console.
3. Shuts off the power supplies after a few milliseconds of delay.

The system displays the following message if temperatures exceed maximum margins, along with a message indicating the reason for the shutdown:

```
Router#
%ENVM-1-SHUTDOWN: Environmental Monitor initiated shutdown
%ENVM-2-TEMP: Inlet temperature has reached SHUTDOWN level at 64(C)
```

Refer to the hardware installation and maintenance publication for your router for more information about environmental specifications.

Embedded Packet Capture

First Published: July 11, 2008
Last Updated: November 20, 2009

Cisco IOS Embedded Packet Capture (EPC) is an onboard packet capture facility that allows network administrators to capture packets flowing to, through or from the device and to analyze them locally or save and export them for offline analysis using a tool like Wireshark. This feature simplifies operations by allowing the devices to become active participants in the management and operation of the network. This feature facilitates better troubleshooting by gathering information on packet format. It also facilitates application analysis and security.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the "Feature Information for Embedded Packet Capture" section on page 12.

Use Cisco Feature Navigator to find information about platform support and Cisco IOS and Catalyst OS software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Contents

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

Prerequisites for Embedded Packet Capture

The EPC software subsystem consumes CPU and memory resources in its operation. You must have adequate system resources for different types of operations. Some guidelines for arranging the system resources are provided in Table 1.

Table 1 System Requirements for EPC subsystem

Hardware	CPU utilization requirements are platform dependent.
Memory	The packet buffer is stored in DRAM. The size of the packet buffer is user specified.
Diskspace	Packets can be exported to external systems. No intermediate storage on flash disk is required.

Restrictions for Embedded Packet Capture

- EPC only captures multicast packets on ingress and does not capture the replicated packets on egress.
- Currently, the capture file can only be exported off the device; for example, TFTP or FTP servers and local disk.

Information About Embedded Packet Capture

To configure the EPC feature, you must understand the following concepts:

- EPC Overview, page 2
- Benefits of EPC, page 2
- Capture Buffer, page 3
- Capture Point, page 3

EPC Overview

EPC provides a better level of embedded systems management that helps in tracing and troubleshooting packets. This feature allows network administrators to capture data packets flowing through, to, and from a Cisco router.

Benefits of EPC

Some of the benefits of this feature include:

- Ability to capture IPv4 and IPv6 packets in the Cisco Express Forwarding (CEF) path.
- A flexible method for specifying the capture buffer parameters.
- Filter captured packets.
- Methods to decode data packets captured with varying degree of detail.

- Facility to export the packet capture in PCAP format suitable for analysis using an external tool.
- Extensible infrastructure for enabling packet capture points.

Capture Buffer

The capture buffer is an area in memory for holding the packet data. You can specify unique names, size and type of the buffer, and configure the buffer to handle incoming data as required.

The following types of data are stored in a capture buffer:

- Packet data
- Metadata

The packet data starts from datagramstart and copies a minimum of the per-packet-capture size or datagramsize to the capture buffer.

The metadata contains descriptive information about a set of packet data. It contains:

- A timestamp of when it is added to a buffer.
- The direction in which the packet data is transmitted—egress or ingress.
- The switch path captured.
- Encapsulation type corresponding to input or output interface to allow the decoding of L2 decoders.

The following actions can be performed on capture buffers:

- Define a capture buffer and associate it with a capture point.
- Clear capture buffers.
- Export capture buffers for offline analysis. Export writes off the file using one of the supported file transfer options: FTP, HTTP, HTTPS, PRAM, RCP, SCP, and TFTP.
- Display content of the capture buffers.

Capture Point

The capture point is a traffic transit point where a packet is captured and associated with a buffer. You can define capture points by providing unique names and different parameters.

The following capture points are available:

- IPv4 CEF/interrupt switching path with interface input and output
- IPv6 CEF/interrupt switching path with interface input and output

You can perform the following actions on the capture point:

- Associate or disassociate capture points with capture buffers. Each capture point can be associated with only one capture buffer.
- Destroy capture points.
- Activate packet capture points on a given interface. Multiple packet capture points can be made active on a given interface. For example, Border Gateway Protocol (BGP) packets can be captured into one capture buffer and Open Shortest Path First (OSPF) packets can captured into another capture buffer.
- Access Control Lists (ACLs) can be applied to capture points.

How to Implement Embedded Packet Capture

This section contains the following tasks:

Starting Packet Data Capture

Perform this task to start capturing packet data for analysis and troubleshooting. To capture packet data, a capture buffer and a capture point need to be defined. The capture point should then be associated with the capture buffer. Enabling the capture point will start the process of capturing packet data.

SUMMARY STEPS

1. **enable**
2. **monitor capture buffer** *buffer-name* [**circular** | **clear** | **export** *export-location* | **filter access-list** {*ip-access-list* | *ip-expanded-list* | *access-list-name*} | **limit** {**allow-nth-pak** *nth-packet* | **duration** *seconds* | **packet-count** *total-packets* | **packets-per-sec** *packets*} | **linear** | **max-size** *element-size* | **size** *buffer-size* [**max-size** *element-size*]]
3. **monitor capture point** {**ip** | **ipv6**} {**cef** *capture-point-name interface-name interface-type* {**both** | **in** | **out**} | **process-switched** *capture-point-name* {**both** | **from-us** | **in** | **out**}}
4. **monitor capture point associate** *capture-point-name capture-buffer-name*
5. **monitor capture point start** {*capture-point-name* | **all**}

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**monitor capture buffer** *buffer-name* [**circular** | **clear** | **export** *export-location* | **filter access-list** {*ip-access-list* | *ip-expanded-list* | *access-list-name*} | **limit** {**allow-nth-pak** *nth-packet* | **duration** *seconds* | **packet-count** *total-packets* | **packets-per-sec** *packets*} | **linear** | **max-size** *element-size* | **size** *buffer-size* [**max-size** *element-size*]] **Example:** `Router# monitor capture buffer pktrace1 size 58 max-size 256 circular`	Defines a capture buffer with the specified name and parameters. • In this example, a circular capture buffer by name pktrace1 with size 58 bytes and maximum size 256 bytes is defined.

	Command or Action	Purpose
Step 3	**monitor capture point** {**ip** | **ipv6**}{**cef** *capture-point-name interface-name interface-type* {**both** | **in** | **out**} | **process-switched** *capture-point-name* {**both** | **from-us** | **in** | **out**}} **Example:** `Router# monitor capture point ip cef ipceffa0/1 fastEthernet 0/1 both`	Defines a capture point with the specified parameters. • In this example, a capture point by name ipceffa0/1 with the Fast Ethernet 0/1 interface in both directions is defined.
Step 4	**monitor capture point associate** *capture-point-name capture-buffer-name* **Example:** `Router# monitor capture point associate ipceffa0/1 pktrace1`	Associates the capture point with the capture buffer specified. • Associating a capture point with a capture buffer results in all packets captured from the specified capture point to be dumped to the associated capture buffer. • In this example, the capture point ipceffa0/1 is associated with the capture buffer pktrace1.
Step 5	**monitor capture point start** {*capture-point-name* | **all**} **Example:** `Router# monitor capture point start ipceffa0/1`	Enables the capture point to start capturing packet data. • In this example, the capture point ipceffa0/1 is enabled.

Stopping Packet Data Capture

Perform this task to stop capturing packet data.

SUMMARY STEPS

1. **enable**
2. **monitor capture point stop** {*capture-point-name* | **all**}

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**monitor capture point stop** {*capture-point-name* | **all**} **Example:** `Router# monitor capture point stop ipceffa0/1`	Disables the capture point and stops the packet data capture process. • In this example, the capture point ipceffa0/1 is disabled.

Exporting Packet Data for Analysis

Perform this task to export the packet data for analysis using an external tool.

SUMMARY STEPS

1. **enable**
2. **monitor capture buffer** *buffer-name* **export** *export-location*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**monitor capture buffer** *buffer-name* **export** *export-location* `Router# monitor capture buffer pktrace1 export tftp://88.1.88.9/pktrace1`	Exports the data for analysis. • In this example, data from the capture buffer pktrace1 is exported using the TFTP protocol.

Monitoring and Maintaining Captured Data

Perform this task to monitor and maintain the packet data captured. Capture buffer details and capture point details can be displayed.

SUMMARY STEPS

1. **enable**
2. **show monitor capture** {**buffer** {*capture-buffer-name* [**parameters**] | **all parameters** | **merged** *capture-buffer-name1 capture-buffer-name2*}[**dump**] [**filter** *filter-parameters*]} | **point** {**all** | *capture-point-name*}}
3. **debug packet-capture**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show monitor capture** {**buffer** {*capture-buffer-name* [**parameters**] | **all parameters** | **merged** *capture-buffer-name1 capture-buffer-name2*}[**dump**] [**filter** *filter-parameters*]} | **point** {**all** | *capture-point-name*}} **Example:** `Router# show monitor capture buffer pktrace1 dump`	Displays the data captured. • In this example, data from the capture buffer pktrace1 is displayed.
Step 3	**debug packet-capture** **Example:** `Router# debug packet-capture`	Enables packet capture infra debugs.

Configuration Examples for Embedded Packet Capture

This section contains the following configuration examples.

Starting Packet Data Capture: Example

The following example shows how to capture packets to and from Fast Ethernet 0/1 interface:

```
Router> enable
Router# monitor capture buffer pktrace1 ip cef ipceffa0/1 fastEthernet 0/1 both
Router# monitor capture point associate ipceffa0/1 pktrace1
Router# monitor capture point start ipceffa0/1

Mar 21 11:13:34.023: %BUFCAP-6-ENABLE: Capture Point ipceffa0/1 enabled.

Router# show monitor capture point all

Status Information for Capture Point ipceffa0/1
IPv4 CEF
Switch Path: IPv4 CEF            , Capture Buffer: pktrace1
Status : Inactive

Configuration:
monitor capture point ip cef ipceffa0/1 FastEthernet0/1 both
```

```
Router# show monitor capture buffer all

Capture buffer pktrace1 (circular buffer)
Buffer Size : 262144 bytes, Max Element Size : 256 bytes, Packets : 31
Allow-nth-pak : 0, Duration : 0 (seconds), Max packets : 0, pps : 0
Associated Capture Points:
Name : ipceffa0/1, Status : Active
Configuration:
monitor capture buffer pktrace1 size 256 max-size 256 circular
monitor capture point associate ipceffa0/1 pktrace1
```

Stopping Packet Data Capture: Example

The following example shows how to stop capturing packet data:

```
Router> enable
Router# monitor capture point stop ipceffa0/1

Mar 21 11:14:20.152: %BUFCAP-6-DISABLE: Capture Point ipceffa0/1 disabled.
```

Exporting Packet Data: Example

The following example shows how to export data for analysis through an external tool:

```
Router> enable
Router# monitor capture buffer pktrace1 export tftp://88.1.88.9/pktrace1
```

Monitoring and Maintaining Captured Data: Example

The EPC feature provides the ability to dump packets in ASCII. The following example shows an IPv4 ICMP echo reply packet from one host to another:

```
<timestamp>: IPv4 packet received on Ethernet0/0 in the IPv4 CEF LES switch path
029E28E0: AABBCC01 2D00AABB CC013000 08004500  *;L.-.*;L.0...E.
029E28F0: 00640001 0000FE01 A8950A00 00020A00  .d....~.(.......
029E2900: 00010000 D5C80001 00000000 00000000  ....UH..........
029E2910: B080ABCD ABCDABCD ABCDABCD ABCDABCD  0.+M+M+M+M+M+M+M
029E2920: ABCDABCD ABCDABCD ABCDABCD ABCDABCD  +M+M+M+M+M+M+M+M
029E2930: ABCDABCD ABCDABCD ABCDABCD ABCDABCD  +M+M+M+M+M+M+M+M
029E2940: ABCDABCD ABCDABCD ABCDABCD ABCDABCD  +M+M+M+M+M+M+M+M
029E2950: ABCD
```

The following example shows how to view the contents of the capture buffer pktrace1. This output is displayed using the **show monitor capture buffer** *capture-buffer-name* **dump** command. This command supports two modes: the default mode and the dump mode. In the dump mode, the hexadecimal dump of the captured packet is also shown.

```
Router> enable
Router# show monitor capture buffer pktrace1 dump

11:13:00.593 EDT Mar 21 2007 : IPv4 Turbo      : Fa2/1 Fa0/1

65B6F500: 080020A2 44D90009 E94F8406 08004500  .. "DY..iO....E.
65B6F510: 00400F00 0000FE01 92AF5801 13025801  .@....~../X...X.
65B6F520: 58090800 4D1A1169 00000000 0005326C  X...M..i......2l
65B6F530: 01CCABCD ABCDABCD ABCDABCD ABCDABCD  .L+M+M+M+M+M+M+M
65B6F540: ABCDABCD ABCDABCD ABCDABCD ABCD00    +M+M+M+M+M+M+M.
```

```
11:13:20.593 EDT Mar 21 2007 : IPv4 Turbo      : Fa2/1 Fa0/1

65B6F500: 080020A2 44D90009 E94F8406 08004500  .. "DY..iO....E.
65B6F510: 00400F02 0000FE01 92AD5801 13025801  .@....~..-X...X.
65B6F520: 58090800 FEF91169 00000000 0005326C  X...~y.i......2l
65B6F530: 4FECABCD ABCDABCD ABCDABCD ABCDABCD  Ol+M+M+M+M+M+M+M
65B6F540: ABCDABCD ABCDABCD ABCDABCD ABCDFF    +M+M+M+M+M+M+M
```

The following example shows how to enable the packet capture infra debugs:

```
Router> enable
Router# debug packet-capture

Buffer Capture Infrastructure debugging is on
```

Additional References

The following sections provide references related to the EPC feature.

Related Documents

Related Topic	Document Title
Cisco IOS commands	*Cisco IOS Master Commands List, All Releases*
Network Management commands (including EEM commands): complete command syntax, defaults, command mode, command history, usage guidelines, and examples.	*Cisco IOS Network Management Command Reference*

Standards

Standard	Title
No new or modified standards are supported, and support for existing standards has not been modified.	—

MIBs

MIB	MIBs Link
No new or modified MIBs are supported by this feature, and support for existing MIBs has not been modified by this feature.	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

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

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for Embedded Packet Capture

Table 2 lists the release history for this feature.

Not all commands may be available in your Cisco IOS software release. For release information about a specific command, see the command reference documentation.

Use Cisco Feature Navigator to find information about platform support and software image support. Cisco Feature Navigator enables you to determine which Cisco IOS and Catalyst OS software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Note Table 2 lists only the Cisco IOS software release that introduced support for a given feature in a given Cisco IOS software release train. Unless noted otherwise, subsequent releases of that Cisco IOS software release train also support that feature.

Table 2 Feature Information for Embedded Packet Capture

Feature Name	Releases	Feature Information
Embedded Packet Capture	12.4(20)T 12.2(33)SRE	Cisco IOS Embedded Packet Capture (EPC) is an onboard packet capture facility that allows network administrators to capture packets flowing to, through or from the device and to analyze them locally or save and export them for offline analysis using a tool like Wireshark. This feature simplifies operations by allowing the devices to become active participants in the management and operation of the network. This feature facilitates better troubleshooting by gathering information on packet format. It also facilitates application analysis and security. This feature was introduced in Cisco IOS Release 12.4(20)T and integrated into Cisco IOS Release 12.2(33)SRE. The following sections provide information about this feature: • Information About Embedded Packet Capture, page 2 The following commands were introduced or modified: **debug packet-capture**, **monitor capture buffer**, **monitor capture point**, **monitor capture point associate**, **monitor capture point disassociate**, **monitor capture point start**, **monitor capture point stop**, **show monitor capture.**

Cisco IOS Scripting with Tcl

Cisco IOS Scripting with Tcl

First Published: July 28, 2003
Last Updated: May 19, 2008

The Cisco IOS Scripting with Tcl feature provides the ability to run Tool Command Language (Tcl) version 8.3.4 commands from the Cisco IOS command-line interface (CLI).

Finding Feature Information in This Module

Your Cisco IOS software release may not support all of the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To reach links to specific feature documentation in this module and to see a list of the releases in which each feature is supported, use the "Feature Information for Cisco IOS Scripting with Tcl" section on page 19.

Finding Support Information for Platforms and Cisco IOS and Catalyst OS Software Images

Use Cisco Feature Navigator to find information about platform support and Cisco IOS and Catalyst OS software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Contents

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

Prerequisites for Cisco IOS Scripting with Tcl

- Familiarity with Tcl programming and Cisco IOS commands is assumed.
- Tcl commands can be executed from the Tcl configuration mode using the Cisco IOS CLI. Tcl configuration mode, like global configuration mode, is accessed from privileged EXEC mode. Access to privileged EXEC mode should be managed by restricting access using the **enable** command password.

Restrictions for Cisco IOS Scripting with Tcl

- If Cisco IOS configuration commands are used within the Tcl scripts, submode commands must be entered as quoted arguments on the same line as the configuration command.
- Error messages are provided, but you must check that the Tcl script will run successfully because errors may cause the Tcl shell to run in an infinite loop.

Caution The use of Tcl server sockets to listen to telnet and FTP ports (23 and 21 respectively) will preempt the normal handling of these ports in Cisco IOS software.

- Table 1 lists Tcl commands and library calls that do not behave within Cisco IOS software as documented in standard Tcl documents.

Table 1 Tcl Command Options That Behave Differently in Cisco IOS Software

Command	Keyword	Argument	Supported	Comments
after	**ms**	*script*	Partially	When the CLI **tclsh** command is used, there is no event loop implemented unless Embedded Syslog Manager (ESM) is active on the same router. Commands entered using the **after** Tcl command will not run unless forced using the **update** command. Sleep mode (the **after** command) works only with the **ms** keyword.
file	**-time**	*atime*	No	The optional **-time** keyword to set the file access time is not supported in Cisco IOS software.
file	**-time**	*mtime*	No	The optional **-time** keyword to set the file modification time is not supported in Cisco IOS software.
fileevent			Partially	When the CLI **tclsh** command is used, there is no event loop implemented unless Embedded Syslog Manager (ESM) is active on the same router. Commands entered using the **fileevent** Tcl command will not run unless forced using the **update** command.

Table 1 Tcl Command Options That Behave Differently in Cisco IOS Software (continued)

Command	Keyword	Argument	Supported	Comments
history	**!***n*		Partially	The **!***n* shortcut does not work in Cisco IOS software. Use the **history** Tcl command with the **redo** *n* keyword.
load			No	When the CLI **load** command is used, an error message stating "dynamic loading not available on this system" is displayed.

Information About Cisco IOS Scripting with Tcl

To create and use Tcl scripts within Cisco IOS software, you should understand the following concepts:

- Tcl Shell for Cisco IOS Software, page 3
- Tcl Precompiler, page 3
- SNMP MIB Object Access, page 4

Tcl Shell for Cisco IOS Software

The Cisco IOS Tcl shell was designed to allow customers to run Tcl commands directly from the Cisco IOS CLI prompt. Cisco IOS software does contain some subsystems such as Embedded Syslog Manager (ESM) and Interactive Voice Response (IVR) that use Tcl interpreters as part of their implementation. These subsystems have their own proprietary commands and keyword options that are not available in the Tcl shell.

Several methods have been developed for creating and running Tcl scripts within Cisco IOS software. A Tcl shell can be enabled, and Tcl commands can be entered line by line. After Tcl commands are entered, they are sent to a Tcl interpreter. If the commands are recognized as valid Tcl commands, the commands are executed and the results are sent to the tty. If a command is not a recognized Tcl command, it is sent to the Cisco IOS CLI parser. If the command is not a Tcl or Cisco IOS command, two error messages are displayed. A predefined Tcl script can be created outside of Cisco IOS software, transferred to flash or disk memory, and run within Cisco IOS software. It is also possible to create a Tcl script and precompile the code before running it under Cisco IOS software.

Multiple users on the same router can be in Tcl configuration mode at the same time without interference because each Tcl shell session launches a separate interpreter and Tcl server process. The tty interface number served by each Tcl process is represented in the server process name and can be displayed using the **show process** CLI command.

The Tcl shell can be used to run Cisco IOS CLI EXEC commands within a Tcl script. Using the Tcl shell to run CLI commands allows customers to build menus to guide novice users through tasks, to automate repetitive tasks, and to create custom output for **show** commands.

Tcl Precompiler

The Cisco IOS Tcl implementation offers support for loading scripts that have been precompiled by the TclPro precompiler. Precompiled scripts allow a measure of security and consistency because they are obfuscated.

SNMP MIB Object Access

Designed to make access to Simple Network Management Protocol (SNMP) MIB objects easier, a set of UNIX-like SNMP commands has been created. The Tcl shell is enabled either manually or by using a Tcl script, and the new commands can be entered to allow you to perform specified get and set actions on MIB objects. To increase usability, the new commands have names similar to those used for UNIX SNMP access. To access the SNMP commands go to, "Using the Tcl Shell to Access SNMP MIB Objects" section on page 7.

How to Configure Cisco IOS Scripting with Tcl

This section contains the following tasks:

Enabling the Tcl Shell and Using the CLI to Enter Commands

Perform this task to enable the interactive Tcl shell and to enter Tcl commands line by line through the Cisco IOS CLI prompt. Optional steps include specifying a default location for encoding files and specifying an initialization script.

Custom Extensions in the Tcl Shell

The Cisco IOS implementation of the Tcl shell contains some custom command extensions. These extensions operate only under Tcl configuration mode. Table 2 displays these command extensions.

Table 2 Cisco IOS Custom Tcl Command Extensions

Command	Description
ios_config	Runs a Cisco IOS CLI configuration command.
log_user	Toggles Tcl command output under Tcl configuration mode.
typeahead	Writes text to the router standard input (stdin) buffer file.
tclquit	Leave Tcl shell—synonym for **exit**.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **scripting tcl encdir** *location-url*
4. **scripting tcl init** *init-url*
5. **exit**
6. **tclsh**
7. Enter the required Tcl command language syntax.

8. **ios_config** "*cmd*" "*cmd-option*"
9. **exec** "*exec-cmd*"
10. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	(Optional) Enters global configuration mode. • Perform Step 2 through Step 6 if you are using encoding files, an initialization script, or both.
Step 3	**scripting tcl encdir** *location-url* **Example:** `Router(config)# scripting tcl encdir tftp://10.18.117.23/enctcl/`	(Optional) Specifies the default location of external encoding files used by the Tcl **encoding** command.
Step 4	**scripting tcl init** *init-url* **Example:** `Router(config)# scripting tcl init ftp://user:password@172.17.40.3/tclscript/initfiles3.tcl`	(Optional) Specifies an initialization script to run when the Tcl shell is enabled.
Step 5	**scripting tcl low-memory** *bytes* **Example:** `Router(config)# scripting tcl low-memory 33117513`	(Optional) Specifies a low water memory mark for free memory for Tcl-based applications. The memory threshold can be set anywhere between 0-4294967295 bytes. **Note** If minimum free RAM drops below this threshold, TCL aborts the current script. This prevents the Tcl interpreter from allocating too much RAM and crashing the router.
Step 6	**exit** **Example:** `Router(config)# exit`	(Optional) Exits global configuration mode and returns to privileged EXEC mode.
Step 7	**tclsh** **Example:** `Router# tclsh`	Enables the interactive Tcl shell and enters Tcl configuration mode.
Step 8	Enter the required Tcl command language syntax. **Example:** `Router(tcl)# proc get_bri {}`	Commands entered in Tcl configuration mode are sent first to the interactive Tcl interpreter. If the command is not a valid Tcl command, it is then sent to the CLI parser.

	Command or Action	Purpose
Step 9	**ios_config "***cmd***" "***cmd-option***"** **Example:** `Router(tcl)# ios_config "interface Ethernet 2/0" "no keepalive"`	(Optional) Modifies the router configuration using a Tcl script by specifying the Tcl command **ios_config** with CLI commands and options. All arguments and submode commands must be entered on the same line as the CLI configuration command. • In this example, the first argument in quotes configures an Ethernet interface and enters interface configuration mode. The second argument in quotes sets the keepalive option. If these two CLI statements were entered on separate Tcl command lines, the configuration would not work.
Step 10	**exec "***exec-cmd***"** **Example:** `Router(tcl)# exec "show interfaces"`	(Optional) Executes Cisco IOS CLI EXEC mode commands from a Tcl script by specifying the Tcl command **exec** with the CLI commands. • In this example, interface information for the router is displayed.
Step 11	**exit** **Example:** `Router(tcl)# exit`	Exits Tcl configuration mode and returns to privileged EXEC mode.

Examples

The following sample partial output shows information about Ethernet interface 0 on the router. The **show interfaces** command has been executed from Tcl configuration mode.

```
Router# tclsh
Router(tcl)# exec "show interfaces"

Ethernet 0 is up, line protocol is up
  Hardware is MCI Ethernet, address is 0000.0c00.750c (bia 0000.0c00.750c)
  Internet address is 10.108.28.8, subnet mask is 255.255.255.0
  MTU 1500 bytes, BW 10000 Kbit, DLY 100000 usec, rely 255/255, load 1/255
  Encapsulation ARPA, loopback not set, keepalive set (10 sec)
  ARP type: ARPA, ARP Timeout 4:00:00
  Last input 0:00:00, output 0:00:00, output hang never
  Last clearing of "show interface" counters 0:00:00
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  Five minute input rate 0 bits/sec, 0 packets/sec
  Five minute output rate 2000 bits/sec, 4 packets/sec
     1127576 packets input, 447251251 bytes, 0 no buffer
     Received 354125 broadcasts, 0 runts, 0 giants, 57186* throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     5332142 packets output, 496316039 bytes, 0 underruns
     0 output errors, 432 collisions, 0 interface resets, 0 restarts
.
.
.
```

Troubleshooting Tips

Use the Tcl **puts** command in a Tcl script to trace command execution.

Using the Tcl Shell to Access SNMP MIB Objects

Perform this optional task to enable the interactive Tcl shell and enter Tcl commands to perform actions on MIB objects.

SNMP MIB Custom Extensions in the Tcl Shell

The Cisco IOS implementation of the Tcl shell contains some custom command extensions for SNMP MIB object access. These extensions operate only under Tcl configuration mode. Table 3 displays these command extensions.

Table 3 Cisco IOS Custom Tcl Command Extensions for SNMP MIB Access

Command	Description
snmp_getbulk	Retrieves a large section of a MIB table. This command is similar to the SNMP **getbulk** command. The syntax is in the following format: **snmp_getbulk** *community-string non-repeaters max-repetitions oid* [*oid2 oid3...*] • Use the *community-string* argument to specify the SNMP community from which the objects will be retrieved. • Use the *non-repeaters* argument to specify the number of objects that can be retrieved with a get-next operation. • Use the *max-repetitions* argument to specify the maximum number of get-next operations to attempt while trying to retrieve the remaining objects. • Use the *oid* argument to specify the object ID(s) to retrieve.
snmp_getid	Retrieves the following variables from the SNMP entity on the router: • sysDescr.0 • sysObjectID.0 • sysUpTime.0 • sysContact.0 • sysName.0 • sysLocation.0 This command is similar to the SNMP **getid** command. The syntax is in the following format: **snmp_getid** *community-string*
snmp_getnext	Retrieves a set of individual variables from the SNMP entity on the router. This command is similar to the SNMP **getnext** command. The syntax is in the following format: **snmp_getnext** *community-string oid* [*oid2 oid3...*]

Table 3 Cisco IOS Custom Tcl Command Extensions for SNMP MIB Access (continued)

Command	Description
snmp_getone	Retrieves a set of individual variables from the SNMP entity on the router. This command is similar to the SNMP **getone** command. The syntax is in the following format: **snmp_getone** *community-string oid* [*oid2 oid3...*]
snmp_setany	Retrieves the current values of the specified variables and then performs a set request on the variables. This command is similar to the SNMP **setany** command. The syntax is in the following format: **snmp_setany** *community-string oid type val* [*oid2 type2 val2...*] • Use the *type* argument to specify the type of object to retrieve. The *type* can be one of the following: – **-i**—Integer. A 32-bit number used to specify a numbered type within the context of a managed object. For example, to set the operational status of a router interface, 1 represents up and 2 represents down. – **-u**—Unsigned32. A 32-bit number used to represent decimal values in the range from 0 to $2^{32} - 1$ inclusive. – **-c**—Counter32. A 32-bit number with a minimum value of 0 and a maximum value of $2^{32} - 1$. When the maximum value is reached, the counter resets to 0 and starts again. – **-g**—Gauge. A 32-bit number with a minimum value of 0 and a maximum value of $2^{32} - 1$. The number can increase or decrease at will. For example, the interface speed on a router is measured using a gauge object type. – **-o**—Octet string. An octet string—in hex notation—used to represent physical addresses. – **-d**—Display string. An octet string—in text notation—used to represent text strings. – **-ipv4**—IP version 4 address. – **-oid**—Object ID. • Use the *val* argument to specify the value of object ID(s) to retrieve.

Prerequisites

The SNMP community configuration must exist in the running configuration of the router.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **scripting tcl encdir** *location-url*
4. **scripting tcl init** *init-url*
5. **exit**

6. **tclsh**
7. Enter any required Tcl command language syntax.
8. **snmp_getbulk** *community-string non-repeaters max-repetitions oid* [*oid2 oid3...*]
9. **snmp_getid** *community-string*
10. **snmp_getnext** *community-string oid* [*oid2 oid3...*]
11. **snmp_getone** *community-string oid* [*oid2 oid3...*]
12. **snmp_setany** *community-string oid type val* [*oid2 type2 val2...*]
13. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	(Optional) Enters global configuration mode. • Perform Step 2 through Step 5 Perform Step 2 through Step 5 if you are using encoding files, an initialization script, or both.
Step 3	**scripting tcl encdir** *location-url* **Example:** `Router(config)# scripting tcl encdir tftp://10.18.117.23/enctcl/`	(Optional) Specifies the default location of external encoding files used by the Tcl **encoding** command.
Step 4	**scripting tcl init** *init-url* **Example:** `Router(config)# scripting tcl init ftp://user:password@172.17.40.3/tclscript/initfiles3.tcl`	(Optional) Specifies an initialization script to run when the Tcl shell is enabled.
Step 5	**exit** **Example:** `Router(config)# exit`	(Optional) Exits global configuration mode and returns to privileged EXEC mode.
Step 6	**tclsh** **Example:** `Router# tclsh`	Enables the interactive Tcl shell and enters Tcl configuration mode.
Step 7	Enter the required Tcl command language syntax. **Example:** `Router(tcl)# proc get_bri {}`	Commands entered in Tcl configuration mode are sent first to the interactive Tcl interpreter. If the command is not a valid Tcl command, it is sent to the CLI parser.

	Command or Action	Purpose
Step 8	**snmp_getbulk** *community-string non-repeaters max-repetitions oid* [*oid2 oid3...*] **Example:** `Router(tcl)# snmp_getbulk public 1 3 1.3.6.1.2.1.1.1 1.3.6.1.2.1.10.18.8.1.1`	(Optional) Retrieves a large section of a MIB table. • Use the *community-string* argument to specify the SNMP community from which the objects will be retrieved. • Use the *non-repeaters* argument to specify the number of objects that can be retrieved with a get-next operation. • Use the *max-repetitions* argument to specify the maximum number of get-next operations to attempt while trying to retrieve the remaining objects. • Use the *oid* argument to specify the object ID(s) to retrieve.
Step 9	**snmp_getid** *community-string* **Example:** `Router(tcl)# snmp_getid private`	(Optional) Retrieves the following variables from the SNMP entity on the router: sysDesrc.0, sysObjectID.0, sysUpTime.0, sysContact.0, sysName.0, and sysLocation.0. • Use the *community-string* argument to specify the SNMP community from which the objects will be retrieved.
Step 10	**snmp_getnext** *community-string oid* [*oid2 oid3...*] **Example:** `Router(tcl)# snmp_getnext public 1.3.6.1.2.1.1.1.0 1.3.6.1.2.1.1.2.0`	(Optional) Retrieves a set of individual variables from a MIB table. • Use the *community-string* argument to specify the SNMP community from which the objects will be retrieved. • Use the *oid* argument to specify the object ID(s) to retrieve.
Step 11	**snmp_getone** *community-string oid* [*oid2 oid3...*] **Example:** `Router(tcl)# snmp_getone public 1.3.6.1.2.1.1.1.0 1.3.6.1.2.1.1.2.0`	(Optional) Retrieves a set of individual variables from a MIB table. • Use the *community-string* argument to specify the SNMP community from which the objects will be retrieved. • Use the *oid* argument to specify the object ID(s) to retrieve.

	Command or Action	Purpose
Step 12	**snmp_setany** *community-string oid type val* [*oid2 type2 val2...*] **Example:** `Router(tcl)# snmp_setany private 1.3.6.1.2.1.1.5.0 -d TCL-SNMP_TEST`	(Optional) Retrieves current values of specified variables from a MIB table and then performs a set request on the variables. • Use the *community-string* argument to specify the SNMP community from which the values of objects will be retrieved and then set. • Use the *oid* argument to specify the object ID(s) to retrieve and set. • Use the *type* argument to specify the type of object to retrieve and set. • Use the *val* argument to specify the value of the object to be retrieved and then set.
Step 13	**exit** **Example:** `Router(tcl)# exit`	Exits Tcl configuration mode and returns to privileged EXEC mode.

Troubleshooting Tips

Use the Tcl **puts** command in a Tcl script to trace command execution.

Running Predefined Tcl Scripts

Perform this optional task to run a predefined Tcl script in Cisco IOS software.

Prerequisites

Before performing this task, you must create a Tcl script that can run on Cisco IOS software. The Tcl script may be transferred to internal flash memory using any file system that the Cisco IOS file system (IFS) supports, including TFTP, FTP, and rcp. The Tcl script may also be sourced from a remote location.

SUMMARY STEPS

1. **enable**
2. **tclsh**
3. Enter the Tcl source command with the filename and path.
4. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**tclsh** **Example:** `Router# tclsh`	Enables the interactive Tcl shell and enters Tcl configuration mode.
Step 3	Enter the Tcl source command with the filename and path. **Example:** `Router(tcl)# source slot0:test.tcl`	Commands entered in Tcl configuration mode are sent first to the interactive Tcl interpreter. If the command is not a valid Tcl command, it is then sent to the CLI parser.
Step 4	**exit** **Example:** `Router(tcl)# exit`	Exits Tcl configuration mode and returns to privileged EXEC mode.

Configuration Examples for Cisco IOS Scripting with Tcl

This section provides the following configuration examples:

Tcl Script Using the show interfaces Command: Example

Using the Tcl regular expression engine, scripts can filter specific information from **show** commands and present it in a custom format. The following is an example of filtering the **show interfaces** command output and creating a comma-separated list of BRI interfaces on the router:

```
tclsh
proc get_bri {} {
    set check ""
    set int_out [exec "show interfaces"]
    foreach int [regexp -all -line -inline "(^BRI\[0-9]/\[0-9])" $int_out] {
        if {![string equal $check $int]} {
            if {[info exists bri_out]} {
                append bri_out "," $int
             } else {
                 set bri_out $int
             }
             set check $int
        }
    }
```

```
    return $bri_out
}
```

Tcl Script for SMTP Support: Example

The following Tcl script is useful for sending e-mail messages from a router.

```
##
## Place required comments here!!!
##

package provide sendmail 2.0

# Sendmail procedure for Support

namespace eval ::sendmail {

    namespace export initialize configure sendmessage sendfile

    array set ::sendmail::sendmail {
        smtphost    mailhub
        from        ""
        friendly    ""
    }

    proc configure {} {}

    proc initialize {smtphost from friendly} {

        variable sendmail

        if {[string length $smtphost]} then {
            set sendmail(smtphost) $smtphost
        }
        if {[string length $from]} then {
            set sendmail(from) $from
        }
        if {[string length $friendly]} then {
            set sendmail(friendly) $friendly
        }
    }

    proc sendmessage {toList subject body {tcl_trace 0}} {

        variable sendmail

        set smtphost $sendmail(smtphost)
        set from $sendmail(from)
        set friendly $sendmail(friendly)

        if {$trace} then {
            puts stdout "Connecting to $smtphost:25"
        }

        set sockid [socket $smtphost 25]

## DEBUG
set status [catch {

        puts $sockid "HELO $smtphost"
        flush $sockid
        set result [gets $sockid]
```

```
        if {$trace} then {
            puts stdout "HELO $smtphost\n\t$result"
        }

        puts $sockid "MAIL From:<$from>"
        flush $sockid
        set result [gets $sockid]

        if {$trace} then {
            puts stdout "MAIL From:<$from>\n\t$result"
        }

        foreach to $toList {
            puts $sockid "RCPT To:<$to>"
            flush $sockid
        }

        set result [gets $sockid]
        if {$trace} then {
            puts stdout "RCPT To:<$to>\n\t$result"
        }

        puts  $sockid "DATA "
        flush $sockid
        set result [gets  $sockid]

        if {$trace} then {
            puts stdout "DATA \n\t$result"
        }

        puts  $sockid "From: $friendly <$from>"
        foreach to $toList {
            puts $sockid "To:<$to>"
        }
        puts  $sockid "Subject: $subject"
        puts  $sockid "\n"

        foreach line [split $body "\n"] {
            puts  $sockid " $line"
        }

        puts  $sockid "."
        puts  $sockid "QUIT"
        flush $sockid
        set result [gets  $sockid]

        if {$trace} then {
            puts stdout "QUIT\n\t$result"
        }
} result]

        catch {close $sockid }
        if {$status} then {
          return -code error $result
        }
        return
    }

    proc sendfile {toList filename subject {tcl_trace 0}} {

        set fd [open $filename r]

        sendmessage $toList $subject [read $fd] $trace
```

```
        return
    }

}
```

Tcl Script for SNMP MIB Access: Examples

Using the Tcl shell, Tcl commands can perform actions on MIBs. The following example shows how to set up the community access strings to permit access to SNMP. Public access is read-only, but private access is read-write. The following example shows how to retrieve a large section of a table at once using the **snmp_getbulk** Tcl command extension.

Two arguments, *non-repeaters* and *max-repetitions*, must be set when an **snmp_getbulk** command is issued. The *non-repeaters* argument specifies that the first N objects are to be retrieved with a simple **snmp_getnext** operation. The *max-repetitions* argument specifies that up to M **snmp_getnext** operations are to be attempted to retrieve the remaining objects.

In this example, three bindings—sysUpTime (1.3.6.1.2.1.1.2.0), ifDescr (1.3.6.1.2.1.2.2.1.2), and ifType (1.3.6.1.2.1.2.2.1.3)—are used. The total number of variable bindings requested is given by the formula N + (M * R), where N is the number of non-repeaters (in this example 1), M is the max-repetitions (in this example 5), and R is the number of request objects (in this case 2, ifDescr and ifType). Using the formula, 1 + (5 * 2) equals 11; and this is the total number of variable bindings that can be retrieved by this **snmp_getbulk** request command.

Sample results for the individual variables include a retrieved value of sysUpTime.0 being 1336090, where the unit is in milliseconds. The retrieved value of ifDescr.1 (the first interface description) is FastEthernet0/0, and the retrieved value of ifType.1 (the first interface type) is 6, which corresponds to the ethernetCsmacd type.

```
snmp-server community public RO
snmp-server community private RW
tclsh
  snmp_getbulk public 1 5 1.3.6.1.2.1.1.2.0  1.3.6.1.2.1.2.2.1.2  1.3.6.1.2.1.2.2.1.3
  {<obj oid='sysUpTime.0' val='1336090'/>}
  {<obj oid='ifDescr.1' val='FastEthernet0/0'/>}
  {<obj oid='ifType.1' val='6'/>}
  {<obj oid='ifDescr.2' val='FastEthernet1/0'/>}
  {<obj oid='ifType.2' val='6'/>}
  {<obj oid='ifDescr.3' val='Ethernet2/0'/>}
  {<obj oid='ifType.3' val='6'/>}
  {<obj oid='ifDescr.4' val='Ethernet2/1'/>}
  {<obj oid='ifType.4' val='6'/>}
  {<obj oid='ifDescr.5' val='Ethernet2/2'/>}
  {<obj oid='ifType.5' val='6'/>}
```

The following example shows how to retrieve the sysDescr.0, sysObjectID.0, sysUpTime.0, sysContact.0, sysName.0, and sysLocation.0 variables—in this example shown as system.1.0, system.2.0, system.3.0, system.4.0, system.5.0, and system.6.0—from the SNMP entity on the router using the **snmp_getid** Tcl command extension.

```
tclsh
 snmp_getid public
 {<obj oid='system.1.0' val='Cisco Internetwork Operating System Software
 Cisco IOS(tm) 7200 Software (C7200-IK9S-M), Experimental Version 12.3(20030507:225511)
 [geotpi2itd1 124]
 Copyright (c) 1986-2003 by Cisco Systems, Inc.
 Compiled Wed 21-May-03 16:16 by engineer'/>}
 {<obj oid='system.2.0' val='products.223'/>}
 {<obj oid='sysUpTime.0' val='6664317'/>}
 {<obj oid='system.4.0' val='1-800-553-2447 - phone the TAC'/>}
```

```
 {<obj oid='system.5.0' val='c7200.myCompany.com'/>}
 {<obj oid='system.6.0' val='Bldg 24, San Jose, CA'/>}
```

The following example shows how to retrieve a set of individual variables from the SNMP entity on the router using the **snmp_getnext** Tcl command extension:

```
snmp_getnext public 1.3.6.1.2.1.1.1.0 1.3.6.1.2.1.1.2.0
 {<obj oid='system.2.0' val='products.223'/>}
 {<obj oid='sysUpTime.0' val='6683320'/>}
```

The following example shows how to retrieve a set of individual variables from the SNMP entity on the router using the **snmp_getone** Tcl command extension:

```
snmp_getone public 1.3.6.1.2.1.1.1.0 1.3.6.1.2.1.1.2.0
 {<obj oid='system.1.0' val='Cisco Internetwork Operating System Software
 Cisco IOS(tm) 7200 Software (C7200-IK9S-M), Experimental Version 12.3(20030507:225511)
 [geotpi2itd1 124]
 Copyright (c) 1986-2003 by Cisco Systems, Inc.
 Compiled Wed 21-May-03 16:16 by engineer'/>}
 {<obj oid='system.2.0' val='products.223'/>}
```

The following example shows how to change something in the configuration of the router using the **snmp_setany** Tcl command extension. In this example, the hostname of the router is changed to TCLSNMP-HOST.

```
tclsh
 snmp_setany private 1.3.6.1.2.1.1.5.0 -d TCLSNMP-HOST
 {<obj oid='system.5.0' val='TCLSNMP-HOST'/>}
```

Additional References

The following sections provide references related to the Cisco IOS Scripting with Tcl feature.

Related Documents

Related Topic	Document Title
Embedded Syslog Manager	Embedded Syslog Manager module
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*

Standards

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

MIBs

MIBs	MIBs Link
No new or modified MIBs are supported by this feature, and support for existing MIBs has not been modified by this feature.	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

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

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/techsupport

Command Reference

The following commands are introduced or modified in the feature or features documented in this module. For information about these commands, see the *Cisco IOS Network Management Command Reference* at http://www.cisco.com/en/US/docs/ios/netmgmt/command/reference/nm_book.html. For information about all Cisco IOS commands, go to the Command Lookup Tool at http://tools.cisco.com/Support/CLILookup or to the *Cisco IOS Master Commands List*.

- **scripting tcl encdir**
- **scripting tcl init**
- **tclsh**

Feature Information for Cisco IOS Scripting with Tcl

Table 4 lists the release history for this feature.

Not all commands may be available in your Cisco IOS software release. For release information about a specific command, see the command reference documentation.

Use Cisco Feature Navigator to find information about platform support and software image support. Cisco Feature Navigator enables you to determine which Cisco IOS and Catalyst OS software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Note Table 4 lists only the Cisco IOS software release that introduced support for a given feature in a given Cisco IOS software release train. Unless noted otherwise, subsequent releases of that Cisco IOS software release train also support that feature.

Table 4 Feature Information for Cisco IOS Scripting with Tcl

Feature Name	Releases	Feature Information
Cisco IOS Scripting with Tcl	12.3(2)T 12.3(7)T 12.2(25)S 12.2(33)SXH 12.2(33)SRC 12.2(33)SB	The Cisco IOS Scripting with Tcl feature provides the ability to run Tool Command Language (Tcl) version 8.3.4 commands from the Cisco IOS command-line interface (CLI). The following commands were introduced or modified: **scripting tcl encdir**, **scripting tcl init**, **scripting tcl low-memory**, **tclquit**, **tclsh**.
Tcl SNMP MIB Access	12.3(7)T 12.2(25)S 12.2(33)SXH 12.2(33)SRC 12.2(33)SB	The Tcl SNMP MIB Access feature introduces a set of UNIX-like SNMP commands to make access to Simple Network Management Protocol (SNMP) MIB objects easier. The following section has more information: • Using the Tcl Shell to Access SNMP MIB Objects, page 7.

Glossary

ESM—Embedded Syslog Manager.

IVR—Interactive Voice Response.

MIB—Management Information Base.

SNMP—Simple Network Management Protocol.

Tcl—Tool Command Language.

Note See *Internetworking Terms and Acronyms* for terms not included in this glossary.

Cisco Networking Services (CNS)

Cisco Networking Services

First Published: November 20, 2006
Last Updated: April 26, 2009

The Cisco Networking Services (CNS) feature is a collection of services that can provide remote event-driven configuring of Cisco IOS networking devices and remote execution of some command-line interface (CLI) commands.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the "Feature Information for CNS" section on page 48.

Use Cisco Feature Navigator to find information about platform support and Cisco IOS and Catalyst OS software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Contents

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

Prerequisites for CNS

- Configure the remote router to support the CNS configuration agent and the CNS event agent.
- Configure a transport protocol on the remote router that is compatible with the remote router's external interface. Table 1 lists the supported transport protocols that can be used depending on the router interface.
- Create the configuration template in the CNS configuration-engine provisioning database. (This task is best done by a senior network designer.)

Table 1 Router Interface and Transport Protocols Required by CNS Services

	Transport Protocol		
Router Interface	**SLARP**	**ATM InARP**	**PPP (IPCP)**
T1	Yes	Yes	Yes
ADSL	No	Yes	Yes
Serial	Yes	No	Yes

CNS Image Agent

- Determine where to store the Cisco IOS images on a file server to make the image available to many other networking devices. If the CNS Event Bus is to be used to store and distribute the images, the CNS event agent must be configured.
- Set up a file server to enable the networking devices to download the new images. Protocols such as TFTP, HTTP, HTTPS, and rcp can be used.
- Determine how to handle error messages generated by CNS image agent operations. Error messages can be sent to the CNS Event Bus or an HTTP or HTTPS URL.

Restrictions for CNS

CNS Configuration Engine

- The CNS configuration engine must be the Cisco Intelligence Engine 2100 (Cisco IE2100) series and must be running software version 1.3.
- The configuration engine must have access to an information database of attributes for building a configuration. This database can reside on the Cisco IE2100 itself.
- Configuration templates must be prepared on the CNS configuration engine before installation of the remote router.
- The user of CNS Flow-Through Provisioning and the CNS configuration engine must be familiar with designing network topologies, designing configuration templates, and using the CNS configuration engine.

CNS Image Agent

During automated image loading operations you must try to prevent the Cisco IOS device from losing connectivity with the file server that is providing the image. Image reloading is subject to memory issues and connection issues. Boot options must also be configured to allow the Cisco IOS device to boot another image if the first image reload fails. For more details see the "Managing Configuration Files" module of the *Cisco IOS Configuration Fundamentals Configuration Guide*.

CNS Frame Relay Zero Touch

The CNS Frame Relay Zero Touch solution does not support switched virtual circuits (SVCs).

The Frame Relay zero touch solution does not support IP over PPP over Frame Relay because routing to an interface (or subinterface) that supports IP over PPP over Frame Relay is not possible.

Command Scheduler

The EXEC CLI specified in a Command Scheduler policy list must neither generate a prompt nor can it be terminated using keystrokes. Command Scheduler is designed as a fully automated facility, and no manual intervention is permitted.

Remote Router

- The remote router must run a Cisco IOS image that supports the CNS configuration agent and CNS event agent.
- Ports must be prepared on the remote router for connection to the network.
- You must ensure that the remote router is configured using Cisco Configuration Express.

Information About CNS

To configure CNS, you should understand the following concepts:

CNS

CNS is a foundation technology for linking users to networking services and provides the infrastructure for the automated configuration of large numbers of network devices. Many IP networks are complex with many devices, and each device must currently be configured individually. When standard configurations do not exist or have been modified, the time involved in initial installation and subsequent upgrading is considerable. The volume of smaller, more standardized, customer networks is also growing faster than the number of available network engineers. Internet service providers (ISPs) now need a method for sending out partial configurations to introduce new services. To address all these issues, CNS has been designed to provide "plug-and-play" network services using a central directory service and distributed agents. CNS features include CNS configuration and event agents and a Flow-Through Provisioning structure. The configuration and event agents use a CNS configuration engine to provide methods for automating initial Cisco IOS device configurations, incremental configurations, and synchronized configuration updates, and the configuration engine reports the status of the configuration load as an event to which a network monitoring or workflow application can subscribe. The CNS Flow-Through Provisioning uses the CNS configuration and event agents to provide an automated workflow, eliminating the need for an on-site technician.

CNS Configuration Agent

The CNS configuration agent is involved in the initial configuration and subsequent partial configurations on a Cisco IOS device. To activate the CNS configuration agent, enter any of the **cns config** CLI commands.

Initial CNS Configuration

When a routing device first comes up, it connects to the configuration server component of the CNS configuration agent by establishing a TCP connection through the use of the **cns config initial** command, a standard CLI command. The device issues a request and identifies itself by providing a unique configuration ID to the configuration server.

When the CNS web server receives a request for a configuration file, it invokes the Java servlet and executes the corresponding embedded code. The embedded code directs the CNS web server to access the directory server and file system to read the configuration reference for this device (configuration ID) and template. The Configuration Agent prepares an instantiated configuration file by substituting all the parameter values specified in the template with valid values for this device. The configuration server forwards the configuration file to the CNS web server for transmission to the routing device.

The CNS configuration agent accepts the configuration file from the CNS web server, performs XML parsing, checks syntax (optional), and loads the configuration file. The routing device reports the status of the configuration load as an event to which a network monitoring or workflow application can subscribe.

For more details on using the Cisco CNS configuration engine to automatically install the initial CNS configuration, see the *Cisco CNS Configuration Engine Administrator's Guide* at http://www.cisco.com/univercd/cc/td/doc/product/rtrmgmt/cns/ce/rel13/ag13/index.htm.

Incremental CNS Configuration

Once the network is up and running, new services can be added using the CNS configuration agent. Incremental (partial) configurations can be sent to routing devices. The actual configuration can be sent as an event payload by way of the event gateway (push operation) or as a signal event that triggers the device to initiate a pull operation.

The routing device can check the syntax of the configuration before applying it. If the syntax is correct, the routing device applies the incremental configuration and publishes an event that signals success to the configuration server. If the device fails to apply the incremental configuration, it publishes an event that indicates an error.

Once the routing device has applied the incremental configuration, it can write the configuration to NVRAM or wait until signaled to do so.

Synchronized Configuration

When a routing device receives a configuration, the device has the option to defer application of the configuration upon receipt of a write-signal event. The CNS Configuration Agent feature allows the device configuration to be synchronized with other dependent network activities.

CNS Config Retrieve Enhancement with Retry and Interval

The Cisco Networking Services (CNS) Config Retrieve Enhancement with Retry and Interval feature adds new functionality to the **cns config retrieve** command enabling you to specify the retry interval and an amount of time in seconds to wait before attempting to retrieve a configuration from a trusted server.

CNS EXEC Agent

The CNS EXEC agent allows a remote application to execute an EXEC mode CLI command on a Cisco IOS device by sending an event message that contains the command. A restricted set of EXEC **show** commands is supported.

CNS Event Agent

Although other CNS agents may be configured, no other CNS agents are operational until the **cns event** command is entered because the CNS event agent provides a transport connection to the CNS event bus for all other CNS agents. The other CNS agents use the connection to the CNS event bus to send and receive messages. The CNS event agent does not read or modify the messages.

CNS Image Agent

Administrators maintaining large networks of Cisco IOS devices need an automated mechanism to load image files onto large numbers of remote devices. Existing network management applications are useful to determine which images to run and how to manage images received from the Cisco online software center. Other image distribution solutions do not scale to cover thousands of devices and cannot

distribute images to devices behind a firewall or using Network Address Translation (NAT). The CNS image agent enables the managed device to initiate a network connection and request an image download allowing devices using NAT, or behind firewalls, to access the image server.

The CNS image agent can be configured to use the CNS Event Bus. To use the CNS Event Bus, the CNS event agent must be enabled and connected to the CNS event gateway in the CNS Configuration Engine. The CNS image agent can also use an HTTP server that understands the CNS image agent protocol. Deployment of CNS image agent operations can use both the CNS Event Bus and an HTTP server.

CNS Results Messages

When a partial configuration has been received by the router, each line of the configuration will be applied in the same order as it was received. If the Cisco IOS parser has an error with one of the lines of the configuration, then all the configuration up to this point will be applied to the router, but none of the configuration beyond the error will be applied. If an error occurs, the **cns config partial** command will retry until the configuration successfully completes. In the pull mode, the command will not retry after an error. By default, NVRAM will be updated except when the **no-persist** keyword is configured.

A message will be published on the CNS event bus after the partial configuration is complete. The CNS event bus will display one of the following status messages:

- cisco.mgmt.cns.config.complete—CNS configuration agent successfully applied the partial configuration.
- cisco.mgmt.cns.config.warning—CNS configuration agent fully applied the partial configuration, but encountered possible semantic errors.
- cisco.mgmt.cns.config.failure(CLI syntax)—CNS configuration agent encountered a command line interface (CLI) syntax error and was not able to apply the partial configuration.
- cisco.mgmt.cns.config.failure(CLI semantic)—CNS configuration agent encountered a CLI semantic error and was not able to apply the partial configuration.

In Cisco IOS Releases 12.4(4)T, 12.2 (33)SRA, and later releases, a second message is sent to the subject "cisco.cns.config.results" in addition to the appropriate message above. The second message contains both overall and line-by-line information about the configuration that was sent and the result of the action requested in the original message. If the action requested was to apply the configuration, then the information in the results message is semantic in nature. If the action requested was to check syntax only, then the information in the results message is syntactical in nature.

CNS Message Formats

SOAP Message Format

Using the Service-Oriented Access Protocol (SOAP) protocol provides a way to format the layout of CNS messages in a consistent manner. SOAP is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment. SOAP uses extensible markup language (XML) technologies to define an extensible messaging framework that provides a message format that can be exchanged over a variety of underlying protocols.

Within the SOAP message structure, there is a security header that enables CNS notification messages to authenticate user credentials.

CNS messages are classified into three message types: request, response and notification. The formats of these three message types are defined below.

Request Message

The following is the format of a CNS request message to the Cisco IOS device:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope">
  <SOAP:Header>
    <wsse:Security xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/04/secext"
SOAP:mustUnderstand="0">
      <wsse:usernameToken>
        <wsse:Username>john</wsse:Username>
        <wsse:Password>cisco</wsse:Password>
      </wsse:usernameToken>
    </wsse:Security>
    <cns:cnsHeader version="1.0" xmlns:cns="http://www.cisco.com/management/cns/envelope">
      <cns:Agent>CNS_CONFIG</cns:Agent>
      <cns:Request>
        <cns:correlationID>IDENTIFIER</cns:correlationID>
        <cns:ReplyTo>
          <cns:URL>http://10.1.36.9:80/cns/ResToServer</cns:URL>
        </cns:ReplyTo>
      </cns:Request>
      <cns:Time>2003-04-23T20:27:19.847Z</cns:Time>
    </cns:cnsHeader>
  </SOAP:Header>
  <SOAP:Body xmlns="http://www.cisco.com/management/cns/config">
    <config-event config-action="read" no-syntax-check="TRUE">
      <config-data>
        <config-id>AAA</config-id>
        <cli>access-list 1 permit any</cli>
      </config-data>
    </config-event>
  </SOAP:Body>
</SOAP:Envelope>
```

Note The ReplyTo field is optional. In the absence of the ReplyTo field, the response to the request will be sent to the destination where the request originated. The body portion of this message contains the payload and is processed by the CNS agent mentioned in the Agent field.

Response Message

The following is the format of a CNS response message from the Cisco IOS device as a response to a request:

```
?xml version="1.0" encoding="UTF-8"?
SOAP:Envelope xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope"
SOAP:Header
wsse:Security xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/04/secext"
SOAP:mustUnderstand="true"
wsse:UsernameToken
wsse:Username infysj-7204-8 /wsse:Username
wsse:Password NTM3NTg2NzIzOTg2MTk2MjgzNQ==/wsse:Password
/wsse:UsernameToken /wsse:Security
CNS:cnsHeader Version="2.0" xmlns:CNS="http://www.cisco.com/management/cns/envelope"
CNS:Agent CNS_CONFIG /CNS:Agent
CNS:Response
CNS:correlationID IDENTIFIER /CNS:correlationID
/CNS:Response
CNS:Time 2005-06-23T16:27:36.185Z /CNS:Time
/CNS:cnsHeader
/SOAP:Header
SOAP:Body xmlns="http://www.cisco.com/management/cns/config"
```

```
config-success config-id AAA /config-id /config-success
/SOAP:Body
/SOAP:Envelope
```

Note The value of CorrelationId is echoed from the corresponding request message.

The body portion of this message contains the response from the Cisco IOS device to a request. If the request is successfully processed, the body portion contains the value of the response put in by the agent that processed the request. If the request cannot be successfully processed, then the body portion will contain an error response.

Notification Message

The following is the format of a CNS notification message sent from the Cisco IOS device:

```
?xml version="1.0" encoding="UTF-8"?
SOAP:Envelope xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope"
SOAP:Header
wsse:Security xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/04/secext"
SOAP:mustUnderstand="true"
wsse:UsernameToken
wsse:Username dvlpr-7200-2 /wsse:Username
wsse:Password /wsse:Password
/wsse:UsernameToken
/wsse:Security
CNS:cnsHeader version="2.0" xmlns:CNS="http://www.cisco.com/management/cns/envelope"
CNS:Agent CNS_CONFIG_CHANGE/CNS:Agent
CNS:Notify /CNS:Notify
CNS:Time 2006-01-09T18:57:08.441Z/CNS:Time
/CNS:cnsHeader
/SOAP:Header
SOAP:Body xmlns="http://www.cisco.com/management/cns/config-change"
configChanged version="1.1" sessionData="complete"
sequence lastReset="2005-12-11T20:18:39.673Z" 7 /sequence
changeInfo
user/user
async port con_0 /port /async
when
absoluteTime 2006-01-09T18:57:07.973Z /absoluteTime
/when
/changeInfo
changeData
changeItem
context /context
enteredCommand
cli access-list 2 permit any /cli
/enteredCommand
oldConfigState
cli access-list 1 permit any /cli
/oldConfigState
newConfigState
cli access-list 1 permit any /cli
cli access-list 2 permit any /cli
/newConfigState
/changeItem
/changeData
/configChanged
/SOAP:Body
/SOAP:Envelope
```

A notification message is sent from the Cisco IOS device without a corresponding request message when a configuration change is made. The body of the message contains the payload of the notification and it may also contain error information. If the request message sent to the Cisco IOS device fails in XML parsing and the CorrelationId field cannot be parsed, then an error notification message will be sent instead of an error response.

Error Reporting

Error is reported in the body of the response or a notification message in the SOAP Fault element. The following is the format for reporting errors.

```
?xml version="1.0" encoding="UTF-8"?
SOAP:Envelope xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope"
SOAP:Header
wsse:Security xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/04/secext"
SOAP:mustUnderstand="true"
wsse:UsernameToken
wsse:Username dvlpr-7200-2 /wsse:Username
wsse:Password /wsse:Password
/wsse:UsernameToken
/wsse:Security
CNS:cnsHeader version="2.0" xmlns:CNS="http://www.cisco.com/management/cns/envelope"
CNS:Agent CNS_CONFIG /CNS:Agent
CNS:Response
CNS:correlationID SOAP_IDENTIFIER /CNS:correlationID
/CNS:Response
CNS:Time 2006-01-09T19:10:10.009Z /CNS:Time
/CNS:cnsHeader
/SOAP:Header
SOAP:Body xmlns="http://www.cisco.com/management/cns/config"
SOAP:Detail
config-failure
config-id AAA /config-id
error-info
line-number 1 /line-number
error-message CNS_INVALID_CLI_CMD /error-message
/error-info
/config-failure
/SOAP:Detail
/SOAP:Fault
/SOAP:Body
/SOAP:Envelope
```

CNS Security Enhancement

Before the introduction of the CNS Security Enhancement feature, the CNS message format did not support security. Using the new CNS SOAP message structure, the username and password are authenticated.

If authentication, authorization, and accounting (AAA) is configured, then CNS SOAP messages will be authenticated with AAA. If AAA is not configured, there will be no authentication. For backward compatibility, CNS will support the existing non-SOAP message format and will respond accordingly without security.

The **cns aaa authentication** command is required to turn on CNS Security Enhancement. This command determines whether the CNS messages are using AAA security or not. If the **cns aaa authentication** command is configured, then all incoming SOAP messages into the device are authenticated by AAA.

CNS Interactive CLI

The CNS Interactive CLI feature provides a XML interface that allows you to send interactive commands to a router, such as commands that generate prompts for user input. A benefit of this feature is that interactive commands can be aborted before they have been fully processed. For example, for commands that generate a significant amount of output, the XML interface can be customized to limit the size of the output or the length of time allowed for the output to accumulate. The capability to use a programmable interface to abort a command before its normal termination (similar to manually aborting a command) can greatly increase the efficiency of diagnostic applications that might use this functionality. The new XML interface also allows for multiple commands to be processed in a single session. The response for each command is packaged together and sent in a single response event.

CNS IDs

The CNS ID is a text string that is used exclusively with a particular CNS agent. The CNS ID is used by the CNS agent to identify itself to the server application with which it communicates. For example, the CNS configuration agent will include the configuration ID when communicating between the networking device and the configuration server. The configuration server uses the CNS configuration ID as a key to locate the attribute containing the Cisco IOS CLI configuration intended for the device that originated the configuration pull.

The network administrator must ensure a match between the CNS agent ID as defined on the routing device and the CNS agent ID contained in the directory attribute that corresponds to the configuration intended for the routing device. Within the routing device, the default value of the CNS agent ID is always set to the hostname. If the hostname changes, the CNS agent ID also changes. If the CNS agent ID is set using the CLI, any change will be followed by a message sent to syslog or an event message will be sent.

The CNS agent ID does not address security issues.

CNS Password

The CNS password is used to authenticate the CNS device. You must configure the CNS password the first time a router is deployed, and the CNS password must be the same as the bootstrap password set on the Configuration Engine (CE). If both the router and the CE bootstrap password use their default settings, a newly deployed router will be able to connect to the CE. Once connected, the CE manages the CNS password. Network administrators must ensure not to change the CNS password. If the CNS password is changed, connectivity to the CE will be lost.

Command Scheduler

The Command Scheduler (KRON) Policy for System Startup feature enables support for the Command Scheduler upon system startup.

The Command Scheduler allows customers to schedule fully-qualified EXEC mode CLI commands to run once, at specified intervals, at specified calendar dates and times, or upon system startup. Originally designed to work with CNS commands, Command Scheduler now has a broader application. Using the CNS image agent feature, remote routers residing outside a firewall or using Network Address Translation (NAT) addresses can use Command Scheduler to launch CLI at intervals, to update the image running in the router.

Command Scheduler has two basic processes. A policy list is configured containing lines of fully-qualified EXEC CLI commands to be run at the same time or same interval. One or more policy lists are then scheduled to run after a specified interval of time, at a specified calendar date and time, or upon system startup. Each scheduled occurrence can be set to run either once only or on a recurring basis.

CNS Flow-Through Provisioning

Cisco Networking Services (CNS) Flow-Through Provisioning provides the infrastructure for automated configuration of large numbers of network devices. Based on CNS event and configuration agents, it eliminates the need for an onsite technician to initialize the device. The result is an automated workflow from initial subscriber-order entry through Cisco manufacturing and shipping to final device provisioning and subscriber billing. This functionality focuses on a root problem of today's service-provider and other similar business models: use of human labor in activating service.

To achieve such automation, CNS Flow-Through Provisioning relies on standardized configuration templates that you create. However, the use of such templates requires a known fixed hardware configuration, uniform for all subscribers. There is no way to achieve this without manually prestaging each line card or module within each chassis. While the inventory within a chassis is known at time of manufacture, controlling which line cards or modules are in which slots thereafter is labor-intensive and error-prone.

To overcome these difficulties, CNS Flow-Through Provisioning defines a new set of Cisco IOS commands—the **cns** commands. When a remote router is first powered on, these commands do the following:

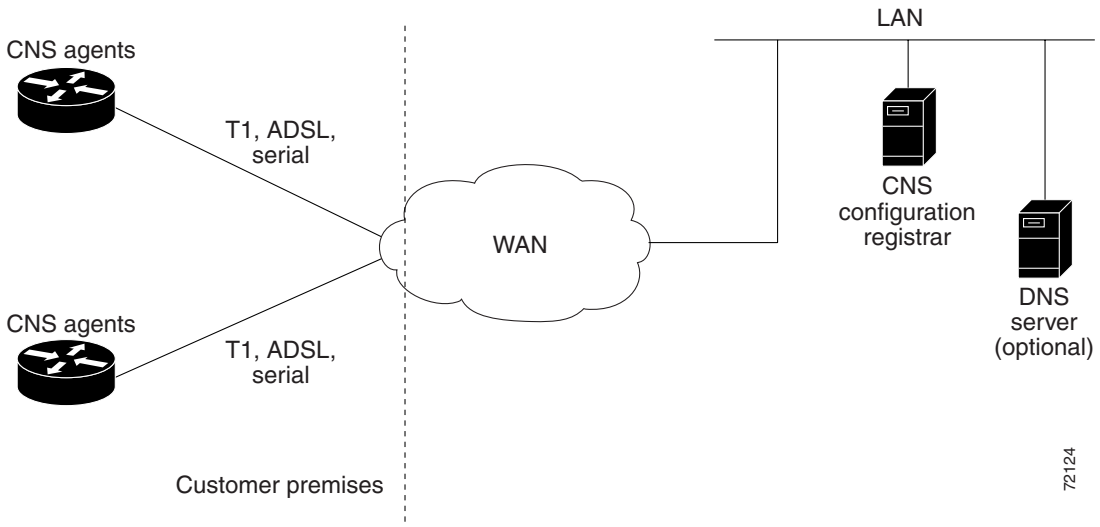
1. To each router interface in turn, applies a preset temporary bootstrap configuration that tries to contact the CNS configuration engine. A successful connection determines the connecting interface.
2. Connects, by way of software called a CNS agent, to a CNS configuration engine housed in a Cisco IE2100 device.
3. Passes to the CNS configuration engine a device-unique ID, along with a human-readable description of the router's line-card or module inventory by product number and location, in XML format.

In turn, the configuration engine does the following:

1. Locates in a Lightweight Directory Access Protocol (LDAP) directory, based on the device IDs, a predefined configuration template for the main chassis and subconfiguration template for each line card or module.
2. Substitutes actual slot numbers from the chassis inventory for the template's slot-number parameters, thus resolving the templates into subscriber-specific configurations that match the true line-card or module slot configuration.
3. Downloads this initial configuration to the target router. The CNS agent directly applies the configuration to the router.

Figure 1 shows the CNS Flow-Through Provisioning architecture.

Figure 1 CNS Flow-Through Provisioning Architecture



Configurations

CNS Flow-Through Provisioning involves three different types of configuration on the remote router:

- Bootstrap configuration

  You specify the preset bootstrap configuration on which this solution depends as part of your order from Cisco using Cisco Configuration Express, an existing service integrated with the Cisco.com order-entry tool. You specify a general-subscriber nonspecific bootstrap configuration that provides connectivity to the CNS configuration engine. Cisco then applies this configuration to all the devices of that order in a totally automated manufacturing step. This configuration runs automatically on power-on.

- Initial configuration

  The CNS configuration engine downloads an initial configuration, once only, to replace the temporary bootstrap configuration. You can either save or not save it in the router's nonvolatile NVRAM memory:

  - If you save the configuration, the bootstrap configuration is overwritten.
  - If you do not save the configuration, the download procedure repeats each time that the router powers off and then back on. Repeating the download procedure enables the router to update to the current Cisco IOS configuration without intervention.

- Incremental (partial) configuration

  On subsequent reboot, incremental or partial configurations are performed to update the configuration without the network having to shut down. Such configurations can be delivered either in a push operation that you initiate or a pull operation on request from the router.

Unique IDs

Key to this solution is the capability to associate, with each device, a simple, manageable, and unique ID that is compatible with your systems for order entry, billing, provisioning, and shipping and can also link your order-entry system to the Cisco order-fulfillment system. Such an ID must have the following characteristics:

- Be available from manufacturing as part of order fulfillment
- Be recordable on the shipping carton and chassis
- Be available to the device's Cisco IOS software
- Be modifiable after the device is first powered up
- Be representative of both a specific chassis and a specific entry point into your network

To define such an ID, CNS Flow-Through Provisioning equips the CNS agent with a new set of commands—the **cns** commands—with which you specify how configurations should be done and, in particular, how the system defines unique IDs. You enable the Cisco IOS software to auto-discover the unique ID according to directions that you specify and information that you provide, such as chassis serial number, MAC address, IP address, and several other possibilities. The **cns** commands are part of the bootstrap configuration of the manufactured device, specified to Cisco Configuration Express at time of order.

Within this scope, Configuration Express and the **cns** commands also allow you to define custom asset tags to your own specifications, which are serialized during manufacture and automatically substituted into the unit's bootstrap configuration.

Cisco appends tags to the carton for all the various types of IDs supported by the **cns** commands, so that these values can be bar-code read at shipping time and fed back into your systems. Alternatively, these IDs are also available through a direct XML-software interface between your system and the Cisco order-status engine, eliminating the need for bar-code reading. The CNS agent also provides a feedback mechanism whereby the remote device can receive XML events or commands to modify the device's ID, in turn causing that same device to broadcast an event indicating the old/new IDs.

Management Point

On most networks, a small percentage of individual remote routers get configured locally. This can potentially be a serious problem, not only causing loss of synchronization across your network but also opening your system to the possibility that an automatic reconfiguration might conflict with an existing configuration and cause a router to become unusable or even to lose contact with the network.

To address this problem, you can designate a management point in your network, typically on the Cisco IE2100 CNS configuration engine, and configure it to keep track of the configurations on all remote routers.

To enable this solution, configure the CNS agent to publish an event on the CNS event bus whenever any change occurs to the running configuration. This event indicates exactly what has changed (old/new), eliminating the need for the management point to perform a highly unscalable set of operations such as telnetting into the device, applying a script, reading back the entire running configuration, and determining the difference between old and new configurations. Additionally, you can arrange for Simple Network Management Protocol (SNMP) notification traps of configuration changes occurring through the SNMP MIB set.

Point-to-Point Event Bus

Today's business environment requires that you be able to ensure your customers a level of service not less than what they are actually paying for. Toward this end, you activate service-assurance applications that broadcast small poll/queries to the entire network while expecting large responses from a typically small subset of devices according to the criteria of the query.

For these queries to be scalable, it is necessary for the replying device to bypass the normal broadcast properties of the event bus and instead reply on a direct point-to-point channel. While all devices need the benefit of the broadcasted poll so that they can all be aware of the query to which they may need to reply, the devices do not have to be aware of each others' replies. Massive copying and retransmission of device query replies, as part of the unnecessary reply broadcast, is a serious scalability restriction.

To address this scalability problem, the CNS event bus has a point-to-point connection feature that communicates directly back to the poller station.

CNS Flow-Through Provisioning provides the following benefits.

Automated Configuration

CNS Flow-Through Provisioning simplifies installation by moving configuration requirements to the CNS configuration engine and allowing the Cisco IOS configuration to update automatically. The registrar uses popular industry standards and technologies such as XML, Active Directory Services Interface (ADSI)/Active Directory, HTTP/Web Server, ATM Switch Processor (ASP), and Publish-Subscribe Event Bus. The CNS configuration agent enables the CNS configuration engine to configure remote routers in a plug-and-play manner.

Unique IP Addresses and Hostname

CNS Flow-Through Provisioning uses DNS reverse lookup to retrieve the hostname by passing the IP address, then assigns the IP address and optionally the hostname to the remote router. Both IP address and hostname are thus guaranteed to be unique.

Reduced Technical Personnel Requirements

CNS Flow-Through Provisioning permits remote routers to be installed by a person with limited or no technical experience. Because configuration occurs automatically on connection to the network, a network engineer or technician is not required for installation.

Rapid Deployment

Because a person with limited or no technical experience can install a remote router immediately without any knowledge or use of Cisco IOS software, the router can be sent directly to its final premises and be brought up without technician deployment.

Direct Shipping

Routers can be shipped directly to the remote end-user site, eliminating warehousing and manual handling. Configuration occurs automatically on connection to the network.

Remote Updates

CNS Flow-Through Provisioning automatically handles configuration updates, service additions, and deletions. The CNS configuration engine performs a push operation to send the information to the remote router.

Security

Event traffic to and from the remote router is opaque to unauthorized listeners or intruders to your network. CNS agents leverage the latest security features in Cisco IOS software.

CNS Zero Touch

The CNS Zero Touch feature provides a zero touch deployment solution where the router contacts a CNS configuration engine to retrieve its full configuration automatically. This capability is made possible through a single generic bootstrap configuration file common across all service provider end customers subscribing to the services. Within the CNS framework, customers can create this generic bootstrap configuration without device-specific or network-specific information such as interface type, line type, or controller type (if applicable).

The CNS connect functionality is configured with a set of CNS connect templates. A CNS connect profile is created for connecting to the CNS configuration engine and to implement the CNS connect templates on a Customer Premise Equipment (CPE) router. CNS connect variables can be used as placeholders within a CNS connect template configuration. These variables, such as the active DLCI, are substituted with real values before the CNS connect templates are sent to the router's parser.

To use the zero touch functionality, the router that is to be initialized must have a generic bootstrap configuration. This configuration includes CNS connect templates, CNS connect profiles, and the **cns config initial** command. This command initiates the CNS connect function.

The CNS connect functionality performs multiple ping iterations through the router's interfaces and lines, as well as any available controllers. For each iteration, the CNS connect function attempts to ping the CNS configuration engine. If the ping is successful, the pertinent configuration information can be downloaded from the CNS configuration engine. If connectivity to the CNS configuration engine is unsuccessful, the CNS connect function removes the configuration applied to the selected interface, and the CNS connect process restarts with the next available interface specified by the CNS connect profile.

The CNS Zero Touch feature provides the following benefits:

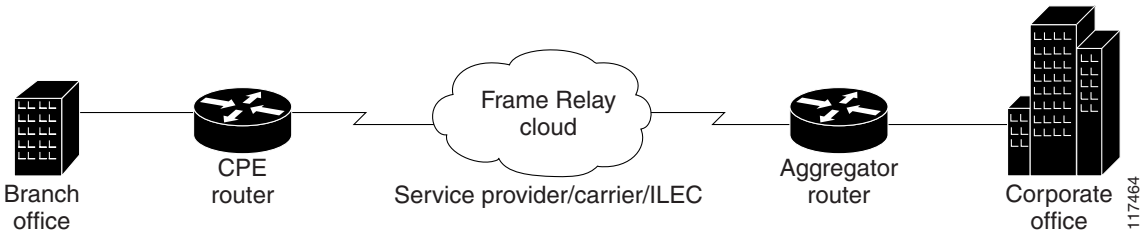
- Ensures consistent CNS commands between Cisco IOS Release 12.3 and 12.3T.
- Use of a channel service unit (E1 or T1 controller) is allowed.

CNS Frame Relay Zero Touch

The CNS Frame Relay Zero Touch feature provides a CNS zero touch deployment solution over Frame Relay where the CPE router discovers its data-link connection identifier (DLCI) and IP address dynamically, and then contacts a CNS engine to retrieve its full configuration automatically. This capability is made possible through a single generic bootstrap configuration file common across all service provider end customers subscribing to the services. Within the CNS framework, customers who deploy Frame Relay can create this generic bootstrap configuration without device-specific or network-specific information such as the DLCI, IP address, interface type, controller type (if applicable), or the next hop interface used for the static default route.

Figure 2 illustrates a typical customer network architecture using Frame Relay.

Figure 2 Connectivity in a Frame Relay Customer Network



The CPE router is deployed at multiple sites. Each site connects to a Frame Relay cloud through a point-to-point permanent virtual circuit (PVC). Connectivity from the Frame Relay cloud to the corporate office is through a PVC that terminates at the corporate office. IP traffic sent to the CNS configuration engine is routed through the corporate office. The PVC is identified by its DLCI. The DLCI can vary between branch offices. In order to support zero touch deployment, the CPE router must be able to learn which DLCI to use to connect to the CNS configuration engine.

To support the zero touch capability, the Frame Relay functionality has been modified in the following two ways:

- A new Cisco IOS command, the **ip address dynamic** command has been introduced to discover the CPE router's IP address dynamically based on the aggregator router's IP address. To configure IP over Frame Relay, the local IP address must be configured on the interface.
- The CPE router can now read Local Management Interface (LMI) messages from a Frame Relay switch and determine the list of available DLCIs.

The CNS connect functionality is configured with a set of CNS connect templates. A CNS connect profile is created for connecting to the CNS configuration engine and to implement the CNS connect templates on a CPE router. CNS connect variables can be used as placeholders within a CNS connect template configuration. These variables, such as the active DLCI, are substituted with real values before the CNS connect templates are sent to the router's parser.

When a CPE router is placed in a Frame Relay network, it contains a generic bootstrap configuration. This configuration includes customer-specific Frame Relay configuration (including the LMI type), CNS connect templates, CNS connect profiles, and the **cns config initial** command. This command initiates the CNS connect function.

The CNS connect functionality begins by selecting the first available controller or interface specified by the CNS connect profile and then performs multiple ping iterations through all the associated active DLCIs. For each iteration, the CNS connect function attempts to ping the CNS configuration engine. If the ping is successful, the pertinent configuration information can be downloaded from the CNS configuration engine.

When iterating over the active DLCIs on a Frame Relay interface, the router must be able to automatically go through a list of active DLCIs returned by the LMI messages for that interface and select an active DLCI to use. When more than one of the active DLCIs allow IP connectivity to the CNS configuration engine, the DLCI used will be the first one tried by the CNS connect functionality. If the ping attempt is unsuccessful, the next active DLCI is tried and so on. If connectivity to the CNS configuration engine is unsuccessful for all active DLCIs, the CNS connect function removes the configuration applied to the selected controller or interface, and the CNS connect process restarts with the next available controller or interface specified by the CNS connect profile.

The CNS Frame Relay Zero Touch feature provides the following benefits:

- A service provider can have a single common bootstrap configuration.
- The generic bootstrap configuration does not require the IP address to be hard-wired.
- The point-to-point DLCI does not need to be known in advance.
- IP directly over Frame Relay is allowed.
- Use of a channel service unit (E1 or T1 controller) is allowed.

How to Configure CNS

This section contains the following tasks:

Deploying the CNS Router

Perform this task to manually install an initial CNS configuration.

Your remote router arrives from the factory with a bootstrap configuration. Upon initial power-on, the router automatically pulls a full initial configuration from the CNS configuration engine, although you can optionally arrange for this manually as well. After initial configuration, you can optionally arrange for periodic incremental (partial) configurations for synchronization purposes.

For more details on using the Cisco CNS configuration engine to automatically install the initial CNS configuration, see the *Cisco CNS Configuration Engine Administrator's Guide* at http://www.cisco.com/univercd/cc/td/doc/product/rtrmgmt/cns/ce/rel13/ag13/index.htm.

Initial CNS Configuration

Initial configuration of the remote router occurs automatically when the router is initialized on the network. Optionally, you can perform this configuration manually.

CNS assigns the remote router a unique IP address or hostname. After resolving the IP address (using Serial Line Address Resolution Protocol (SLARP), ATM Inverse ARP (ATM InARP), or PPP protocols), the system optionally uses Domain Name System (DNS) reverse lookup to assign a hostname to the router and invokes the CNS agent to download the initial configuration from the CNS configuration engine.

Incremental Configuration

Incremental or partial configuration allows the remote router to be incrementally configured after its initial configuration. You must perform these configurations manually through the CNS configuration engine. The registrar allows you to change the configuration templates, edit parameters, and submit the new configuration to the router without a software or hardware restart.

Prerequisites

Before you can configure an incremental configuration, CNS must be operational and the required CNS agents configured.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **cns template connect** *name*
4. **cli** *config-text*
5. Repeat Step 4 to add all required CLI commands.
6. **exit**
7. **cns connect** *name* [**retry-interval** *interval-seconds*] [**retries** *number-retries*] [**timeout** *timeout-seconds*] [**sleep** *sleep-seconds*]
8. **discover** {**line** *line-type* | **controller** *controller-type* | **interface** [*interface-type*]}
   or
   template *name*
9. **exit**
10. **cns config initial** {*host-name* | *ip-address*} [**encrypt**] [*port-number*] [**page** *page*] [**syntax-check**] [**no-persist**] [**source** *ip-address*] [**status** *url*] [**event**] [**inventory**]
11. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**cns template connect** *name* **Example:** `Router(config)# cns template connect template 1`	Enters CNS template connect configuration mode and defines the name of a CNS connect template.
Step 4	**cli** *config-text* **Example:** `Router(config-templ-conn)# cli encapsulation ppp`	Specifies commands to configure the interface.

	Command or Action	Purpose
Step 5	Repeat Step 4 to add all required CLI commands. **Example:** `Router(config-templ-conn)# cli ip directed-broadcast`	Repeat Step 4 to add other CLI commands to configure the interface or to configure the modem lines.
Step 6	**exit** **Example:** `Router(config-templ-conn)# exit`	Exits CNS template connect configuration mode and completes the configuration of a CNS connect template. **Note** Entering the **exit** command is required. This requirement was implemented to prevent accidentally entering a command without the **cli** command.
Step 7	**cns connect** *name* [**retry-interval** *interval-seconds*] [**retries** *number-retries*] [**timeout** *timeout-seconds*] [**sleep** *sleep-seconds*] **Example:** `Router(config)# cns connect profile-1 retry-interval 15 timeout 90`	Enters CNS connect configuration mode and defines the parameters of a CNS connect profile for connecting to the CNS configuration engine.

Step 8	**discover** {**line** *line-type* | **controller** *controller-type* | **interface** [*interface-type*]}
	or
	template *name*
	
	Example:
	`Router(config-cns-conn)# discover interface serial`
	or
	
	Example:
	`Router(config-cns-conn)# template template-1`
		(Optional) Configures a generic bootstrap configuration.
		• **discover**—Defines the interface parameters within a CNS connect profile for connecting to the CNS configuration engine.
		or
		• **template**—Specifies a list of CNS connect templates within a CNS connect profile to be applied to a router's configuration.

Step 9	**exit** **Example:** `Router(config-cns-conn)# exit`	Exits CNS connect configuration mode and returns to global configuration mode.
Step 10	**cns config initial** {*host-name* | *ip-address*} [**encrypt**] [*port-number*] [**page** *page*] [**syntax-check**] [**no-persist**] [**source** *interface name*] [**status** *url*] [**event**] [**inventory**] **Example:** `Router(config)# cns config initial 10.1.1.1 no-persist`	Starts the CNS configuration agent, connects to the CNS configuration engine, and initiates an initial configuration. You can use this command only before the system boots for the first time. **Note** The optional **encrypt** keyword is available only in images that support Secure Socket Layer (SSL). **Caution** If you write the new configuration to NVRAM by omitting the **no-persist** keyword, the original bootstrap configuration is overwritten.

	Command or Action	Purpose
Step 11	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode and returns to privileged EXEC mode.

Configuring the CNS Event and EXEC Agents

Perform this task to enable and configure the CNS Event and EXEC agents.

CNS Event Agent Parameters

The CNS event agent command—**cns event**—has several parameters that can be configured. The **failover-time** keyword is useful if you have a backup CNS event gateway configured. If the CNS event agent is trying to connect to the gateway and it discovers that the route to the backup gateway is available before the route to the primary gateway, the *seconds* argument specifies how long the CNS event agent will continue to search for a route to the primary gateway before attempting to link to the backup gateway.

Unless you are using a bandwidth-constrained link, you should set a keepalive timeout and retry count. Doing so allows the management network to recover gracefully should a Cisco IE2100 configuration engine ever fail. Without the keepalive data, such a failure requires manual intervention on every device. The *seconds* value multiplied by the *retry-count* value determines the length of idle time before the CNS event agent will disconnect and attempt to reconnect to the gateway. We recommend a minimum *retry-count* value of 2.

If the optional **source** keyword is used, the source IP address might be a secondary IP address of a specific interface to allow a management network to run on top of a production network.

Note Although other CNS agents may be configured, no other CNS agents are operational until the **cns event** command is entered because the CNS event agent provides a transport connection to the CNS event bus for all other CNS agents.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **cns config partial** {*host-name* | *ip-address*} [**encrypt**] [*port-number*] [**source** *ip-address*] [**inventory**]
4. **logging cns-events** [*severity-level*]
5. **cns exec** [*host-name* | *ip-address*] [**encrypt** [*enc-port-number*]] [*port-number*] [**source** *ip-address*]
6. **cns event** {*host-name* | *ip-address*} [**encrypt**] [*port-number*] [**backup**] [**failover-time** *seconds*] [**keepalive** *seconds retry-count*] [**source** *ip-address*] [**clock-timeout** *time*] [**reconnect** *time*]
7. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**cns config partial** {*host-name* | *ip-address*} [**encrypt**] [*port-number*] [**source** *interface name*] [**inventory**] **Example:** `Router(config)# cns config partial 172.28.129.22 80`	(Optional) Starts the CNS configuration agent, which provides CNS configuration services to Cisco IOS clients, and initiates an incremental (partial) configuration. • Use the optional *port-number* argument to specify the port number for the configuration server. The default is 80. • Use the optional **source** keyword and *ip-address* argument to specify the use of an IP address as the source for CNS configuration agent communications. • Use the optional **inventory** keyword to send an inventory of the line cards and modules in the router to the CNS configuration engine as part of the HTTP request. **Note** The optional **encrypt** keyword is available only in images that support SSL.
Step 4	**logging cns-events** [*severity-level*] **Example:** `Router(config)# logging cns-events 2`	(Optional) Enables XML-formatted system event message logging to be sent through the CNS event bus. • Use the optional *severity-level* argument to specify the number or name of the desired severity level at which messages should be logged. The default is level 7 (debugging).
Step 5	**cns exec** [*host-name* | *ip-address*] [**encrypt** [enc-*port-number*]] [*port-number*] [**source** *ip-address*] **Example:** `Router(config)# cns exec 10.1.2.3 93 source 172.17.2.2`	(Optional) Enables and configures the CNS EXEC agent, which provides CNS EXEC services to Cisco IOS clients. • Use the optional *port-number* argument to specify the port number for the EXEC server. The default is 80. • Use the optional **source** keyword and *ip-address* argument to specify the use of an IP address as the source for CNS EXEC agent communications. **Note** The optional **encrypt keyword** is available only in images that support SSL.

	Command or Action	Purpose
Step 6	**cns event** {*hostname* | *ip-address*} [**encrypt**] [*port-number*] [**backup**] [**failover-time** *seconds*] [**keepalive** *seconds retry-count*] [**source** *ip-address*] [**clock-timeout** *time*] [**reconnect** *time*] **Example:** `Router(config)# cns event 172.28.129.22 source 172.22.2.1`	Configures the CNS event gateway, which provides CNS event services to Cisco IOS clients. • The optional **encrypt** keyword is available only in images that support SSL. • Use the optional *port-number* argument to specify the port number for the event server. The default is 11011 with no encryption and 11012 with encryption. • Use the optional **backup** keyword to indicate that this is the backup gateway. Before configuring a backup gateway, ensure that a primary gateway is configured. • Use the optional **failover-time** keyword and *seconds* argument to specify a time interval in seconds to wait for the primary gateway route after the route to the backup gateway is established. • Use the optional **keepalive** keyword with the *seconds* and *retry-count* arguments to specify the keepalive timeout in seconds and the retry count. • Use the optional **source** keyword and *ip-address* argument to specify the use of an IP address as the source for CNS event agent communications. • Use the optional **clock-timeout** keyword to specify the maximum time, in minutes, that the CNS event agent will wait for the clock to be set for transports (such as SSL) that require an accurate clock. • Use the optional **reconnect** keyword to specify the configurable upper limit of the maximum retry timeout. **Note** Until the **cns event** command is entered, no transport connections to the CNS event bus are made and therefore no other CNS agents are operational.
Step 7	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode and returns to privileged EXEC mode.

Troubleshooting Tips

- Use the **show cns event connections** command to check that the CNS event agent is connected to the CNS event gateway.
- Use the **show cns event subject** command to check that the image agent subject names are registered. Subject names for the CNS image agent begin with cisco.mgmt.cns.image.

Configuring the CNS Image Agent

Perform this task to configure CNS image agent parameters using CLI commands.

CNS Image Agent ID

CNS uses a unique identifier to identify an image agent associated with that Cisco IOS device. Using the same process as CNS event and configuration agents, the configuration of the **cns id** command determines whether an IP address or MAC address of a specified interface, the hardware serial hardware number of the device, an arbitrary text string, or the hostname of the device is used as the image ID. By default, the system uses the hostname of the device.

The CNS image ID is sent in the content of the messages sent by the image agent and allows an application to know the unique image ID of the Cisco IOS device that generated the message. A password can be configured and associated with the image ID in the image agent messages.

Prerequisites

- To configure the CNS image agent to use HTTP or HTTP over SSL (HTTPS) to communicate with an image server, you need to know the URL for the image server and the URL to which status messages can be sent.
- If you are using HTTPS to communicate with the image server, you must set up security certificates to allow the server to be authenticated by the image agent when the connection is established.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **cns id** *type number* {**ipaddress** | **mac-address**} [**event** | **image**]
   or
   cns id {**hardware-serial** | **hostname** | **string** *text*} [**event** | **image**]
4. **cns password** *password*
5. **cns image** [**server** *server-url* [**status** s*tatus-url*]]
6. **cns image password** *image-password*
7. **cns image retry** *seconds*
8. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.

Step 3	**cns id** *type number* {**ipaddress** | **mac-address**} [**event** | **image**]
	or
	cns id {**hardware-serial** | **hostname** | **string** *text*} [**event** | **image**]
	
	Example:
	`Router(config)# cns id fastethernet 0/1 ipaddress image`
	or
	
	Example:
	`Router(config)# cns id hardware-serial image`
		Specifies a unique CNS ID and interface type and number from which to retrieve the unique ID.
		or
		Specifies a unique CNS ID assigned from the hardware serial number, device hostname, or an arbitrary text string.
		The following information applies to either version of the syntax.
		• Use the **event** keyword to specify an event agent ID.
		• Use the **image** keyword to specify an image agent ID.
		• If no keywords are used, the configuration agent ID is configured.

Step 4	**cns password** *password* **Example:** `Router(config)# cns password password1`	Specifies a password for the CNS ID. You must configure the CNS password the first time a router is deployed, and the CNS password must be the same as the bootstrap password set on the Configuration Engine (CE).
Step 5	**cns image** [**server** *server-url* [**status** *status-url*]] **Example:** `Router(config)# cns image server https://10.21.2.3/cns/imgsvr status https://10.21.2.3/cns/status/`	Enables CNS image agent services and specifies the URL of the image distribution server. • Use the optional **status** keyword and *status-url* argument to specify the URL of a web server to which error messages are written. • If the **status** keyword and *status-url* argument are not specified, status messages are sent as events on the CNS Event Bus. To view the status messages on the CNS Event Bus, the CNS event agent must be configured.
Step 6	**cns image password** *image-password* **Example:** `Router(config)# cns image password abctext`	(Optional) Specifies a password for CNS image agent services. • If a password is configured, the password is included with the image ID in CNS image agent messages sent out by the image agent. The receiver of these messages can use this information to authenticate the sending device.

	Command or Action	Purpose
Step 7	**cns image retry** *seconds* **Example:** `Router(config)# cns image retry 240`	(Optional) Specifies an image upgrade retry interval in seconds. • The default interval is 60 seconds.
Step 8	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode and returns the router to privileged EXEC mode.

What to Do Next

Proceed to the "Retrieving a CNS Image from a Server" section to connect to the web server and download an image.

If any of the commands in the task fail, proceed to the "Troubleshooting CNS Agents" section to try to determine the problem.

Configuring CNS Security Features

Perform this task to configure CNS security features.

CNS Trusted Servers

Use the **cns trusted-server** command to specify a trusted server for an individual CNS agent or for all the CNS agents. To avoid security violations, you can build a list of trusted servers from which CNS agents can receive messages. An attempt to connect to a server not on the list will result in an error message being displayed.

Configure a CNS trusted server when a CNS agent will redirect its response to a server address that is not explicitly configured on the command line for the specific CNS agent. For example, the CNS exec agent may have one server configured but receive a message from the CNS event bus that overrides the configured server. The new server address has not been explicitly configured, so the new server address is not a trusted server. An error will be generated when the CNS exec agent tries to respond to this new server address unless the **cns trusted-server** command has been configured for the new server address.

CNS Security Enhancement

CNS messages can be configured to use the CNS SOAP message structure, in which the username and password are authenticated. If AAA is configured, then CNS SOAP messages will be authenticated with AAA. If AAA is not configured, there will be no authentication.

Use the **cns aaa authentication** command to determine whether the CNS messages are using AAA security or not. If the **cns aaa authentication** command is configured, then all incoming SOAP messages into the device are authenticated by AAA.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **cns trusted-server** {**all-agents** | **config** | **event** | **exec** | **image**} *name*

4. **cns message format notification** [**version 1** | **version 2**]
5. **cns aaa authentication** *authentication-method*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**cns trusted-server** {**all-agents** | **config** | **event** | **exec** | **image**} *name* **Example:** `Router(config)# cns trusted-server event 10.19.2.5`	Configures a CNS trusted server for the specified hostname or IP address.
Step 4	**cns message format notification** [**version 1** | **version 2**] **Example:** `Router(config)# cns message format notification version 1`	Configures the message format for notification messages from a CNS device. Received messages which do not conform to the configured message format are rejected. Use version 1 to configure the non-SOAP message format. Use version 2 for SOAP message format.
Step 5	**cns aaa authentication** *authentication-method* **Example:** `Router(config)# cns aaa authentication method1`	Enables CNS AAA options. **Note** The authentication methods must be configured within AAA.

Retrieving a CNS Image from a Server

Perform this task to poll the image distribution server using HTTP or HTTPS.

Prerequisites

This task assumes that you have already configured the CNS image agent using the tasks in the "Configuring the CNS Image Agent" section.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **cns image retrieve** [**server** *server-url* [**status** *status-url*]]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**cns image retrieve** [**server** *server-url* [**status** *status-url*]] **Example:** `Router(config)# cns image retrieve server https://10.19.2.3/imgsvr/ status https://10.19.2.3/imgsvr/status/`	Contacts a Cisco CNS image distribution server and downloads a new image if a new image exists. • Use the optional **status** keyword and *status-url* argument to specify the URL of a web server to which status messages are written. • If the **server** and **status** keywords are not specified, the server and status URLs configured with the **cns image** command are used. **Note** We recommend using the **cns trusted-server** command to specify the host part of the server or status URL as a trusted server.

Troubleshooting Tips

- If the web server appears to be down, use the **ping** command to check connectivity.
- If using HTTP, use the **show ip http client all** command to display information about HTTP clients and connections.

Retrieving a CNS Configuration from a Server

Use this task to request the configuration of a device from a configuration server. Use the **cns trusted-server** command to specify which configuration server can be used (trusted).

Prerequisites

This task assumes that you have specified a trusted server using tasks in the "CNS Security Enhancement" section.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **cns config retrieve** {*host-name* | *ip-address*} [**encrypt**] [*port-number*] [**page** *page*] [**overwrite-startup**] [**retry** *retries* **interval** *seconds*] [**syntax-check**] [**no-persist**] [**source** *interface name*] [**status** *url*] [**event**] [**inventory**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**cns config retrieve** {*host-name* | *ip-address*} [**encrypt**] [*port-number*] [**page** *page*] [**overwrite-startup**] [**retry** *retries* **interval** *seconds*] [**syntax-check**] [**no-persist**] [**source** *interface name*] [**status** *url*] [**event**] [**inventory**] **Example:** `Router(config)# cns config retrieve server1 retry 5 interval 45`	Allows the router to retrieve configuration data from a web server. • The **retry** keyword is a number in the range 1 to 100, and will prompt for an **interval** in the range 1 to 3600 seconds.

Troubleshooting Tips

If you need to stop the retrieval process, enter the Ctrl+Shift+6 key sequence.

Configuring Command Scheduler Policy Lists and Occurrences

Perform this task to set up Command Scheduler policy lists of EXEC CNS commands and configure a Command Scheduler occurrence to specify the time or interval after which the CNS commands will run.

Command Scheduler Policy Lists

Policy lists consist of one or more lines of fully-qualified EXEC CLI commands. All commands in a policy list are executed when the policy list is run by Command Scheduler using the **kron occurrence** command. Use separate policy lists for CLI commands that are run at different times. No editor function is available, and the policy list is run in the order in which it was configured. To delete an entry, use the **no** form of the **cli** command followed by the appropriate EXEC command. If an existing policy list name is used, new entries are added to the end of the policy list. To view entries in a policy list, use the **show running-config** command. If a policy list is scheduled to run only once, it will not be displayed by the **show running-config** command after it has run.

Policy lists can be configured after the policy list has been scheduled, but each policy list must be configured before it is scheduled to run.

Command Scheduler Occurrences

An occurrence for Command Scheduler is defined as a scheduled event. Policy lists are configured to run after a specified interval of time, at a specified calendar date and time, or upon system startup. Policy lists can be run once, as a one-time event, or as recurring events over time.

Command Scheduler occurrences can be scheduled before the associated policy list has been configured, but a warning will advise you to configure the policy list before it is scheduled to run.

Prerequisites

The clock time must be set on the routing device before a Command Scheduler occurrence is scheduled to run. If the clock time is not set, a warning message will appear on the console screen after the **kron occurrence** command has been entered. Use the **clock** command or Network Time Protocol (NTP) to set the clock time.

The EXEC CLI to be run by Command Scheduler must be tested on the routing device to determine if it will run without generating a prompt or allowing execution interruption by keystrokes. Initial testing is important because Command Scheduler will delete the entire policy list if any CLI syntax fails. Removing the policy list ensures that any CLI dependencies will not generate more errors.

If you use the **conditional** keyword with the **kron policy-list** command, execution of the commands will stop when an error is encountered.

Restrictions

- No more than 31 policy lists can be scheduled to run at the same time.
- If a one-time occurrence is scheduled, the occurrence will not be displayed by the **show running-config** command after the occurrence has run.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **kron policy-list** *list-name* [**conditional**]
4. **cli** *command*
5. **exit**
6. **kron occurrence** *occurrence-name* [**user** *username*] {**in** [[*numdays***:**]*numhours***:**]*nummin* | **at** *hours***:***min* [[*month*] *day-of-month*] [*day-of-week*]} {**oneshot** | **recurring** | **system-startup**}
7. **policy-list** *list-name*
8. **exit**
9. **show kron schedule**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**kron policy-list** *list-name* [**conditional**] **Example:** `Router(config)# kron policy-list cns-weekly`	Specifies a name for a new or existing Command Scheduler policy list and enters kron-policy configuration mode. • If the *list-name* is new, a new policy list structure is created. • If the *list-name* exists, the existing policy list structure is accessed. The policy list is run in configured order with no editor function. • If the optional **conditional** keyword is used, execution of the commands stops when an error is encountered.
Step 4	**cli** *command* **Example:** `Router(config-kron-policy)# cli cns image retrieve server https://10.19.2.3/cnsweek/ status https://10.19.2.3/cnsstatus/week/`	Specifies the fully-qualified EXEC command and associated syntax to be added as an entry in the specified Command Scheduler policy list. • Each entry is added to the policy list in the order in which it is configured. • Repeat this step to add other EXEC CLI commands to a policy list to be executed at the same time or interval. **Note** EXEC commands that generate a prompt or can be terminated using keystrokes will cause an error.
Step 5	**exit** **Example:** `Router(config-kron-policy)# exit`	Exits kron-policy configuration mode and returns the router to global configuration mode.
Step 6	**kron occurrence** *occurrence-name* [**user** *username*] {**in** [[*numdays***:**]*numhours***:**]*nummin* | **at** *hours***:***min* [[*month*] *day-of-month*] [*day-of-week*]} {**oneshot** | **recurring** | **system-startup**} **Example:** `Router(config)# kron occurrence may user sales at 6:30 may 20 oneshot`	Specifies a name and schedule for a new or existing Command Scheduler occurrence and enters kron-occurrence configuration mode. • Use the **in** keyword to specify a delta time interval with a timer that starts when this command is configured. • Use the **at** keyword to specify a calendar date and time. • Choose either the **oneshot** or **recurring** keyword to schedule Command Scheduler occurrence once or repeatedly. Add the optional **system-startup** keyword for the occurrence to be at system startup.

	Command or Action	Purpose
Step 7	**policy-list** *list-name* **Example:** `Router(config-kron-occurrence)# policy-list sales-may`	Specifies a Command Scheduler policy list. • Each entry is added to the occurrence list in the order in which it is configured. **Note** If the CLI commands in a policy list generate a prompt or can be terminated using keystrokes, an error will be generated and the policy list will be deleted.
Step 8	**exit** **Example:** `Router(config-kron-occurrence)# exit`	Exits kron-occurrence configuration mode and returns the router to global configuration mode. • Repeat this step to exit global configuration mode.
Step 9	**show kron schedule** **Example:** `Router# show kron schedule`	(Optional) Displays the status and schedule information of Command Scheduler occurrences.

Examples

In the following example, output information is displayed about the status and schedule of all configured Command Scheduler occurrences:

```
Router# show kron schedule

Kron Occurrence Schedule
cns-weekly inactive, will run again in 7 days 01:02:33
may inactive, will run once in 32 days 20:43:31 at 6:30 on May 20
```

Troubleshooting Tips

Use the **debug kron** command in privileged EXEC mode to troubleshoot Command Scheduler command operations. Use any debugging command with caution because the volume of output generated can slow or stop the router operations.

Configuring Advanced CNS Features

Perform this task to configure more advanced CNS features. After the CNS agents are operational, you can configure some other features. You can enable the CNS inventory agent—that is, send an inventory of the router's line cards and modules to the CNS configuration engine—and enter CNS inventory mode.

Some other advanced features allow you to use the Software Developer's Toolkit (SDK) to specify how CNS notifications should be sent or how to access MIB information. Two encapsulation methods can be used: either nongranular (SNMP) encapsulation or granular (XML) encapsulation.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **cns mib-access encapsulation** {**snmp** | **xml** [**size** *bytes*]}

4. **cns notification encapsulation** {**snmp** | **xml**}
5. **cns inventory**
6. **transport event**
7. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**cns mib-access encapsulation** {**snmp** | **xml** [**size** *bytes*]} **Example:** `Router(config)# cns mib-access encapsulation snmp`	(Optional) Specifies the type of encapsulation to use when accessing MIB information. • Use the **snmp** keyword to specify that nongranular encapsulation is used to access MIB information. • Use the **xml** keyword to specify that granular encapsulation is used to access MIB information. The optional **size** keyword specifies the maximum size for response events, in bytes. The default byte value is 3072.
Step 4	**cns notifications encapsulation** {**snmp** | **xml**} **Example:** `Router(config)# cns notifications encapsulation xml`	(Optional) Specifies the type of encapsulation to use when sending CNS notifications. • Use the **snmp** keyword to specify that nongranular encapsulation is used when CNS notifications are sent. • Use the **xml** keyword to specify that granular encapsulation is used when CNS notifications are sent.
Step 5	**cns inventory** **Example:** `Router(config)# cns inventory`	Enables the CNS inventory agent and enters CNS inventory mode. • An inventory of the router's line cards and modules is sent to the CNS configuration engine.
Step 6	**transport event** **Example:** `Router(cns-inv)# transport event`	Specifies that inventory requests are sent out with each CNS inventory agent message.
Step 7	**exit** **Example:** `Router(cns-inv)# exit`	Exits CNS inventory mode and returns to global configuration mode. • Repeat this command to return to privileged EXEC mode.

Troubleshooting CNS Agents

This section explains how to troubleshoot CNS agent issues.

The **show** commands created for the CNS image agent display information that is reset to zero after a successful reload of the device. Depending on the configuration of the image distribution process, the new image may not reload immediately. When a reload is not immediate or has failed, use the CNS image agent **show** commands to determine whether the image agent has connected to the image distribution server over HTTP or whether the image agent is receiving events from an application over the CNS Event Bus.

SUMMARY STEPS

1. **enable**
2. **show cns image status**
3. **clear cns image status**
4. **show cns image connections**
5. **show cns image inventory**
6. **debug cns image** [**agent** | **all** | **connection** | **error**]
7. **show cns event connections**
8. **show cns event subject** [*name*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables higher privilege levels, such as privileged EXEC mode. • Enter your password if prompted.
Step 2	**show cns image status** **Example:** `Router# show cns image status`	(Optional) Displays information about the CNS image agent status.
Step 3	**clear cns image status** **Example:** `Router# clear cns image status`	(Optional) Clears CNS image agent status statistics.
Step 4	**show cns image connections** **Example:** `Router# show cns image connections`	(Optional) Displays information about CNS image management server HTTP or HTTPS connections.

	Command or Action	Purpose
Step 5	**show cns image inventory** **Example:** `Router# show cns image inventory`	(Optional) Displays inventory information about the CNS image agent. • This command displays a dump of XML that would be sent out in response to an image agent inventory request message. The XML output can be used to determine the information requested by an application.
Step 6	**debug cns image** [**agent** | **all** | **connection** | **error**] **Example:** `Router# debug cns image all`	(Optional) Displays debugging messages for CNS image agent services.
Step 7	**show cns event connections** **Example:** `Router# show cns event connections`	(Optional) Displays the status of the CNS event agent connection—such as whether it is connecting to the gateway, connected, or active—and to display the gateway used by the event agent and its IP address and port number.
Step 8	**show cns event subject** [*name*] **Example:** `Router# show cns event subject subject1`	(Optional) Displays a list of subjects of the CNS event agent that are subscribed to by applications.

Examples

This section provides the following output examples:

- Sample Output for the show cns image status Command
- Sample Output for the show cns image connections Command
- Sample Output for the show cns image inventory Command
- Sample Output for the debug cns image Command

Sample Output for the show cns image status Command

In the following example, status information about the CNS image agent is displayed using the **show cns image status** privileged EXEC command:

```
Router# show cns image status

Last upgrade started at 11:45:02.000 UTC Mon May 6 2003
Last upgrade ended at 11:56:04.000 UTC Mon May 6 2003 status SUCCESS

Last successful upgrade ended at 11:56:04.000 UTC Mon May 6 2003
Last failed upgrade ended at 06:32:15.000 UTC Wed Apr 16 2003
Number of failed upgrades: 2
Number of successful upgrades: 6
 messages received: 12
 receive errors: 5
Transmit Status
  TX Attempts:4
    Successes:3         Failures 2
```

Sample Output for the show cns image connections Command

In the following example, information about the status of the CNS image management HTTP connections is displayed using the **show cns image connections** privileged EXEC command:

```
show cns image connections

CNS Image Agent:  HTTP connections
Connection attempts 1
never connected:0   Abrupt disconnect:0
Last successful connection at 11:45:02.000 UTC Mon May 6 2003
```

Sample Output for the show cns image inventory Command

In the following example, information about the CNS image agent inventory is displayed using the **show cns image inventory** privileged EXEC command:

```
show cns image inventory

Inventory Report
imageInventoryReport deviceName imageID Router /imageID hostName Router /ho
IOS (tm) C2600 Software (C2600-I-M), Experimental Version 12.3(20030414:081500)]
Copyright (c) 1986-2003 by cisco Systems, Inc.
Compiled Mon 14-Apr-03 02:03 by engineer /versionString imageFile tftp://10.25.2.1.
```

Sample Output for the debug cns image Command

In the following example, debugging messages for all CNS image agent services are displayed using the **debug cns image** privileged EXEC command. The CNS image agent in this example is connecting to an image server over HTTP. After connecting, the image server asks for an inventory of the Cisco IOS device.

```
Router# debug cns image all

    All cns image debug flags are on

Router# cns image retrieve

May  7 06:11:42.175: CNS Image Agent: set EXEC lock
May  7 06:11:42.175: CNS Image Agent: received message from EXEC
May  7 06:11:42.175: CNS Image Agent: set session lock 1
May  7 06:11:42.175: CNS Image Agent: attempting to send to
destination(http://10.1.36.8:8080/imgsrv/xgate):
?xml version="1.0" encoding="UTF-8"? cnsMessageversion="1.0" senderCredentials userName
dvlpr-7200-6 /userName /senderCredentials
messageID dvlpr-7200-6_2 /messageID sessionControl imageSessionStart version="1.0"
initiatorInfotrigger EXEC/trigger initiatorCredentials userName dvlpr-7200-6/userName
/initiatorCredentials /initiatorInfo /imageSessionStart /sessionControl /cnsMessage
May  7 06:11:42.175: CNS Image Agent: clear EXEC lock
May  7 06:11:42.175: CNS Image Agent: HTTP message sent
url:http://10.1.36.8:8080/imgsrv/xgate

May  7 06:11:42.191: CNS Image Agent: response data alloc 4096 bytes
May  7 06:11:42.191: CNS Image Agent: HTTP req data free
May  7 06:11:42.191: CNS Image Agent: response data freed
May  7 06:11:42.191: CNS Image Agent: receive message
?xml version="1.0" encoding="UTF-8"?
cnsMessage version="1.0"
senderCredentials
userName myImageServer.cisco.com/userName
passWord R0lGODlhcgGSALMAAAQCAEMmCZtuMFQxDS8b/passWord
/senderCredentials
messageID dvlpr-c2600-2-476456/messageID
request
replyTo
```

```
serverReply http://10.1.36.8:8080/imgsrv/xgate /serverReply
/replyTo
imageInventory
inventoryItemList
all/
/inventoryItemList
/imageInventory
/request
/cnsMessage
```

Sample Output for the show cns event Commands

The following example displays the IP address and port number of the primary and backup gateways:

```
Router# show cns event connections

The currently configured primary event gateway:
        hostname is 10.1.1.1.
        port number is 11011.
Event-Id is Internal test1
Keepalive setting:
        none.
Connection status:
        Connection Established.
The currently configured backup event gateway:
        none.
The currently connected event gateway:
        hostname is 10.1.1.1.
        port number is 11011.
```

The following sample displays a list of subjects of the CNS event agent that are subscribed to by applications:

```
Router# show cns event subject

The list of subjects subscribed by applications.
   cisco.cns.mibaccess:request
   cisco.cns.config.load
   cisco.cns.config.reboot
   cisco.cns.exec.cmd
```

Configuration Examples for CNS

This section provides the following configuration examples:

Deploying the CNS Router: Example

The following example shows an initial configuration on a remote router. The hostname of the remote router is the unique ID. The CNS configuration engine IP address is 172.28.129.22.

```
cns template connect template1
 cli ip address negotiated
 cli encapsulation ppp
 cli ip directed-broadcast
 cli no keepalive
 cli no shutdown
 exit
cns connect host1 retry-interval 30 retries 3
exit
 hostname RemoteRouter
 ip route 172.28.129.22 255.255.255.0 10.11.11.1
 cns id Ethernet 0 ipaddress
 cns config initial 10.1.1.1 no-persist
 exit
```

Configuring a Partial Configuration: Example

Incremental or partial configuration allows the remote router to be incrementally configured after its initial configuration. You must perform these configurations manually through the CNS configuration engine. The registrar allows you to change the configuration templates, edit parameters, and submit the new configuration to the router without a software or hardware restart.

The following example shows incremental (partial) configuration on a remote router. The CNS configuration engine IP address is 172.28.129.22, and the port number is 80.

```
cns config partial 172.28.129.22 80
```

Enabling and Configuring CNS Agents: Example

The following example shows various CNS agents being enabled and configured starting with the configuration agent being enabled with the **cns config partial** command to configure an incremental (partial) configuration on a remote router. The CNS configuration engine IP address is 172.28.129.22, and the port number is 80. The CNS exec agent is enabled with an IP address of 172.28.129.23, and the CNS event agent is enabled with an IP address of 172.28.129.24. Until the CNS event agent is enabled, no other CNS agents are operational.

```
 cns config partial 172.28.129.22 80
 cns exec 172.28.129.23 source 172.22.2.2
 cns event 172.28.129.24 source 172.22.2.1
 exit
```

In the following example, the CNS image agent parameters are configured using the CLI. An image ID is specified to use the IP address of the FastEthernet interface 0/1, a password is configured for the CNS image agent services, the CNS image upgrade retry interval is set to four minutes, and image management and status servers are configured.

```
cns id FastEthernet0/1 ipaddress image
cns image retry 240
cns image password abctext
cns image server https://10.21.2.3/cns/imgsvr status https://10.21.2.3/cns/status/
```

In the following example, the CNS image agent is configured to use the CNS Event Bus. An image ID is specified as the hardware serial number of the networking device, the CNS event agent is enabled with a number of parameters, and the CNS image agent is enabled without any keywords or options. The CNS image agent will listen for events on the CNS Event Bus.

```
cns id hardware-serial image
cns event 10.21.9.7 11011 keepalive 240 120 failover-time 5
cns image
cns image password abctext
```

CNS Flow-Through Provisioning: Examples

Cisco Configuration Express File Using T1 over HDLC Protocol Example

The following example shows use of the Cisco Configuration Express file to configure the remote router before delivery to its final premises. In the example, 172.28.129.22 is the IP address of the CNS configuration engine.

```
cns config initial 172.28.129.22 no-persist
!cns configure and event agents
cns event 172.28.129.22
controller t1 0
!T1 configuration
framing esf
linecode b8zs
channel-group 0 timeslots 1-24 speed 64
exit
cns id s0:0 ipaddress
interface s0:0
!Assigns IP address to s0:0
ip address slarp retry 2
exit
ip route 10.0.0.0 0.0.0.0 s0:0
!IP static route
end
```

T1 Configuration Template Example

The following example shows use of the T1 configuration template to build the configuration for use on T1:

```
hostname ${LDAP://this:attrName=IOShostname}
enable password ${LDAP://this:attrName=IOSpassword}
controller T1 0
clock source ${LDAP://this:attrName=IOST1-clocksource}
linecode ${LDAP://this:attrName=IOST1-line}
framing ${LDAP://this:attrName=IOST1-framing}
channel-group ${LDAP://this:attrName=IOST1-channel-group}
timeslots ${LDAP://this:attrName=IOST1-timeslots}
speed ${LDAP://this:attrName=IOST1-speed}
```

Voice Configuration Template Example

The following example shows use of the voice configuration template to build the configuration for using voice:

```
voice-port 1/1
codec ${LDAP://this:attrName=IOSvoice-port1}
exit
dial-peer voice 1 pots
application ${LDAP://this:attrName=IOSdial-peer1}
port 1/1
```

Remote Router Example

The following example shows a remote router configuration:

```
Router# show running-config

    Current configuration: 1659 bytes
    !
    version 12.2
    no service pad
    service timestamps debug uptime
    service timestamps log uptime
    no service password-encryption
    !
    hostname tira-24V
    !
    !
    network-clock base-rate 64k
    ip subnet-zero
    ip cef
    !
    ip audit notify log
    ip audit po max-events 100
    !
    class-map match-any voice
    match access-group 100
    !
    !
    policy-map qos
    class voice
    priority percent 70
    voice service voip
    h323
    !
    no voice confirmation-tone
    voice-card 0
    !
    !
    controller T1 0
    framing sf
    linecode ami
    !
    controller T1 1
    mode cas
    framing esf
    linecode b8zs
    ds0-group 0 timeslots 1 type e&m-immediate-start
    ds0-group 1 timeslots 2 type e&m-immediate-start
    !
    !
    interface Ethernet0
    ip address 10.1.1.2 255.255.0.0
    !
    interface Serial0
    bandwidth 1536
    ip address 10.11.11.1 255.255.255.0
    no ip mroute-cache
    load-interval 30
    clockrate 148000
    !
    ip classless
    ip route 223.255.254.254 255.255.255.0 10.3.0.1
    !
    no ip http server
```

```
ip pim bidir-enable
!
access-list 100 permit udp any range 16384 32767 any
access-list 100 permit tcp any any eq 1720
call rsvp-sync
!
voice-port 1:0
timeouts wait-release 3
!
voice-port 1:1
timeouts wait-release 3
!
!
mgcp profile default
!
dial-peer cor custom
!
dial-peer voice 1000 pots
destination-pattern 1000
port 1:0
forward-digits 0
!
dial-peer voice 1001 pots
destination-pattern 1001
no digit-strip
port 1:1
forward-digits 0
!
dial-peer voice 2000 voip
destination-pattern 2000
session target ipv4:10.11.11.2
codec g711ulaw
!
dial-peer voice 2001 voip
destination-pattern 2001
session target ipv4:10.11.11.2
signal-type ext-signal
codec g711ulaw
!
!
line con 0
line aux 0
line 2 3
line vty 0 4
```

The following example shows configuration of a serial interface to connect to and download a configuration from a Cisco IE2100 CNS configuration engine. The IE2100 IP address is 10.1.1.1. The gateway IP address to reach the 10.1.1.0 network is 10.11.11.1. The CNS default ID is the hostname, so that **cns id** command is not needed. However, the **hostname** command is key to retrieving the configuration file on the CNS configuration engine.

This configuration auto-tries ever serial interface on the remote router in turn, applies the **config-cli** commands to that interface, and tries to ping the address in the **cns config initial** command. When it succeeds, it performs a normal initial configuration.

```
! Initial basic configuration (serial interface) PPP
    cns connect serial retry-interval 1 retries 1
    config-cli ip address negotiated
    config-cli encapsulation ppp
    config-cli ip directed-broadcast
    config-cli no keepalive
    config-cli no shutdown
    exit
```

```
    hostname 26ML
    ip route 10.1.1.1 255.255.255.0 10.11.11.1
    cns config initial 10.1.1.1 no-persist
    cns inventory config
! Initial basic configuration (serial interface) HDLC
    cns config connect serial retry-interval 1 retries 1
    config-cli ip address slarp retry 1
    config-cli no shutdown
    exit
    hostname tira-36V
    ip route 10.1.1.1 255.255.255.0 10.11.11.1
    cns config initial 10.1.1.1 no-persist
    cns inventory config
    Incremental configuration (serial interface)
    cns config partial 10.1.1.1
    cns event 10.1.1.1
```

Command Scheduler Policy Lists and Occurrences: Examples

In the following example, a Command Scheduler policy named cns-weekly is configured to run two sets of EXEC CLI involving CNS commands. The policy is then scheduled with two other policies to run every seven days, one hour and thirty minutes.

```
kron policy-list cns-weekly
    cli cns image retrieve server http://10.19.2.3/week/ status
    http://10.19.2.5/status/week/
    cli cns config retrieve page /testconfig/config.asp no-persist
    exit
kron occurrence week in 7:1:30 recurring
    policy-list cns-weekly
    policy-list itd-weekly
    policy-list mkt-weekly
```

In the following example, a Command Scheduler policy named sales-may is configured to run a CNS command to retrieve a specified image from a remote server. The policy is then scheduled to run only once on May 20, at 6:30 a.m.

```
kron policy-list sales-may
cli cns image retrieve server 10.19.2.3 status 10.19.2.3
exit
kron occurrence may at 6:30 May 20 oneshot
policy-list sales-may
```

In the following example, a Command Scheduler policy named image-sunday is configured to run a CNS command to retrieve a specified image from a remote server. The policy is then scheduled to run every Sunday at 7:30 a.m.

```
kron policy-list image-sunday
    cli cns image retrieve server 10.19.2.3 status 10.19.2.3
    exit
kron occurrence sunday user sales at 7:30 sunday recurring
    policy-list image-sunday
```

In the following example, a Command Scheduler policy named file-retrieval is configured to run a CNS command to retrieve a specific file from a remote server. The policy is then scheduled to run on system startup.

```
kron policy-list file-retrieval
cli cns image retrieve server 10.19.2.3 status 10.19.2.3
exit
kron occurrence system-startup
policy-list file-retrieval
```

Retrieving a CNS Image from a Server: Example

In the following example, the CNS image agent polls a file server using the **cns image retrieve** command. Assuming that the CNS image agent is already enabled, the file server and status server paths specified here will overwrite any existing image agent server and status configuration. The new file server will be polled and a new image, if it exists, will be downloaded to the networking device.

```
cns image retrieve server https://10.19.2.3/cns/ status https://10.19.2.3/cnsstatus/
```

Retrieving a CNS Configuration from a Server: Examples

Retrieving Configuration Data from the CNS Trusted Server

The following example shows how to request a configuration from a trusted server at 10.1.1.1:

```
cns trusted-server all 10.1.1.1
exit
cns config retrieve 10.1.1.1
```

The following example shows how to request a configuration from a trusted server at 10.1.1.1 and to configure a CNS configuration retrieve interval using the **cns config retrieve** command:

```
cns trusted-server all 10.1.1.1
exit
cns config retrieve 10.1.1.1 retry 50 interval 1500
CNS Config Retrieve Attempt 1 out of 50 is in progress
Next cns config retrieve retry is in 1499 seconds (Ctrl-Shft-6 to abort this command).
..
00:26:40: %CNS-3-TRANSPORT: CNS_HTTP_CONNECTION_FAILED:10.1.1.1 -Process= "CNS config
retv", ipl= 0, pid= 43
00:26:40: %CNS-3-TRANSPORT: CNS_HTTP_CONNECTION_FAILED -Process= "CNS config retv", ipl=
0, pid= 43......
cns config retrieve 10.1.1.1
```

Applying the Retrieved Data to the Running Configuration File

The following example shows how to check and apply configuration data retrieved from the server to running configuration file only. The CNS Configuration Agent will attempt to retrieve configuration data at 30-second intervals until the attempt is successful, or is unsuccessful five times in these attempts.

```
cns config retrieve 10.1.1.1 syntax-check no-persist retry 5 interval 30
```

Overwriting the Startup Configuration File with the Retrieved Data

The following example shows how to overwrite the startup configuration file with the configuration data retrieved from the server. The configuration data will not be applied to the running configuration.

```
cns config retrieve 10.1.1.1 syntax-check no-persist retry 5 interval 30
cns config retrieve 10.1.1.1 overwrite-startup
```

Using the CNS Zero Touch Solution: Examples

Configuring PPP on a Serial Interface

The following example shows the bootstrap configuration for configuring PPP on a serial interface:

```
cns template connect ppp-serial
cli ip address negotiated
cli encapsulation ppp
cli ip directed-broadcast
cli no keepalive
exit
cns template connect ip-route
cli ip route 10.0.0.0 0.0.0.0 ${next-hop}
exit
cns connect serial-ppp ping-interval 1 retries 1
discover interface serial
template ppp-serial
template ip-route
exit
hostname 26ML
cns config initial 10.1.1.1 no-persist inventory
```

Configuring PPP on an Asynchronous Interface

The following example shows the bootstrap configuration for configuring PPP on an asynchronous interface:

```
cns template connect async
cli modem InOut
 .
 .
 .
exit
cns template connect async-interface
cli encapsulation ppp
cli ip unnumbered FastEthernet0/0
cli dialer rotary-group 0
exit
cns template connect ip-route
cli ip route 10.0.0.0 0.0.0.0 ${next-hop}
exit

cns connect async
discover line Async
template async
discover interface
template async-interface
template ip-route
exit
hostname async-example
cns config initial 10.1.1.1 no-persist inventory
```

Configuring HDLC on a Serial Interface

The following example shows the bootstrap configuration for configuring High-Level Data Link Control (HDLC) on a serial interface:

```
cns template connect hdlc-serial
cli ip address slarp retry 1
exit
cns template connect ip-route
cli ip route 0.0.0.0 0.0.0.0 ${next-hop}
exit
```

```
cns connect hdlc-serial ping-interval 1 retries 1
discover interface serial
template hdlc-serial
template ip-route
exit
hostname host1
cns config initial 10.1.1.1 no-persist inventory
```

Configuring Aggregator Router Interfaces

The following examples show how to configure a standard serial interface and a serial interface bound to a controller on an aggregator router (also known as the DCE). In order for connectivity to be established, the aggregator router must have a point-to-point subinterface configured.

Standard Serial Interface

```
interface Serial0/1
 no ip address
 encapsulation frame-relay
 frame-relay intf-type dce
exit
interface Serial0/1.1 point-to-point
 10.0.0.0 255.255.255.0
 frame-relay interface-dlci 8
```

Serial Interface Bound to a Controller

```
controller T1 0
 framing sf
 linecode ami
 channel-group 0 timeslots 1-24
exit
interface Serial0:0
 no ip address
 encapsulation frame-relay
 frame-relay intf-type dce
exit
interface Serial0:0.1 point-to-point
 ip address ip-address mask
 frame-relay interface-dlci dlci
```

Configuring IP over Frame Relay

The following example shows the bootstrap configuration for configuring IP over Frame Relay on a CPE router:

```
cns template connect setup-frame
 cli encapsulation frame-relay
 exit
cns template connect ip-over-frame
 cli frame-relay interface-dlci ${dlci}
 cli ip address dynamic
 exit
cns template connect ip-route
 cli ip route 10.0.0.0 0.0.0.0 ${next-hop}
 exit
cns connect ip-over-frame
 discover interface Serial
 template setup-frame
 discover dlci
 template ip-over-frame
 template ip-route
exit
cns config initial 10.1.1.1
```

Configuring IP over Frame Relay over T1

The following example shows the bootstrap configuration for configuring IP over Frame Relay over T1 on a CPE router:

```
cns template connect setup-frame
 cli encapsulation frame-relay
 exit
cns template connect ip-over-frame
 cli frame-relay interface-dlci ${dlci}
 cli ip address dynamic
 exit
cns template connect ip-route
 cli ip route 0.0.0.0 0.0.0.0 ${next-hop}
 exit
cns template connect t1-controller
 cli framing esf
 cli linecode b8zs
 cli channel-group 0 timeslots 1-24 speed 56
 exit
cns connect ip-over-frame-over-t1
 discover controller T1
 template t1-controller
 discover interface
 template setup-frame
 discover dlci
 template ip-over-frame
 template ip-route
exit
cns config initial 10.1.1.1
```

Additional References

The following sections provide references related to the CNS feature.

Related Documents

Related Topic	Document Title
CNS commands: complete command syntax, command mode, command history, defaults, usage guidelines, and examples	*Cisco IOS Network Management Command Reference*

Related Topic	Document Title
CNS Configuration Engine	*Cisco Intelligence Engine 2100 Configuration Registrar Manual, Release 1.1 or later* *Cisco CNS Configuration Engine Administrator's Guide*
IAD and Router Hardware and Software	• Cisco IAD2420 series hardware and software documents, available online at http://www.cisco.com/univercd/cc/td/doc/product/access/iad/iad2420/index.htm • Cisco 2600 series hardware and software documents, available online at http://www.cisco.com/univercd/cc/td/doc/product/access/acs_mod/cis2600/hw_inst/index.htm • Cisco 3600 series hardware and software documents, available online at http://www.cisco.com/univercd/cc/td/doc/product/access/acs_mod/cis3600/hw_inst/index.htm

Standards

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

MIBs

MIB	MIBs Link
The CNS Flow-Through Provisioning feature provides two mechanisms for accessing MIBs: a nongranular mechanism using SNMP encapsulation and a granular mechanism using XML encapsulation. These mechanisms enable you to access the MIBS currently available in the remote router. The MIBS currently available depend on the router platform and Cisco IOS release.	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

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

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/techsupport

Feature Information for CNS

Table 2 lists the features in this module and provides links to specific configuration information. Only features that were introduced or modified in Cisco IOS Release 12.2(1), 12.0(3)S or a later release appear in the table.

Not all commands may be available in your Cisco IOS software release. For release information about a specific command, see the command reference documentation.

Use Cisco Feature Navigator to find information about platform support and software image support. Cisco Feature Navigator enables you to determine which Cisco IOS and Catalyst OS software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Note Table 2 lists only the Cisco IOS software release that introduced support for a given feature in a given Cisco IOS software release train. Unless noted otherwise, subsequent releases of that Cisco IOS software release train also support that feature.

Table 2 Feature Information for CNS

Feature Name	Releases	Feature Information
CNS	12.2(25)S 12.2(33) SRA 12.2(33)SB 12.2(33)SXI	The CNS feature is a collection of services that can provide remote event-driven configuring of Cisco IOS networking devices and remote execution of some command-line interface (CLI) commands. The following sections provide information about this feature: • Prerequisites for CNS, page 2 • Restrictions for CNS, page 2 • Information About CNS, page 3 • CNS, page 4 • CNS Configuration Agent, page 4 • How to Configure CNS, page 17 • Configuration Examples for CNS, page 36 The following commands were introduced or modified by this feature: **clear cns config stats**, **clear cns counters**, **clear cns event stats**, **cli (cns)**, **cns config cancel**, **cns config initial**, **cns config notify**, **cns config partial**, **cns config retrieve**, **cns connect**, **cns event**, **cns exec**, **cns id**, **cns template connect**, **cns trusted-server**, **debug cns config**, **debug cns exec**, **debug cns xml-parser**, **logging cns-events**, **show cns config stats**, **show cns event connections**, **show cns event stats**, **show cns event subject**.

Table 2 Feature Information for CNS (continued)

Feature Name	Releases	Feature Information
CNS Configuration Agent	12.0(18)ST 12.0(22)S 12.2(2)T 12.2(8)T 12.2(33)SRA 12.2(33)SB 12.2(33)SXI	The CNS Configuration Agent feature supports routing devices by providing the following: • Initial configurations • Incremental (partial) configurations • Synchronized configuration updates The following sections provide information about this feature: • CNS Configuration Agent, page 4 • Initial CNS Configuration, page 4 • Incremental CNS Configuration, page 5 • Synchronized Configuration, page 5 • Configuring the CNS Event and EXEC Agents, page 20 • Troubleshooting CNS Agents, page 33 The following commands were introduced or modified by this feature: **cns config cancel**, **cns config initial**, **cns config partial**, **cns config retrieve**, **cns password**, **debug cns config**, **debug cns xml-parser**, **show cns config outstanding, show cns config stats**, **show cns config status**.
CNS Config Retrieve Enhancement with Retry and Interval	12.4(15)T 12.2(33)SRC 12.2(33)SB	The Cisco Networking Services (CNS) Config Retrieve Enhancement with Retry and Interval feature adds two options to the **cns config retrieve** command enabling you to specify an amount of time in seconds to wait before attempting to retrieve a configuration from a trusted server. The number of retries is restricted to 100 to prevent the configuration agent from indefinitely attempting to reach an unreachable server. Use the keyboard combination **Ctrl-Shift-6** to abort the **cns config retrieve** command. The following sections provide information about this feature: • CNS Config Retrieve Enhancement with Retry and Interval, page 4 • Retrieving a CNS Configuration from a Server, page 27 • Retrieving a CNS Configuration from a Server: Example, page 43 The following command was modified by this feature: **cns config retrieve**.

Table 2 Feature Information for CNS (continued)

Feature Name	Releases	Feature Information
CNS Enhanced Results Message	12.2(33)SRA 12.4(4)T	The CNS Enhanced Results Message feature sends a second CNS result message to the subject "cisco.cns.config.results" in addition to the CNS results messages sent to the CNS Event bus after a partial configuration is complete. The following sections provide information about this feature: • CNS Results Messages, page 6 • Configuring the CNS Event and EXEC Agents, page 20 • Configuring a Partial Configuration: Example, page 37 The following command was modified by this feature: **cns config partial**.
CNS Event Agent	12.0(18)ST 12.0(22)S 12.2(2)T 12.2(33)SRA 12.2(33)SB 12.2(33)SXI	The CNS Event Agent is part of the Cisco IOS infrastructure that allows Cisco IOS applications to publish and subscribe to events on a CNS Event Bus. CNS Event Agent works in conjunction with the CNS Configuration Agent feature. The following sections provide information about this feature: • CNS Event Agent, page 5 • Configuring the CNS Event and EXEC Agents, page 20 • Troubleshooting CNS Agents, page 33 The following commands were introduced or modified by this feature: **cns event**, **show cns event connections**, **show cns event stats**, **show cns event subject**.

Table 2 Feature Information for CNS (continued)

Feature Name	Releases	Feature Information
CNS Flow-Through Provisioning	12.2(2)T 12.2(2)XB 12.2(11)YT 12.2(11)YV	Cisco Networking Services (CNS) Flow-Through Provisioning provides the infrastructure for automated configuration of large numbers of network devices. Based on CNS event and config agents, it eliminates the need for an onsite technician to initialize the device. The result is an automated workflow from initial subscriber-order entry through Cisco manufacturing and shipping to final device provisioning and subscriber billing. This focuses on a root problem of today's service-provider and other similar business models: use of human labor in activating service. The following sections provide information about this feature: • Prerequisites for CNS, page 2 • CNS Flow-Through Provisioning, page 11 • Configuring the CNS Event and EXEC Agents, page 20 • CNS Flow-Through Provisioning: Examples, page 38 The following commands were introduced or modified by this feature: **cns config cancel**, **cns config connect-intf**, **cns config initial**, **cns config partial**, **cns config notify**, **cns event**, **cns id**, **cns inventory**, **cns mib-access encapsulation**, **cns notifications encapsulation**, **config-cli**, **debug cns config**, **debug cns event**, **debug cns management**, **debug cns xml-parser**, **line-cli**, **show cns config connections**, **show cns config outstanding**, **show cns event stats**, **show cns event subject**. **Note** The **cns config connect-intf** command was replaced by the **cns connect** and **cns template connect** commands in Cisco IOS Release 12.3(2)XF, 12.3(8)T, 12.3(9), 12.2(33)SRA, 12.2(31)SB2, and later releases. **Note** The **config-cli** and **line-cli** commands were replaced by the **cli (cns)** command in Cisco IOS Release 12.3(2)XF, 12.3(8)T, 12.3(9), 12.2(33)SRA, 12.2(31)SB2, and later releases.

Table 2 Feature Information for CNS (continued)

Feature Name	Releases	Feature Information
CNS Frame-Relay Zero Touch	12.3(2)XF 12.3(8)T	The CNS Frame Relay Zero Touch feature provides a CNS zero touch deployment solution over Frame Relay where the CPE router discovers its DLCI and IP address dynamically and then contacts a CNS engine to retrieve its full configuration automatically. The following sections provide information about this feature: • Restrictions for CNS, page 2 • CNS Frame Relay Zero Touch, page 15 • Deploying the CNS Router, page 17 • Using the CNS Zero Touch Solution: Examples, page 43 The following commands were introduced or modified by this feature: **cli (cns)**, **cns config connect-intf**, **cns connect,** **cns template connect**, **config-cli**, **discover (cns)**, **line-cli**, **template (cns)**. **Note** The **cns config connect-intf** command was replaced by the **cns connect** and **cns template connect** commands in Cisco IOS Releases 12.3(2)XF, 12.3(8)T, 12.3(9), 12.2(33)SRA, 12.2(31)SB2, and later releases. **Note** The **config-cli** and **line-cli** commands were replaced by the **cli (cns)** command in Cisco IOS Releases 12.3(2)XF, 12.3(8)T, 12.3(9), 12.2(33)SRA, 12.2(31)SB2, and later releases.

Table 2 Feature Information for CNS (continued)

Feature Name	Releases	Feature Information
CNS Image Agent	12.2(33)SEE 12.3(1) 12.2(31)SB2 12.2(33)SRB 12.2(33)SB 12.2(33)SXI	The CNS Image Agent feature is an infrastructure in Cisco IOS software to enable automated installation and activation of Cisco IOS images on Cisco IOS networking devices. The following sections provide information about this feature: • Prerequisites for CNS, page 2 • Restrictions for CNS, page 2 • CNS Image Agent, page 5 • Configuring the CNS Image Agent, page 23 • Retrieving a CNS Image from a Server, page 26 • Troubleshooting CNS Agents, page 33 • Enabling and Configuring CNS Agents: Example, page 37 • Retrieving a CNS Image from a Server: Example, page 42 The following commands were introduced or modified by this feature: **clear cns image connections**, **clear cns image status**, **cns id**, **cns image**, **cns image password**, **cns image retrieve**, **cns image retry**, **debug cns image**, **show cns image connections**, **show cns image inventory**, **show cns image status**.
CNS Interactive CLI	12.0(28)S 12.2(18)SXE 12.2(18)SXF2 12.2(33)SRC 12.2(33)SXI	The CNS Interactive CLI feature introduces a new XML interface that allows you to send interactive commands to a router, such as commands that generate prompts for user input. The following section provides information about this feature: • CNS Interactive CLI, page 10
CNS Security Enhancement	12.4(9)T 12.2(33)SRA	The CNS Security Enhancement feature improves the security of Cisco Networking Services (CNS) messages by authenticating sender credentials through the use of the Service-Oriented Access Protocol (SOAP) message format. The following sections provide information about this feature: • CNS Message Formats, page 6 • CNS Security Enhancement, page 9 • Configuring CNS Security Features, page 25 The following commands were introduced or modified by this feature: **cns aaa authentication**, **cns message format notification**.

Table 2 Feature Information for CNS (continued)

Feature Name	Releases	Feature Information
CNS Zero Touch	12.3(9)	The CNS Zero Touch feature provides a zero touch deployment solution where the router contacts a CNS configuration engine to retrieve its full configuration automatically. The following sections provide information about this feature: • Prerequisites for CNS, page 2 • Restrictions for CNS, page 2 • CNS Zero Touch, page 15 • Deploying the CNS Router, page 17 • Using the CNS Zero Touch Solution: Examples, page 43 The following commands were introduced or modified by this feature: **cli (cns)**, **cns config connect-intf**, **cns connect**, **cns template connect**, **config-cli**, **discover (cns)**, **line-cli, template (cns)**. **Note** The **cns config connect-intf** command was replaced by the **cns connect** and **cns template connect** commands in Cisco IOS Release 12.3(2)XF, 12.3(8)T, 12.3(9), 12.2(33)SRA, 12.2(31)SB2, and later releases. **Note** The **config-cli** and **line-cli** commands were replaced by the **cli (cns)** command in Cisco IOS Release 12.3(2)XF, 12.3(8)T, 12.3(9), 12.2(33)SRA, 12.2(31)SB2, and later releases.
Command Scheduler	12.3(1) 12.2(33)SRA 12.2(33)SRC 12.2(33)SB 12.2(33)SXI	The Command Scheduler feature provides the ability to schedule some EXEC command-line interface (CLI) commands to run at specific times or at specified intervals. The following sections provide information about this feature: • Restrictions for CNS, page 2 • Command Scheduler, page 10 • Configuring Command Scheduler Policy Lists and Occurrences, page 28 • Command Scheduler Policy Lists and Occurrences: Examples, page 41 The following commands were introduced or modified by this feature: **cli**, **debug kron**, **kron occurrence**, **kron policy-list**, **policy-list**, **show kron schedule**.

Network Configuration Protocol

First Published: February 28, 2007
Last Updated: June 15, 2009

The Network Configuration Protocol (NETCONF) defines a simple mechanism through which a network device can be managed, configuration data information can be retrieved, and new configuration data can be uploaded and manipulated. NETCONF uses Extensible Markup Language (XML)-based data encoding for the configuration data and protocol messages.

You can use the NETCONF over SSHv2 feature to perform network configurations via the Cisco command-line interface (CLI) over an encrypted transport. The NETCONF Network Manager, which is the NETCONF client, must use Secure Shell Version 2 (SSHv2) as the network transport to the NETCONF server. Multiple NETCONF clients can connect to the NETCONF server.

You can use the NETCONF over BEEP feature to send notifications of any configuration change over NETCONF. A notification is an event indicating that a configuration change has happened. The change can be a new configuration, deleted configuration, or changed configuration. The notifications are sent at the end of a successful configuration operation as one message showing the set of changes, rather than individual messages for each line in the configuration that is changed.

Blocks Extensible Exchange Protocol (BEEP) can use the Simple Authentication and Security Layer (SASL) profile to provide simple and direct mapping to the existing security model. Alternatively, NETCONF over BEEP can use the transport layer security (TLS) to provide a strong encryption mechanism with either server authentication or server and client-side authentication.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the "Feature Information for NETCONF" section on page 30.

Use Cisco Feature Navigator to find information about platform support and Cisco IOS and Catalyst OS software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

Contents

Prerequisites for NETCONF

- NETCONF over SSHv2 requires that a vty line be available for each NETCONF session as specified in the **netconf max-session** command.
- A vty line must be available for each NETCONF session as specified by the **netconf max-session** command.
- NETCONF over BEEP listeners require SASL to be configured.

Restrictions for NETCONF

- NETCONF SSHv2 supports a maximum of 16 concurrent sessions.
- Only SSH version 2 is supported.
- You must be running a crypto image in order to configure BEEP using TLS.

Information About NETCONF

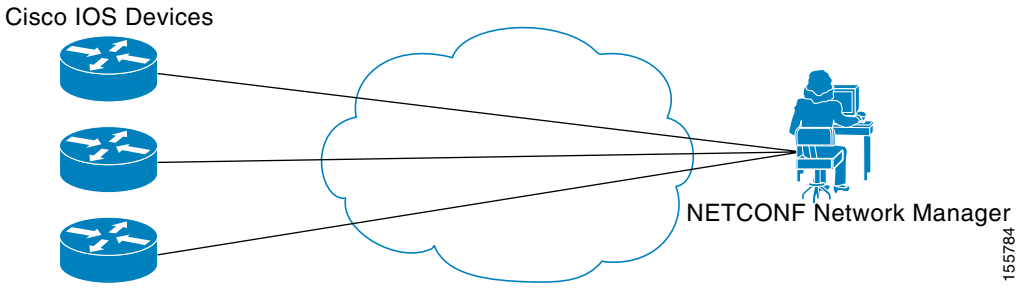
To configure NETCONF, you should understand the following concepts:

NETCONF over SSHv2

To run the NETCONF over SSHv2 feature, the client (a Cisco device running Cisco IOS software) establishes an SSH transport connection with the server (a NETCONF network manager). Figure 1 shows a basic NETCONF over SSHv2 network configuration. The client and server exchange keys for security and password encryption. The user ID and password of the SSHv2 session running NETCONF are used for authorization and authentication purposes. The user privilege level is enforced and the client session may not have full access to the NETCONF operations if the privilege level is not high enough.

If authentication, authorization, and accounting (AAA) is configured, the AAA service is used as if a user had established an SSH session directly to the device. Using the existing security configuration makes the transition to NETCONF almost seamless. Once the client has been successfully authenticated, the client invokes the SSH connection protocol and the SSH session is established. After the SSH session is established, the user or application invokes NETCONF as an SSH subsystem called “netconf.”

Figure 1 NETCONF over SSHv2



Secure Shell Version 2

SSHv2 runs on top of a reliable transport layer and provides strong authentication and encryption capabilities. SSHv2 provides a means to securely access and securely execute commands on another computer over a network.

NETCONF does not support SSH version 1. The configuration for the SSH Version 2 server is similar to the configuration for SSH version 1. Use the **ip ssh version** command to specify which version of SSH that you want to configure. If you do not configure this command, SSH by default runs in compatibility mode; that is, both SSH version 1 and SSH version 2 connections are honored.

Note SSH version 1 is a protocol that has never been defined in a standard. If you do not want your router to fall back to the undefined protocol (version 1), you should use the **ip ssh version** command and specify version 2.

Use the **ip ssh rsa keypair-name** command to enable an SSH connection using Rivest, Shamir, and Adelman (RSA) keys that you have configured. If you configure the **ip ssh rsa keypair-name** command with a key-pair name, SSH is enabled if the key pair exists, or SSH will be enabled if the key pair is generated later. If you use this command to enable SSH, you do not need to configure a hostname and a domain name.

NETCONF over BEEP

The NETCONF over BEEP feature allows you to enable BEEP as the transport protocol to use during NETCONF sessions. Using NETCONF over BEEP, you can configure either the NETCONF server or the NETCONF client to initiate a connection, thus supporting large networks of intermittently connected devices, and those devices that must reverse the management connection where there are firewalls and Network Address Translators (NATs).

BEEP is a generic application protocol framework for connection-oriented, asynchronous interactions. It is intended to provide the features that traditionally have been duplicated in various protocol implementations. BEEP typically runs on top of TCP and allows the exchange of messages. Unlike HTTP and similar protocols, either end of the connection can send a message at any time. BEEP also includes facilities for encryption and authentication and is highly extensible.

The BEEP protocol contains a framing mechanism that permits simultaneous and independent exchanges of messages between peers. These messages are usually structured using XML. All exchanges occur in the context of a binding to a well-defined aspect of the application, such as transport security, user authentication, or data exchange. This binding forms a channel; each channel has an associated profile that defines the syntax and semantics of the messages exchanged.

The BEEP session is mapped onto the NETCONF service. When a session is established, each BEEP peer advertises the profiles it supports. During the creation of a channel, the client (the BEEP initiator) supplies one or more proposed profiles for that channel. If the server (the BEEP listener) creates the channel, it selects one of the profiles and sends it in a reply. The server may also indicate that none of the profiles are acceptable, and decline creation of the channel.

BEEP allows multiple data exchange channels to be simultaneously in use.

Although BEEP is a peer-to-peer protocol, each peer is labelled according to the role it is performing at a given time. When a BEEP session is established, the peer that awaits new connections is the BEEP listener. The other peer, which establishes a connection to the listener, is the BEEP initiator. The BEEP peer that starts an exchange is the client, and the other BEEP peer is the server. Typically, a BEEP peer that acts in the server role also performs in the listening role. However, because BEEP is a peer-to-peer protocol, the BEEP peer that acts in the server role is not required to also perform in the listening role.

Simple Authentication and Security Layer

The SASL is an Internet standard method for adding authentication support to connection-based protocols. SASL can be used between a security appliance and an Lightweight Directory Access Protocol (LDAP) server to secure user authentication.

Transport Layer Security

The TLS is an application-level protocol that provides for secure communication between a client and server by allowing mutual authentication, the use of hash for integrity, and encryption for privacy. TLS relies upon certificates, public keys, and private keys.

Certificates are similar to digital ID cards. They prove the identity of the server to clients. Each certificate includes the name of the authority that issued it, the name of the entity to which the certificate was issued, the entity's public key, and time stamps that indicate the certificate's expiration date.

Public and private keys are the ciphers used to encrypt and decrypt information. Although the public key is shared, the private key is never given out. Each public-private key pair works together. Data encrypted with the public key can be decrypted only with the private key.

Access Lists

You can optionally configure access lists for use with NETCONF over SSHv2 sessions. An access list is a sequential collection of permit and deny conditions that apply to IP addresses. The Cisco IOS software tests addresses against the conditions in an access list one by one. The first match determines whether the software accepts or rejects the address. Because the software stops testing conditions after the first match, the order of the conditions is critical. If no conditions match, the software rejects the address.

The two main tasks involved in using access lists are as follows:

1. Creating an access list by specifying an access list number or name and access conditions.
2. Applying the access list to interfaces or terminal lines.

For more information about configuring access lists, see IP Access List Overview and Creating an IP Access List and Applying It to an Interface modules in the *Cisco IOS Security Configuration Guide: Securing the Data Plane*.

NETCONF Notifications

NETCONF sends notifications of any configuration change over NETCONF. A notification is an event indicating that a configuration change has occurred. The change can be a new configuration, deleted configuration, or changed configuration. The notifications are sent at the end of a successful configuration operation as one message that shows the set of changes rather than showing individual messages for each line that is changed in the configuration.

How to Configure NETCONF

This section contains the following tasks:

Enabling SSH Version 2 Using a Hostname and Domain Name

Perform this task to configure your router for SSH version 2 using a hostname and domain name. You may also configure SSH version 2 by using the RSA key pair configuration (see "Enabling SSH Version 2 Using RSA Key Pairs" section on page 6).

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **hostname** *hostname*
4. **ip domain-name** *name*

5. **crypto key generate rsa**
6. **ip ssh** [**timeout** *seconds* | **authentication-retries** *integer*]
7. **ip ssh version 2**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**hostname** *hostname* **Example:** `Router(config)# hostname host1`	Configures a hostname for your router.
Step 4	**ip domain-name** *name* **Example:** `Router(config)# ip domain-name domain1.com`	Configures a domain name for your router.
Step 5	**crypto key generate rsa** **Example:** `Router(config)# crypto key generate rsa`	Enables the SSH server for local and remote authentication.
Step 6	**ip ssh** [**timeout** *seconds* | **authentication-retries** *integer*] **Example:** `Router(config)# ip ssh timeout 120`	(Optional) Configures SSH control variables on your router.
Step 7	**ip ssh version 2** **Example:** `Router(config)# ip ssh version 2`	Specifies the version of SSH to be run on your router.

Enabling SSH Version 2 Using RSA Key Pairs

Perform this task to enable SSH version 2 without configuring a hostname or domain name. SSH version 2 will be enabled if the key pair that you configure already exists or if it is generated later. You may also configure SSH version 2 by using the hostname and domain name configuration. (See ""Enabling SSH Version 2 Using a Hostname and Domain Name" section on page 5.)

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip ssh rsa keypair-name** *keypair-name*
4. **crypto key generate rsa usage-keys label** *key-label* **modulus** *modulus-size*
5. **ip ssh** [**timeout** *seconds* | **authentication-retries** *integer*]
6. **ip ssh version 2**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**ip ssh rsa keypair-name** *keypair-name* **Example:** `Router(config)# ip ssh rsa keypair-name sshkeys`	Specifies which RSA keypair to use for SSH usage. **Note** A Cisco IOS router can have many RSA key pairs.
Step 4	**crypto key generate rsa usage-keys label** *key-label* **modulus** *modulus-size* **Example:** `Router(config)# crypto key generate rsa usage-keys label sshkeys modulus 768`	Enables the SSH server for local and remote authentication on the router. For SSH version 2, the modulus size must be at least 768 bits. **Note** To delete the RSA key pair, use the **crypto key zeroize rsa** command. After you have deleted the RSA command, you automatically disable the SSH server.
Step 5	**ip ssh** [**timeout** *seconds* | **authentication-retries** *integer*] **Example:** `Router(config)# ip ssh timeout 120`	Configures SSH control variables on your router.
Step 6	**ip ssh version 2** **Example:** `Router(config)# ip ssh version 2`	Specifies the version of SSH to be run on a router.

Starting an Encrypted Session with a Remote Device

Perform this task to start an encrypted session with a remote networking device. (You do not have to enable your router. SSH can be run in disabled mode.)

From any UNIX or UNIX-like device, the following command is typically used to form an SSH session:

```
ssh -2 -s user@router.example.com netconf
```

SUMMARY STEPS

1. **ssh** [**-v** {**1** | **2**}] [**-c** {**3des** | **aes128-cbc** | **aes192-cbc** | **aes256-cbc**}] [**-m** {**hmac-md5** | **hmac-md5-96** | **hmac-sha1** | **hmac-sha1-96**}] [**l** *userid*] [**-o numberofpasswordprompts** *n*] [**-p** *port-num*] {*ip-addr* | *hostname*} [*command*]

DETAILED STEPS

	Command or Action	Purpose

Step 1	**ssh** [**-v** {**1** | **2**}] [**-c** {**3des** | **aes128-cbc** | **aes192-cbc** | **aes256-cbc**}] [**-m** {**hmac-md5** | **hmac-md5-96** | **hmac-sha1** | **hmac-sha1-96**}] [**l** *userid*] [**-o numberofpasswordprompts** *n*] [**-p** *port-num*] {*ip-addr* | *hostname*} [*command*]
	
	Example:
	`Router# ssh -v 2 -c aes256-cbc -m hmac-sha1-96 -l user2 10.76.82.24`
	
	or
	
	`Router# ssh -v 2 -c aes256-cbc -m hmac-sha1-96 user2@10.76.82.24`
		Starts an encrypted session with a remote networking device.
		
		The first example adheres to the SSH version 2 conventions. A more natural and common way to start a session is by linking the username with the hostname. For example, the second configuration example provides an end result that is identical to that of the first example.

Troubleshooting Tips

The **ip ssh version** command can be used for troubleshooting your SSH configuration. By changing versions, you can determine which SSH version has a problem.

What to Do Next

For more information about the **ssh** command, see the *Cisco IOS Security Command Reference*.

Verifying the Status of the Secure Shell Connection

Perform this task to display the status of the SSH connection on your router.

SUMMARY STEPS

1. **enable**
2. **show ssh**
3. **show ip ssh**

Note You can use the following **show** commands in user EXEC or privileged EXEC mode.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	(Optional) Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show ssh** **Example:** `Router# show ssh`	Displays the status of SSH server connections.
Step 3	**show ip ssh** **Example:** `Router# show ip ssh`	Displays the version and configuration data for SSH.

Examples

The following output from the **show ssh** command displays status about SSH version 2 connections.

```
Router# show ssh

Connection Version Mode Encryption  Hmac                State
Username
1          2.0     IN   aes128-cbc  hmac-md5     Session started       lab
1          2.0     OUT  aes128-cbc  hmac-md5     Session started       lab
%No SSHv1 server connections running.
```

The following output from the **show ip ssh** command displays the version of SSH that is enabled, the authentication timeout values, and the number of authentication retries.

```
Router# show ip ssh

SSH Enabled - version 2.0
Authentication timeout: 120 secs; Authentication retries: 3
```

Enabling NETCONF over SSHv2

Perform this task to enable NETCONF over SSHv2.

Prerequisites

SSHv2 must be enabled.

Note There must be at least as many vty lines configured as there are concurrent NETCONF sessions.

Restrictions

- A minimum of four concurrent NETCONF sessions must be configured.
- A maximum of 16 concurrent NETCONF sessions can be configured.
- NETCONF does not support SSHv1.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **netconf ssh** [**acl** *access-list-number*]
4. **netconf lock-time** *seconds*
5. **netconf max-sessions** *session*
6. **netconf max-message** *size*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**netconf ssh** [**acl** *access-list-number*] **Example:** `Router(config)# netconf ssh acl 1`	Enables NETCONF over SSHv2. • Optionally, you can configure an access control list for this NETCONF session.
Step 4	**netconf lock-time** *seconds* **Example:** `Router(config)# netconf lock-time 60`	(Optional) Specifies the maximum time, in seconds, a NETCONF configuration lock is in place without an intermediate operation. • The valid range is 1 to 300. The default value is 10 seconds.

	Command or Action	Purpose
Step 5	**netconf max-sessions** *session* **Example:** `Router(config)# netconf max-sessions 5`	(Optional) Specifies the maximum number of concurrent NETCONF sessions allowed. • The valid range is 4 to 16. The default value is 4.
Step 6	**netconf max-message** size **Example:** `Router(config)# netconf max-message 37283`	(Optional) Specifies the maximum size, in kilobytes (KB), for the messages received in a NETCONF session. • The valid range is 1 to 2147483. The default value is infinite. • To set the maximum size to infinite, use the **no netconf max-message** command.

Configuring an SASL Profile

To enable NETCONF over BEEP using SASL, you must first configure an SASL profile, which specifies which users are allowed access into the router. Perform this task to configure an SASL profile.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **sasl profile** *profile-name*
4. **mechanism digest-md5**
5. **server** *user-name* **password** *password*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**sasl profile** *profile-name* **Example:** `Router(config)# sasl profile beep`	Configures an SASL profile and enters SASL profile configuration mode.
Step 4	**mechanism digest-md5** **Example:** `Router(config-SASL-profile)# mechanism digest-md5`	Configures the SASL profile mechanism.
Step 5	**server** *user-name* **password** *password* **Example:** `Router(config-SASL-profile)# server user1 password password1`	Configures an SASL server.

Enabling NETCONF over BEEP

Perform this task to enable NETCONF over BEEP.

Prerequisites

- There must be at least as many vty lines configured as there are concurrent NETCONF sessions.
- If you configure NETCONF over BEEP using SASL, you must first configure an SASL profile.

Restrictions

- A minimum of four concurrent NETCONF sessions must be configured.
- A maximum of 16 concurrent NETCONF sessions can be configured.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto key generate rsa general-keys**

4. **crypto pki trustpoint** *name*
5. **enrollment url** *url*
6. **subject-name** *name*
7. **revocation-check** *method1* [*method2*[*method3*]]
8. **exit**
9. **crypto pki authenticate** *name*
10. **crypto pki enroll** *name*
11. **netconf lock-time** *seconds*
12. **line vty** *line-number* [*ending-line-number*]
13. **netconf max-sessions** *session*
14. **netconf beep initiator** {*hostname* | *ip-address*} *port-number* **user** *sasl-user* **password** *sasl-password* [**encrypt** *trustpoint*] [**reconnect-time** *seconds*]
15. **netconf beep listener** [*port-number*] [**acl** *access-list-number*] [**sasl** *sasl-profile*] [**encrypt** *trustpoint*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**crypto key generate rsa general-keys** **Example:** `Router(config)# crypto key generate rsa general-keys`	Generates RSA key pairs and specifies that the general-purpose key pair should be generated. Perform this step only once.
Step 4	**crypto pki trustpoint** *name* **Example:** `Router(config)# crypto pki trustpoint my_trustpoint`	Declares the trustpoint that your router should use and enters ca-trustpoint configuration mode.
Step 5	**enrollment url** *url* **Example:** `Router(ca-trustpoint)# enrollment url http://10.2.3.3:80`	Specifies the enrollment parameters of a certification authority (CA).

	Command or Action	Purpose
Step 6	**subject-name** *name* **Example:** `Router(ca-trustpoint)# subject-name CN=dns_name_of_host.com`	Specifies the subject name in the certificate request. **Note** The subject name should be the Domain Name System (DNS) name of the device.
Step 7	**revocation-check** *method1* [*method2*[*method3*]] **Example:** `Router(ca-trustpoint)# revocation-check none`	Checks the revocation status of a certificate.
Step 8	**exit** **Example:** `Router(ca-trustpoint)# exit`	Exits ca-trustpoint configuration mode and returns to global configuration mode.
Step 9	**crypto pki authenticate** *name* **Example:** `Router(config)# crypto pki authenticate my_trustpoint`	Authenticates the certification authority (by getting the certificate of the CA).
Step 10	**crypto pki enroll** *name* **Example:** `Router(config)# crypto pki enroll my_trustpoint`	Obtains the certificate or certificates for your router from CA.
Step 11	**netconf lock-time** *seconds* **Example:** `Router(config)# netconf lock-time 60`	(Optional) Specifies the maximum time a NETCONF configuration lock is in place without an intermediate operation. The valid value range for the seconds argument is 1 to 300 seconds. The default value is 10 seconds.
Step 12	**line vty** line-number [*ending-line-number*] **Example:** `Router(config)# line vty 0 15`	Identifies a specific virtual terminal line for remote console access. You must configure the same number of vty lines as maximum NETCONF sessions.
Step 13	**netconf max-sessions** *session* **Example:** `Router(config)# netconf max-sessions 16`	(Optional) Specifies the maximum number of concurrent NETCONF sessions allowed.

	Command or Action	Purpose
Step 14	**netconf beep initiator** {*hostname* | *ip-address*} *port-number* **user** *sasl-user* **password** *sasl-password* [**encrypt** *trustpoint*] [**reconnect-time** *seconds*] **Example:** `Router(config)# netconf beep initiator host1 23 user user1 password password1 encrypt 23 reconnect-time 60`	(Optional) Specifies BEEP as the transport protocol for NETCONF sessions and configures a peer as the BEEP initiator. **Note** Perform this step to configure a NETCONF BEEP initiator session. You can also optionally configure a BEEP listener session.
Step 15	**netconf beep listener** [*port-number*] [**acl** *access-list-number*] [**sasl** *sasl-profile*] [**encrypt** *trustpoint*] **Example:** `Router(config)# netconf beep listener 26 acl 101 sasl profile1 encrypt 25`	(Optional) Specifies BEEP as the transport protocol for NETCONF and configures a peer as the BEEP listener. **Note** Perform this step to configure a NETCONF BEEP listener session. You can also optionally configure a BEEP initiator session.

Configuring the NETCONF Network Manager Application

Step 1 Use the following CLI string to configure the NETCONF Network Manager application to invoke NETCONF as an SSH subsystem:

```
Unix Side: ssh-2 -s companyname@10.1.1.1 netconf
```

Step 2 As soon as the NETCONF session is established, indicate the server capabilities by sending an XML document containing a <hello>:

```
<?xml version="1.0" encoding="UTF-8"?>
    <hello>
      <capabilities>
        <capability>
            urn:ietf:params:xml:ns:netconf:base:1.0
          </capability>
          <capability>
            urn:ietf:params:ns:netconf:capability:startup:1.0
          </capability>
       </capabilities>
    <session-id>4<session-id>
</hello>]]>]]>
```

The client also responds by sending an XML document containing a <hello>:

```
<?xml version="1.0" encoding="UTF-8"?>
 <hello>
   <capabilities>
       <capability>
           urn:ietf:params:xml:ns:netconf:base:1.0
     </capability>
    </capabilities>
</hello>]]>]]>
```

Note Although the example shows the server sending a <hello> message followed by the client's message, both sides send the message as soon as the NETCONF subsystem is initialized, perhaps simultaneously.

Tip All NETCONF requests must end with]]>]]> which denotes an end to the request. Until the]]>]]> sequence is sent, the device will not process the request.

See "Configuring NETCONF over SSHv2: Example" section on page 23 for a specific example.

Step 3 Use the following XML string to enable the NETCONF network manager application to send and receive NETCONF notifications:

```
<?xml version="1.0" encoding="UTF-8" ?>
<rpc message-id="9.0"><notification-on/>
</rpc>]]>]]>
```

Step 4 Use the following XML string to stop the NETCONF network manager application from sending or receiving NETCONF notifications:

```
<?xml version="1.0" encoding="UTF-8" ?>
<rpc message-id="9.13"><notification-off/>
</rpc>]]>]]>
```

Delivering NETCONF Payloads

Use the following XML string to deliver the NETCONF payload to the network manager application:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://www.cisco.com/cpi_10/schema"
elementFormDefault="qualified" attributeFormDefault="unqualified"
xmlns="http://www.cisco.com/cpi_10/schema" xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <!--The following elements define the cisco extensions for the content of the filter
element in a <get-config> request. They allow the client to specify the format of the
response and to select subsets of the entire configuration to be included.-->
   <xs:element name="config-format-text-block">
      <xs:annotation>
         <xs:documentation>If this element appears in the filter, then the cllient is
requesting that the response data be sent in config command block
format.</xs:documentation>
      </xs:annotation>
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="text-filter-spec" minOccurs="0"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:element name="config-format-text-cmd">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="text-filter-spec"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:element name="config-format-xml">
      <xs:annotation>
         <xs:documentation>When this element appears in the filter of a get-config
request, the results are to be returned in E-DI XML format. The content of this element is
treated as a filter.</xs:documentation>
      </xs:annotation>
      <xs:complexType>
         <xs:complexContent>
            <xs:extension base="xs:anyType"/>
```

```
         </xs:complexContent>
      </xs:complexType>
   </xs:element>
   <!--These elements are used in the filter of a <get> to specify operational data to
return.-->
   <xs:element name="oper-data-format-text-block">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="show" type="xs:string" maxOccurs="unbounded"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:element name="oper-data-format-xml">
      <xs:complexType>
         <xs:sequence>
            <xs:any/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <!--When confing-format-text format is specified, the following describes the content
of the data element in the response-->
   <xs:element name="cli-config-data">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="cmd" type="xs:string" maxOccurs="unbounded">
               <xs:annotation>
                  <xs:documentation>Content is a command. May be multiple
lines.</xs:documentation>
               </xs:annotation>
            </xs:element>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:element name="cli-config-data-block" type="xs:string">
      <xs:annotation>
         <xs:documentation>The content of this element is the device configuration as it
would be sent to a terminal session. It contains embedded newline characters that must be
preserved as they represent the boundaries between the individual command
lines</xs:documentation>
      </xs:annotation>
   </xs:element>
   <xs:element name="text-filter-spec">
      <xs:annotation>
         <xs:documentation>If this element is included in the config-format-text element,
then the content is treated as if the string was appended to the "show running-config"
command line.</xs:documentation>
      </xs:annotation>
   </xs:element>
   <xs:element name="cli-oper-data-block">
      <xs:complexType>
         <xs:annotation>
            <xs:documentation> This element is included in the response to get operation.
Content of this element is the operational data in text format.</xs:documentation>
         </xs:annotation>
         <xs:sequence>
            <xs:element name="item" maxOccurs="unbounded">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="show"/>
                     <xs:element name="response"/>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
```

```
      </xs:complexType>
   </xs:element>
   <xs:schema>
```

Formatting NETCONF Notifications

The NETCONF network manager application uses .xsd schema files to describe the format of the XML NETCONF notification messages being sent between a NETCONF network manager application and a router running NETCONF over SSHv2 or BEEP. These files can be displayed in a browser or a schema reading tool. You can use these schema to validate that the XML is correct. These schema describe the format, not the content, of the data being exchanged.

NETCONF uses the <edit-config> function to load all of a specified configuration to a specified target configuration. When this new configuration is entered, the target configuration is not replaced. The target configuration is changed according to the data and requested operations of the requesting source.

The following are schemas for the NETCONF <edit-config> function in CLI, CLI block, and XML format.

NETCONF <edit-config> Request: CLI Format

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <edit-config>
      <target>
         <running/>
      </target>
      <config>
         <cli-config-data>
           <cmd>hostname test</cmd>
            <cmd>interface fastEthernet0/1</cmd>
            <cmd>ip address 192.168.1.1 255.255.255.0</cmd>
       </cli-config-data>
      </config>
   </edit-config>
</rpc>]]>]]>
```

NETCONF <edit-config> Response: CLI Format

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:netconf:base:1.0">
   <ok/>
</rpc-reply>]]>]]>
```

NETCONF <edit-config> Request: CLI-Block Format

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="netconf.mini.edit.3">
   <edit-config>
      <target>
         <running/>
      </target>
      <config>
         <cli-config-data-block>
            hostname bob
            interface fastEthernet0/1
            ip address 192.168.1.1 255.255.255.0
         </cli-config-data-block>
      </config>
   </edit-config>
</rpc>]]>]]>
```

NETCONF <edit-config> Response: CLI-Block Format

```
<?xml version="1.0" encoding=\"UTF-8\"?>
<rpc-reply message-id="netconf.mini.edit.3" xmlns="urn:ietf:params:netconf:base:1.0">
   <ok/>
</rpc-reply>]]>]]>
```

The following are schemas for the NETCONF <get-config> function in CLI and CLI-block format.

NETCONF <get-config> Request: CLI Format

```
<?xml version="1.0" encoding=\"UTF-8\"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <get-config>
      <source>
         <running/>
      </source>
      <filter>
         <config-format-text-cmd>
            <text-filter-spec> | inc interface </text-filter-spec>
         </config-format-text-cmd>
</filter>
   </get-config>
</rpc>]]>]]>
```

NETCONF <get-config> Response: CLI Format

```
<?xml version="1.0" encoding=\"UTF-8\"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <data>
      <cli-config-data>
         <cmd>interface FastEthernet0/1</cmd>
         <cmd>interface FastEthernet0/2</cmd>
      </cli-config-data>
   </data>
</rpc-reply>]]>]]>
```

NETCONF <get-config> Request: CLI-Block Format

```
<?xml version="1.0" encoding=\"UTF-8\"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <get-config>
      <source>
         <running/>
      </source>
      <filter>
         <config-format-text-block>
            <text-filter-spec> | inc interface </text-filter-spec>
         </config-format-text-block>
      </filter>
   </get-config>
</rpc>]]>]]>
```

NETCONF <get-config> Response: CLI-Block Format

```
<?xml version="1.0" encoding=\"UTF-8\"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <data>
      <cli-config-data-block>
         interface FastEthernet0/1
         interface FastEthernet0/2
      </cli-config-data-block>
   </data>
</rpc-reply>]]>]]>
```

NETCONF uses the <get> function to retrieve configuration and device-state information. The NETCONF <get> format is the equivalent of a Cisco IOS **show** command. The <filter> parameter specifies the portion of the system configuration and device-state data to retrieve. If the <filter> parameter is empty, nothing is returned.

The following are schemas for the <get> function in CLI and CLI-block format.

NETCONF <get> Request: CLI Format

```
<?xml version="1.0" encoding=\"UTF-8\"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <get>
       <filter>
          <config-format-text-cmd>
             <text-filter-spec> | include interface </text-filter-spec>
          </config-format-text-cmd>
          <oper-data-format-text-block>
             <show>interfaces</show>
             <show>arp</show>
          </oper-data-format-text-block>
       </filter>
    </get>
 </rpc>]]>]]>
```

NETCONF <get> Response: CLI Format

```
<?xml version="1.0" encoding=\"UTF-8\"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
       <cli-config-data>
         <cmd>interface Loopback0</cmd>
         <cmd>interface GigabitEthernet0/1</cmd>
         <cmd>interface GigabitEthernet0/2</cmd>
       </cli-config-data>
       <cli-oper-data-block>
          <item>
             <show>interfaces</show>
             <response>
                <!-- output of "show interfaces" ------>
             </response>
          <show>arp</show>
          <item>
             <show>arp</show>
             <response>
                <!-- output of "show arp" ------>
             </response>
          </item>
       </cli-oper-data-block>
    </data>
</rpc-reply>]]>]]>
```

NETCONF <get> Request: CLI-Block Format

```
<?xml version="1.0" encoding=\"UTF-8\"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <get>
       <filter>
          <config-format-text-block>
             <text-filter-spec> | include interface </text-filter-spec>
          </config-format-text-block>
          <oper-data-format-text-block>
             <show>interfaces</show>
             <show>arp</show>
          </oper-data-format-text-block>
```

```
      </filter>
   </get>
</rpc>]]>]]>
```

NETCONF <get> Response: CLI-Block Format

```
<?xml version="1.0" encoding=\"UTF-8\"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
       <cli-config-data-block>
        interface Loopback0
        interface GigabitEthernet0/1
        interface GigabitEthernet0/2

       </cli-config-data-block>
       <cli-oper-data-block>
          <item>
             <show>interfaces</show>
             <response>
                <!-- output of "show interfaces" ------>
             </response>
          <show>arp</show>
          <item>
             <show>arp</show>
             <response>
                <!-- output of "show arp" ------>
             </response>
          </item>
       </cli-oper-data-block>
    </data>
</rpc-reply>]]>]]>
```

Monitoring and Maintaining NETCONF Sessions

Perform this task to monitor and maintain NETCONF sessions.

Restrictions

- A minimum of four concurrent NETCONF sessions must be configured.
- A maximum of 16 concurrent NETCONF sessions can be configured.
- NETCONF does not support SSHv1.

SUMMARY STEPS

1. **enable**
2. **show netconf** {**counters** | **session** | **schema**}
3. **debug netconf** {**all** | **error**}
4. **clear netconf** {**counters** | **sessions**}

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show netconf** {**counters** | **session** | **schema**} **Example:** `Router# show netconf counters`	Displays NETCONF information.
Step 3	**debug netconf** {**all** | **error**} **Example:** `Router# debug netconf error`	Enables debugging of NETCONF sessions.
Step 4	**clear netconf** {**counters** | **sessions**} **Example:** `Router# clear netconf sessions`	Clears NETCONF statistics counters and NETCONF sessions, and frees associated resources and locks.

Configuration Examples for NETCONF

This section provides the following configuration examples:

Enabling SSHv2 Using a Hostname and Domain Name: Example

The following example shows how to configure SSHv2 using a hostname and a domain name:

```
Router# configure terminal
Router(config)# hostname host1
Router(config)# ip domain-name domain1.com
Router(config)# crypto key generate rsa
Router(config)# ip ssh timeout 120
Router(config)# ip ssh version 2
```

Enabling Secure Shell Version 2 Using RSA Keys: Example

The following example shows how to configure SSHv2 using RSA keys:

```
Router# configure terminal
Router(config)# ip ssh rsa keypair-name sshkeys
Router(config)# crypto key generate rsa usage-keys label sshkeys modulus 768
Router(config)# ip ssh timeout 120
Router(config)# ip ssh version 2
```

Starting an Encrypted Session with a Remote Device: Example

The following example shows how to start an encrypted SSH session with a remote networking device, from any UNIX or UNIX-like device:

```
Router(config)# ssh -2 -s user@router.example.com netconf
```

Configuring NETCONF over SSHv2: Example

The following example shows how to configure NETCONF over SSHv2:

```
Router# configure terminal
Router(config)# netconf ssh acl 1
Router(config)# netconf lock-time 60
Router(config)# netconf max-sessions 5
Router(config)# netconf max-message 2345
Router# ssh-2 -s username@10.1.1.1 netconf
```

The following example shows how to get the configuration for loopback interface 113.

Step 1 First, send the “hello”:

```
<?xml version="1.0" encoding=\"UTF-8\"?>
<hello><capabilities>
     <capability>u?rn:ietf:params:netconf:base:1.0</capability>
      <capability>urn:ietf:params:netconf:capability:writeable-running:1.0</capability>
       <capability>urn:ietf:params:netconf:ca?pability:roll?back-on-error:1.0</capability>
        <capability>urn:ietf:params:netconf:capability:startup:1.0</capability>
         <capability>urn:ietf:params:netconf:ca?pability:url:?1.0</capability>
        <capability>urn:cisco:params:netconf:capability:pi-data-model:1.0</capability>
       <capability>urn:cisco:params:netconf:capabili?ty:notificati?on:1.0</capability>
   </capabilities>
</hello>]]>]]>
```

Step 2 Next, send the get-config request:

```
<?xml version="1.0"?>
<rpc
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"xmlns:cpi="http://www.cisco.com/cpi_10/sche
ma" message-id="101">
   <get-config>
        <source>
             <running/>
         </source>
          <filter>
              <config-format-text-cmd>
               <text-filter-spec>
        interface Loopback113
                </text-filter-spec>
```

```
            </config-format-text-cmd>
          </filter>
      </get-config>
</rpc>]]>]]>
```

The following output is shown on the router:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101"xmlns=\"urn:ietf:params:netconf:base:1.0\">
   <data>
      <cli-config-data>
interface Loopback113
description test456
no ip address
load-interval 30
end
       </cli-config-data>
    </data>
</rpc-rep?ly>]]>]]>
```

Configuring NETCONF over BEEP: Example

The following example shows how to configure NETCONF over BEEP:

```
Router# configure terminal

Router(config)# crypto key generate rsa general-keys
Router(ca-trustpoint)# crypto pki trustpoint my_trustpoint
Router(ca-trustpoint)# enrollment url http://10.2.3.3:80
Router(ca-trustpoint)# subject-name CN=dns_name_of_host.com
Router(ca-trustpoint)# revocation-check none

Router(ca-trustpoint)# crypto pki authenticate my_trustpoint
Router(ca-trustpoint)# crypto pki enroll my_trustpoint
Router(ca-trustpoint)# line vty 0 15
Router(ca-trustpoint)# exit
Router(config)# netconf lock-time 60
Router(config)# netconf max-sessions 16

Router(config)# netconf beep initiator host1 23 user my_user password my_password encrypt
my_trustpoint reconnect-time 60

Router(config)# netconf beep listener 23 sasl user1 encrypt my_trustpoint
```

Configuring NETCONF Network Manager Application: Example

The following example shows how to configure the NETCONF Network Manager application to invoke NETCONF as an SSH subsystem:

```
Unix Side: ssh-2 -s companyname@10.1.1.1 netconf
```

As soon as the NETCONF session is established, indicate the server capabilities by sending an XML document containing a <hello>:

```
<?xml version="1.0" encoding="UTF-8"?>
    <hello>
      <capabilities>
        <capability>
            urn:ietf:params:xml:ns:netconf:base:1.0
```

```
          </capability>
          <capability>
            urn:ietf:params:ns:netconf:capability:startup:1.0
          </capability>
       </capabilities>
    <session-id>4<session-id>
</hello>]]>]]>
```

The client also responds by sending an XML document containing a <hello>:

```
<?xml version="1.0" encoding="UTF-8"?>
 <hello>
   <capabilities>
       <capability>
           urn:ietf:params:xml:ns:netconf:base:1.0
     </capability>
    </capabilities>
</hello>]]>]]>
```

Use the following XML string to enable the NETCONF network manager application to send and receive NETCONF notifications:

```
<?xml version="1.0" encoding="UTF-8" ?>
<rpc message-id="9.0"><notification-on/>
</rpc>]]>]]>
```

Use the following XML string to stop the NETCONF network manager application from sending or receiving NETCONF notifications:

```
<?xml version="1.0" encoding="UTF-8" ?>
<rpc message-id="9.13"><notification-off/>
</rpc>]]>]]>
```

Monitoring NETCONF Sessions: Example

The following is sample output from the **show netconf counters** command:

```
Router# show netconf counters

NETCONF Counters
Connection Attempts:0: rejected:0 no-hello:0 success:0
Transactions
        total:0, success:0, errors:0
detailed errors:
        in-use 0        invalid-value 0         too-big 0
        missing-attribute 0     bad-attribute 0         unknown-attribute 0
        missing-element 0       bad-element 0   unknown-element 0
        unknown-namespace 0     access-denied 0         lock-denied 0
        resource-denied 0       rollback-failed 0       data-exists 0
        data-missing 0  operation-not-supported 0       operation-failed 0
        partial-operation 0
```

The following is sample output from the **show netconf session** command:

```
Router# show netconf session

(Current | max) sessions:   3 | 4
Operations received: 100              Operation errors: 99
Connection Requests: 5                Authentication errors: 2   Connection Failures: 0
ACL dropped : 30
Notifications  Sent: 20
```

The output of the **show netconf schema** command describes the element structure for a NETCONF request and the resulting reply. This schema can be used to construct proper NETCONF requests and parse the resulting replies. The nodes in the schema are defined in RFC 4741. The following is sample output from the **show netconf schema** command:

```
Router# show netconf schema

New Name Space 'urn:ietf:params:xml:ns:netconf:base:1.0'
<VirtualRootTag> [0, 1] required
  <rpc-reply> [0, 1] required
    <ok> [0, 1] required
    <data> [0, 1] required
    <rpc-error> [0, 1] required
      <error-type> [0, 1] required
      <error-tag> [0, 1] required
      <error-severity> [0, 1] required
      <error-app-tag> [0, 1] required
      <error-path> [0, 1] required
      <error-message> [0, 1] required
      <error-info> [0, 1] required
        <bad-attribute> [0, 1] required
        <bad-element> [0, 1] required
        <ok-element> [0, 1] required
        <err-element> [0, 1] required
        <noop-element> [0, 1] required
        <bad-namespace> [0, 1] required
        <session-id> [0, 1] required
  <hello> [0, 1] required
    <capabilities> 1 required
      <capability> 1+ required
  <rpc> [0, 1] required
    <close-session> [0, 1] required
    <commit> [0, 1] required
      <confirmed> [0, 1] required
      <confirm-timeout> [0, 1] required
    <copy-config> [0, 1] required
      <source> 1 required
        <config> [0, 1] required
          <cli-config-data> [0, 1] required
            <cmd> 1+ required
          <cli-config-data-block> [0, 1] required
          <xml-config-data> [0, 1] required
            <Device-Configuration> [0, 1] required
              <> any subtree is allowed
        <candidate> [0, 1] required
        <running> [0, 1] required
        <startup> [0, 1] required
        <url> [0, 1] required
      <target> 1 required
        <candidate> [0, 1] required
        <running> [0, 1] required
        <startup> [0, 1] required
        <url> [0, 1] required
    <delete-config> [0, 1] required
      <target> 1 required
        <candidate> [0, 1] required
        <running> [0, 1] required
        <startup> [0, 1] required
        <url> [0, 1] required
    <discard-changes> [0, 1] required
    <edit-config> [0, 1] required
      <target> 1 required
        <candidate> [0, 1] required
```

```
    <running> [0, 1] required
    <startup> [0, 1] required
    <url> [0, 1] required
  <default-operation> [0, 1] required
  <test-option> [0, 1] required
  <error-option> [0, 1] required
  <config> 1 required
    <cli-config-data> [0, 1] required
      <cmd> 1+ required
    <cli-config-data-block> [0, 1] required
    <xml-config-data> [0, 1] required
      <Device-Configuration> [0, 1] required
        <> any subtree is allowed
<get> [0, 1] required
  <filter> [0, 1] required
    <config-format-text-cmd> [0, 1] required
      <text-filter-spec> [0, 1] required
    <config-format-text-block> [0, 1] required
      <text-filter-spec> [0, 1] required
    <config-format-xml> [0, 1] required
    <oper-data-format-text-block> [0, 1] required
      <show> 1+ required
    <oper-data-format-xml> [0, 1] required
      <show> 1+ required
<get-config> [0, 1] required
  <source> 1 required
    <config> [0, 1] required
      <cli-config-data> [0, 1] required
        <cmd> 1+ required
      <cli-config-data-block> [0, 1] required
      <xml-config-data> [0, 1] required
        <Device-Configuration> [0, 1] required
          <> any subtree is allowed
    <candidate> [0, 1] required
    <running> [0, 1] required
    <startup> [0, 1] required
    <url> [0, 1] required
  <filter> [0, 1] required
    <config-format-text-cmd> [0, 1] required
      <text-filter-spec> [0, 1] required
    <config-format-text-block> [0, 1] required
      <text-filter-spec> [0, 1] required
    <config-format-xml> [0, 1] required
<kill-session> [0, 1] required
  <session-id> [0, 1] required
<lock> [0, 1] required
  <target> 1 required
    <candidate> [0, 1] required
    <running> [0, 1] required
    <startup> [0, 1] required
    <url> [0, 1] required
<unlock> [0, 1] required
  <target> 1 required
    <candidate> [0, 1] required
    <running> [0, 1] required
    <startup> [0, 1] required
    <url> [0, 1] required
<validate> [0, 1] required
  <source> 1 required
    <config> [0, 1] required
      <cli-config-data> [0, 1] required
        <cmd> 1+ required
      <cli-config-data-block> [0, 1] required
      <xml-config-data> [0, 1] required
```

```
            <Device-Configuration> [0, 1] required
              <> any subtree is allowed
          <candidate> [0, 1] required
          <running> [0, 1] required
          <startup> [0, 1] required
          <url> [0, 1] required
      <notification-on> [0, 1] required
      <notification-off> [0, 1] required
```

Additional References

The following sections provide references related to the NETCONF feature.

Related Documents

Related Topic	Document Title
IP access lists	IP Access List Overview and Creating an IP Access List and Applying It to an Interface modules in the Cisco IOS Security Configuration Guide: Securing the Data Plane.
Secure Shell and Secure Shell Version 2	"Configuring Secure Shell" module in the Cisco IOS Security Configuration Guide: Securing User Services.
NETCONF commands: complete command syntax, command mode, command history, defaults, usage guidelines, and examples	*Cisco IOS Network Management Command Reference*
IP access lists commands: complete command syntax, command mode, command history, defaults, usage guidelines, and examples Security commands: complete command syntax, command mode, command history, defaults, usage guidelines, and examples	*Cisco IOS Security Command Reference*

Standards

Standard	Title
None	—

MIBs

MIB	MIBs Link
None	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
RFC 2222	*Simple Authentication and Security Layer (SASL)*
RFC 2246	*The TLS Protocol Version 1.0*
RFC 3080	*The Blocks Extensible Exchange Protocol Core*
RFC 4251	*The Secure Shell (SSH) Protocol Architecture*
RFC 4252	*The Secure Shell (SSH) Authentication Protocol*
RFC 4741	*NETCONF Configuration Protocol*
RFC 4742	*Using the NETCONF Configuration Protocol over Secure SHell (SSH)*
RFC 4744	*Using the NETCONF Protocol over the Blocks Extensible Exchange Protocol (BEEP)*

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/techsupport

Feature Information for NETCONF

Table 1 lists the features in this module and provides links to specific configuration information.

Not all commands may be available in your Cisco IOS software release. For release information about a specific command, see the command reference documentation.

Use Cisco Feature Navigator to find information about platform support and software image support. Cisco Feature Navigator enables you to determine which Cisco IOS and Catalyst OS software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Note Table 1 lists only the Cisco IOS software release that introduced support for a given feature in a given Cisco IOS software release train. Unless noted otherwise, subsequent releases of that Cisco IOS software release train also support that feature.

Table 1 Feature Information for NETCONF

Feature Name	Releases	Feature Information
NETCONF over SSHv2	12.2(33)SRA 12.4(9)T 12.2(33)SB 12.2(33)SXI	The NETCONF over SSHv2 feature enables you to perform network configurations via the Cisco command-line interface (CLI) over an encrypted transport. The NETCONF protocol defines a simple mechanism through which a network device can be managed, configuration data information can be retrieved, and new configuration data can be uploaded and manipulated. NETCONF uses an Extensible Markup Language (XML)-based data encoding for the configuration data and protocol messages. • In 12.4(9)T, this feature was introduced. The following sections provide information about this feature: • NETCONF over SSHv2, page 2 • NETCONF Notifications, page 5 • Enabling SSH Version 2 Using a Hostname and Domain Name, page 5 • Enabling SSH Version 2 Using RSA Key Pairs, page 6 • Enabling NETCONF over SSHv2, page 9 • Configuring the NETCONF Network Manager Application, page 15 • Configuring NETCONF over SSHv2: Example, page 23 The following commands were introduced or modified by this feature: **clear netconf**, **debug netconf**, **netconf lock-time**, **netconf max-sessions**, **netconf ssh**, **show netconf**.

Table 1 Feature Information for NETCONF (continued)

Feature Name	Releases	Feature Information
NETCONF Access for Configuration over BEEP	12.4(9)T 12.2(33)SRB 12.2(33)SB 12.2(33)SXI	The NETCONF over BEEP feature allows you to enable either the NETCONF server or the NETCONF client to initiate a connection, thus supporting large networks of intermittently connected devices and those devices that must reverse the management connection where there are firewalls and network address translators (NATs). • In 12.4(9)T, this feature was introduced. The following sections provide information about this feature: • NETCONF over BEEP, page 3 • NETCONF Notifications, page 5 • Configuring an SASL Profile, page 11 • Enabling NETCONF over BEEP, page 12 • Configuring the NETCONF Network Manager Application, page 15 • Configuring NETCONF over SSHv2: Example, page 23 The following commands were introduced or modified by this feature: **netconf beep initiator**, **netconf beep listener**.

Glossary

BEEP—Blocks Extensible Exchange Protocol. A generic application protocol framework for connection-oriented, asynchronous interactions.

NETCONF—Network Configuration Protocol. A protocol that defines a simple mechanism through which a network device can be managed, configuration data information can be retrieved, and new configuration data can be uploaded and manipulated.

SASL—Simple Authentication and Security Layer. An Internet standard method for adding authentication support to connection-based protocols. SASL can be used between a security appliance and an Lightweight Directory Access Protocol (LDAP) server to secure user authentication.

SSHv2—Secure Shell Version 2. SSH runs on top of a reliable transport layer and provides strong authentication and encryption capabilities. SSHv2 provides a means to securely access and securely execute commands on another computer over a network.

TLS—Transport Layer Security. An application-level protocol that provides for secure communication between a client and server by allowing mutual authentication, the use of hash for integrity, and encryption for privacy. TLS relies upon certificates, public keys, and private keys.

XML—Extensible Markup Language. A standard maintained by the World Wide Web Consortium (W3C) that defines a syntax that lets you create markup languages to specify information structures. Information structures define the type of information (for example, subscriber name or address), not how the information looks (bold, italic, and so on). External processes can manipulate these information structures and publish them in a variety of formats. XML allows you to define your own customized markup language.

Distributed Director

Configuring a DRP Server Agent

First Published: July 6, 1999
Last Updated: October 11, 2006

This module describes how to configure a Director Response Protocol (DRP) Agent and how to configure support for the boomerang metric on a DRP Server Agent.

Finding Feature Information in This Module

Your Cisco IOS software release may not support all the features documented in this module. To reach links to specific feature documentation in this module and to see a list of the releases in which each feature is supported, use the "Feature Information for Configuring a DRP Server Agent" section on page 17.

Finding Support Information for Platforms and Cisco IOS and Catalyst OS Software Images

Use Cisco Feature Navigator to find information about platform support and Cisco IOS and Catalyst OS software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Contents

Restrictions for Configuring a DRP Server Agent

- When DistributedDirector is upgraded to include the boomerang function, DRP Server Agents must be made aware that boomerang is present.

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

Information About Configuring a DRP Server Agent

To configure a DRP Server Agent or to configure one with Boomerang support, you should understand the following concepts:

Director Response Protocol

DRP is a simple User Datagram Protocol (UDP)-based application developed by Cisco Systems. DRP enables the Cisco DistributedDirector product to query routers (DRP Server Agents) in the field for Border Gateway Protocol (BGP) and Interior Gateway Protocol (IGP) routing table metrics between distributed servers and clients. DistributedDirector, separate standalone software, uses DRP to transparently redirect end-user service requests to the topologically closest responsive server. DRP enables DistributedDirector to provide dynamic, scalable, and "network intelligent" Internet traffic load distribution among multiple geographically dispersed servers.

DRP Server Agent

A DRP Server Agent is a border router or peer to a border router that supports the geographically distributed servers for which DistributedDirector service is desired. DistributedDirector makes decisions based on BGP and IGP information, meaning that all DRP Server Agents must have full access to BGP and IGP routing tables.

Racing Message

A racing message occurs when DistributedDirector receives a Domain Name System (DNS) query from a DNS client for a hostname that has the boomerang metric configured. DistributedDirector issues a DNS racing message to the different DRP Server Agents. In the message, DistributedDirector instructs each DRP Server Agent to respond directly to the client with the answer. The instruction, which is determined by the DistributedDirector configuration, also specifies whether the response should be sent at a specific time or after a certain delay.

Boomerang Metric

Boomerang is a DRP metric for DistributedDirector. When the boomerang metric is active, DistributedDirector instructs the DRP to send DNS responses directly to the querying client. The DNS response contains addresses of sites associated with a specific DRP Server Agent. All involved DRP Server Agents send their DNS responses at the same time. The packet of the DRP that is nearest to the client in terms of delay arrives first. The client may take the first answer and ignore subsequent ones, which is a standard behavior of all local DNS server implementations. Full boomerang support can be configured on a DRP Server Agent. The boomerang client is the DRP Server Agent.

The boomerang metric enables a boomerang client on the DRP Server Agent to communicate with boomerang-supported servers. The metric promotes interoperability among different content routers within Cisco. The boomerang client on the DRP Server Agent can communicate with any boomerang server, not only servers implemented on DistributedDirector.

When a boomerang DRP Server Agent receives a DNS racing message from boomerang servers, the DRP extracts the domain name specified in the DNS message. A DRP Server Agent with Boomerang support can be configured on this specified domain.

Benefits of a DRP Server Agent

DRP Server Agents provide the following benefits:

- Use of DistributedDirector service is facilitated.
- A means to select a site with the fastest response time is provided with Boomerang support.
- Congestion and link failures are dynamically recognized and avoided with Boomerang support.

How to Configure a DRP Server Agent

Perform these tasks to configure and maintain a DRP Server Agent.

Enabling the DRP Server Agent

Perform this task to enable a DRP Server Agent (it is disabled by default).

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip drp server**
4. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**ip drp server** **Example:** `Router(config)# ip drp server`	Enables a DRP Server Agent.
Step 4	**exit** **Example:** `Router(config)# exit`	Returns the CLI to privileged EXEC mode.

Limiting the Source of DRP Queries

As a security measure, you can limit the source of valid DRP queries. When a standard IP access list is applied to an interface, the DRP Server Agent will respond only to DRP queries originating from an IP address in that list. If no access list is configured, the DRP Server Agent answers all queries.

When both an access group and a key chain (described in the next section) have been configured, both security mechanisms must allow access before a request is processed.

Perform this task to limit the source of valid DRP queries.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip drp access-group** *access-list-number*
4. **exit**

DETAILED STEPS

	Command	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**ip drp access-group** *access-list-number* **Example:** `Router(config)# ip drp access-group 1`	Controls the sources of valid DRP queries by applying a standard IP access list. In this instance, the access list is number 1.
Step 4	**exit** **Example:** `Router(config)# exit`	Returns the command-line interface (CLI) to privileged EXEC mode.

Configuring Authentication of DRP Queries and Responses

Perform this task to define a key chain, identify the keys that belong to the key chain, and optionally specify the time period during which each key is valid.

Authentication Keys and Key Chains

Another available security measure is to configure the DRP Server Agent to authenticate DRP queries and responses.

When configuring key chains and keys, use the following guidelines:

- The name of the key chain configured for DRP authentication must match the name of the key chain configured.
- The key configured in the primary agent in the remote router must match the key configured in the DRP Server Agent for responses to be processed.
- You can configure multiple keys with lifetimes and the software will rotate through them.
- If authentication is enabled and multiple keys on the key chain are active based on the **send-lifetime** values, the software uses only the first key it encounters for authentication.
- Use the **show key chain** command to display key chain information.

Restrictions

- To configure lifetimes for DRP authentication, you must configure time services for your router.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip drp authentication key-chain** *name-of-chain*
4. **key chain** *name-of-chain*
5. **key** *key-id*
6. **key-string** *text*
7. **accept-lifetime** *start-time* {**infinite** | *end-time* | **duration** *seconds*}
8. **send-lifetime** *start-time* {**infinite** | *end-time* | **duration** *seconds*}
9. **exit**

DETAILED STEPS

	Command	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**ip drp authentication key-chain** *name-of-chain* **Example:** `Router(config)# ip drp authentication key-chain mktg`	Identifies the key chain to be used for authenticating all DRP requests and responses.
Step 4	**key chain** *name-of-chain* **Example:** `Router(config)# key chain mktg`	Identifies the key chain named in Step 3 and places the CLI in key chain configuration mode.
Step 5	**key** *key-id* **Example:** `Router(config-keychain)# key 1`	Identifies the key number 1.
Step 6	**key-string** *text* **Example:** `Router(config-keychain-key)# key-string internal`	Identifies the key string as internal.

	Command	Purpose
Step 7	**accept-lifetime** *start-time* {**infinite** | *end-time* | **duration** *seconds*} **Example:** `Router(config-keychain-key)# accept-lifetime 15:00:00 Oct 12 2006 600`	(Optional) Specifies the time period during which the key can be received. In this instance, the time period is 600 seconds.
Step 8	**send-lifetime** *start-time* {**infinite** | *end-time* | **duration** *seconds*} **Example:** `Router(config-keychain-key)# send-lifetime 14:30:00 Oct 12 2006 300`	(Optional) Specifies the time period during which the key can be sent. In this instance, the time period is 300 seconds.
Step 9	**exit** **Example:** `Router(config)# exit`	Returns the CLI to privileged EXEC mode.

Monitoring and Maintaining a DRP Server Agent

Perform this task to monitor and maintain a DRP Server Agent.

SUMMARY STEPS

1. **enable**
2. **clear ip drp**
3. **show ip drp**
4. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**clear ip drp** **Example:** `Router# clear ip drp`	Clears statistics being collected for DRP requests and responses.
Step 3	**show ip drp** **Example:** `Router# show ip drp`	Displays information about the DRP Server Agent.
Step 4	**exit** **Example:** `Router# exit`	Returns the CLI to user EXEC mode.

Adding a New Domain or Configuring an Existing Domain

Perform this task to add a new domain to the DistributedDirector client or to configure an existing domain. This task is performed on the DRP Server Agent.

1. **enable**
2. **configure terminal**
3. **ip drp domain** *domain-name*
4. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**ip drp domain** *domain-name* **Example:** `Router(config)# ip drp domain www.boom1.com`	Specifies a domain to be added or configured and puts the CLI in boomerang configuration mode. The domain in this example is named www.boom1.com.
Step 4	**exit** **Example:** `Router(config-boomerang)# exit`	Returns the CLI to privileged EXEC mode.

Configuring a Domain Name Alias

Perform this task to configure an alias name for a specified domain.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip drp domain** *domain-name*
4. **alias** *alias-name*
5. **exit**

DETAILED STEPS

	Command	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**ip drp domain** *domain-name* **Example:** `Router(config)# ip drp domain www.boom1.com`	Specifies a domain to be added or configured and puts the CLI in boomerang configuration mode.
Step 4	**alias** *alias-name* **Example:** `Router(config-boomerang)# alias www.boom2.com`	Configures an alias name for a specified domain. The alias name in this example is www.boom2.com.
Step 5	**exit** **Example:** `Router(config-boomerang)# exit`	Returns the CLI to privileged EXEC mode.

Configuring the Server Address of a Domain

Perform this task to configure the server address for a specified boomerang domain.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip drp domain** *domain-name*
4. **server** *server-ip-address*
5. **exit**

DETAILED STEPS

	Command	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**ip drp domain** *domain-name* **Example:** `Router(config)# ip drp domain www.boom1.com`	Specifies a domain to be added or configured and puts the CLI in boomerang configuration mode.
Step 4	**server** *server-ip-address* **Example:** `Router(config-boomerang)# server 172.16.101.101`	Configures an IP address for a specified domain.
Step 5	**exit** **Example:** `Router(config-boomerang)# exit`	Returns the CLI to privileged EXEC mode.

Configuring an IP Time-to-Live Value

Perform this task to configure the IP time-to-live (TTL) value for packets sent from a boomerang client to a DNS client, in number of hops.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip drp domain** *domain-name*
4. **ttl ip** *hops*
5. **exit**

DETAILED STEPS

	Command	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**ip drp domain** *domain-name* **Example:** `Router(config)# ip drp domain www.boom1.com`	Specifies a domain to be added or configured and puts the CLI in boomerang configuration mode.
Step 4	**ttl ip** *hops* **Example:** `Router(config-boomerang)# ttl ip 2`	Configures the maximum number of hops between the boomerang client and the DNS client, after which the boomerang response packet fails. The number of hops in this example is 2.
Step 5	**exit** **Example:** `Router(config-boomerang)# exit`	Returns the CLI to privileged EXEC mode.

Configuring a DNS TTL Value

Perform this task to configure the number of seconds that a DNS client will cache an answer received from a boomerang client.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip drp domain** *domain-name*
4. **ttl dns** *seconds*
5. **exit**

DETAILED STEPS

	Command	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**ip drp domain** *domain-name* **Example:** `Router(config)# ip drp domain www.boom1.com`	Specifies a domain to be added or configured and puts the CLI in boomerang configuration mode.
Step 4	**ttl dns** *seconds* **Example:** `Router(config-boomerang)# ttl dns 10`	Configures the number of seconds for which the DNS client can cache a boomerang reply from a boomerang client. The number of seconds in this example is 10.
Step 5	**exit** **Example:** `Router(config-boomerang)# exit`	Returns the CLI to privileged EXEC mode.

Verifying Boomerang Information on a DRP Server Agent

Perform this task to verify that boomerang support was successfully configured on a DRP Server Agent.

Step 1 Enter the **show ip drp boomerang** command to display boomerang information on a DRP Server Agent. The output of this command verifies information such as whether the server is up or down, the number of DNS requests received, and the number of requests dropped because the server is down.

```
Router# show ip drp boomerang

DNS packets with unknown domain 0

Domain www.boom1.com
Content server            172.16.101.101 up
Origin server                    0.0.0.0
DNS A record requests                  0
Dropped (server down)                  0
Dropped (no origin server)             0
Security failures                      0

Alias www.boom2.com
DNS A record requests                  0
```

Step 2 Enter the **show ip drp** command to display additional information such as the number of requests received from DistributedDirector, the total number of boomerang requests, and the number of boomerang responses made by a DRP Server Agent.

```
Router# show ip drp

Director Responder Protocol Agent is enabled
3 director requests:
0 successful route table lookups
0 successful measured lookups
0 no route in table
0 nortt
0 DRP packet failures returned
3 successful echos
6 Boomerang requests
0 Boomerang-raced DNS responses
Authentication is enabled, using "DD" key-chain
rttprobe source port is     :53
rttprobe destination port is:53
```

Troubleshooting Tips

If the **ip drp domain** *domain-name* command is configured on the DRP Server Agent, but a corresponding server address is not specified for this domain name, the content-server field defaults to 0.0.0.0. The **show ip drp boomerang** command displays this information. In this case, the DRP Server Agent would be removed from the boomerang configuration. To include it again, enter boomerang configuration mode and specify a server address.

```
Router> enable
Router# configure terminal
Router(config)# ip drp domain www.boom1.com
Router(config-boomerang)# server 172.16.101.101
```

Configuration Examples for Configuring a DRP Server Agent

Enabling a DRP Server Agent and Limiting Query Sources: Example

The following example shows how to enable the DRP Server Agent, limit the sources of DRP queries to those listed in access list 1, and configure authentication for DRP queries and responses. The access list permits queries from only the host at address 192.168.5.5.

```
ip drp server
access-list 1 permit 192.168.5.5
ip drp access-group 1
ip drp authentication key-chain mktg
key chain mktg
 key 1
  key-string internal
```

Adding a New Domain or Configuring an Existing Domain: Example

In the following example, a domain named www.boom1.com is added on a boomerang client:

```
ip drp domain www.boom1.com

show running-configuration
.
.
ip drp domain www.boom1.com
```

Configuring a Domain Name Alias: Example

In the following example, the domain name alias configured for www.boom1.com is www.boom2.com:

```
ip drp domain www.boom1.com
alias www.boom2.com

show running-configuration
.
.
ip drp domain www.boom1.com
alias www.boom2.com
```

Configuring the Server Address of a Domain: Example

In the following example, the server address is configured for www.boom1.com. The server address for www.boom1.com is 172.16.101.101.

```
ip drp domain www.boom1.com
server 172.16.101.101

show running-configuration
.
.
ip drp domain www.boom1.com
content-server 172.16.101.101
```

Configuring an IP TTL Value: Example

In the following example, the number of hops that occur between the boomerang client and the DNS client before the boomerang response packet fails is 2:

```
ip drp domain www.boom1.com
ttl ip 2

show running-configuration
.
.
ip drp domain www.boom1.com
ip-ttl 2
```

Configuring a DNS TTL Value: Example

In the following example, the number of seconds for which the DNS client can cache a boomerang reply from a boomerang client is 10:

```
ip drp domain www.boom1.com
ttl dns 10

show running-configuration
.
.
ip drp domain www.boom1.com
dns-ttl 10
```

Additional References

The following sections provide references related to the Configuring a DRP Server Agent module.

Related Documents

Related Topic	Document Title
DRP Server Agent related commands	*Cisco IOS Network Management Command Reference*, Release 12.4
Configuring DistributedDirector	The "DistributedDirector Configuration" chapter of the *Cisco IOS Network Management Configuration Guide,* Release 12.4 *Cisco DistributedDirector 4700-M Installation and Configuration Guide*
DistributedDirector Boomerang Support	The "DistributedDirector Boomerang Support" chapter of the *Cisco IOS Network Management Configuration Guide,* Release 12.4
Network Time Protocol and setting time services	The "Performing Basic System Management" chapter of the *Cisco IOS Network Management Configuration Guide,* Release 12.4

Technical Assistance

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

Feature Information for Configuring a DRP Server Agent

Table 1 lists the features in this module and provides links to specific configuration information. Only features that were introduced or modified in Cisco IOS Release 11.2(4)F or Cisco IOS Release 12.2(8)T or a later release appear in the table.

Not all commands may be available in your Cisco IOS software release. For release information about a specific command, see the command reference documentation.

Use Cisco Feature Navigator to find information about platform support and software image support. Cisco Feature Navigator enables you to determine which Cisco IOS and Catalyst OS software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Note Table 1 lists only the Cisco IOS software release that introduced support for a given feature in a given Cisco IOS software release train. Unless noted otherwise, subsequent releases of that Cisco IOS software release train also support that feature.

Table 1 Feature Information for Configuring a DRP Server Agent

Feature Name	Releases	Feature Information
DRP Agent—Boomerang Support	12.2(8)T	Boomerang is a DRP metric for DistributedDirector. When the boomerang metric is active, DistributedDirector instructs the DRP to send DNS responses directly to the querying client. The DNS response contains the addresses of sites associated with a specific DRP Server Agent. The following sections provide information about this feature: • Director Response Protocol, page 2 • DRP Server Agent, page 2 • Racing Message, page 2 • Boomerang Metric, page 2 • Benefits of a DRP Server Agent, page 3 • How to Configure a DRP Server Agent, page 3 • Adding a New Domain or Configuring an Existing Domain, page 8 • Configuring a Domain Name Alias, page 9 • Configuring the Server Address of a Domain, page 10 • Configuring an IP Time-to-Live Value, page 11 • Configuring a DNS TTL Value, page 12 • Verifying Boomerang Information on a DRP Server Agent, page 13
DRP Server Agent	11.2(4)F	A DRP Server Agent is a border router or peer to a border router that supports the geographically distributed servers for which DistributedDirector service is desired. The following sections provide information about this feature: • Director Response Protocol, page 2 • DRP Server Agent, page 2 • Enabling the DRP Server Agent, page 3 • Limiting the Source of DRP Queries, page 4 • Configuring Authentication of DRP Queries and Responses, page 5 • Monitoring and Maintaining a DRP Server Agent, page 7

DistributedDirector MIB Support

First Published: February 25, 2002
Last Updated: February 27, 2009

Note Effective with Cisco IOS Release 12.4(24)T, this feature is not available in Cisco IOS software.

Feature History

Release	Modification
12.2(8)T	This feature was introduced.
12.4(24)T	This feature was removed.

This document describes DistributedDirector MIB support and the enhancements and modifications made to the Cisco IOS Simple Network Management Protocol (SNMP) infrastructure in order to support DistributedDirector in Cisco IOS Release 12.2(8)T. It includes the following sections:

Feature Overview

Network management takes place between two major types of systems: those in control, called managing systems, and those observed and controlled, called managed systems. The most common type of managing system is called a *network management system* (NMS). Managed systems can include hosts, servers, or network components such as routers or intelligent repeaters.

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

To promote interoperability, cooperating systems must adhere to a common framework and a common language, called a *protocol*. In the Internet network management framework, that protocol is the SNMP.

In a managed device, specialized low-impact software modules, called *agents*, access information about the device and make it available to the NMS. Managed devices maintain values for a number of variables and report those, as required, to the NMS. For example, an agent might report such data as the number of bytes and packets passing in and out of the device, or the number of broadcast messages sent and received. In the Internet network management framework, each variable is referred to as a *managed object*, which is anything that an agent can access and report back to the NMS.

All managed objects are contained in the Management Information Base (MIB), which is a database of the managed objects. The managed objects, or variables, can be set or read to provide information on network devices and interfaces. An NMS can control a managed device by sending a message to an agent of that managed device requiring the device to change the value of one or more of its variables.

The Cisco DistributedDirector MIB provides MIB support for DistributedDirector. This MIB contains DistributedDirector statistics, configurations, and status.

The DistributedDirector MIB contains five groups of object type definitions:

- ciscoDistDirGeneralGroup—A group of objects related to DistributedDirector general configurations, statistics, and status.
- ciscoDistDirHostGroup—A group of objects related to DistributedDirector host-specific configurations, statistics, and status.
- ciscoDistDirServerGroup—A group of objects related to DistributedDirector server-specific configurations, statistics, and status.
- ciscoDistDirMappingGroup—A group of objects related to associations between DistributedDirector host names and real servers.
- ciscoDistDirNotificatonGroup—A group of objects related to DistributedDirector significant events.

The DistributedDirector MIB defines the following tables:

- cddGeneralMetricProfTable—DistributedDirector metric profiles. A profile contains priority and weight values of DistributedDirector metrics, which can be applied to specific hosts or to the DistributedDirector default configuration.
- cddHostTable—DistributedDirector virtual host name or subdomain-specific configurations, statistics, and status entries.
- cddHostConnectCfgTable—DistributedDirector per-host server connect test configuration information entries.
- cddHostTolCfgTable—DistributedDirector per-host priority-level metric tolerance configuration information entries.
- cddServerTable—DistributedDirector server-specific information entries. This information includes the configuration parameters and statistics for each server.
- cddServerPortTable—DistributedDirector server port-specific information entries. This information includes the configuration parameters, statistics, and availability status for each service port on servers.
- cddServerPortMetricTable—DistributedDirector per-service per-metric weight entries.
- cddHostServerMappingTable—DistributedDirector associations of virtual host name to real server.

The DistributedDirector MIB defines the following notifications:

- ciscoDistDirEventServerUp—This trap is generated whenever a distributed server changes to the “up” state.

- ciscoDistDirEventServerDown—This trap is generated whenever a distributed server changes to the "down" state.
- ciscoDistDirEventHitRateHigh—This trap is generated whenever the incoming Domain Name system (DNS) HTTP query rate reaches a certain threshold. Use the Event MIB described in RFC 2981 to control the trigger of this notification.

The ciscoDistDirEventServerUp and ciscoDistDirEventServerDown notifications can be enabled or disabled using the Cisco IOS **snmp-server enable traps director** and **snmp-server host** commands.

The **snmp-server host** command is used in conjunction with the **snmp-server enable traps director** command. Use the **snmp-server enable traps director** command to specify which DistributedDirector SNMP notifications are sent globally. For a host to receive most notifications, at least one **snmp-server enable traps director** command and the **snmp-server host** command for that host must be enabled.

Benefits

The DistributedDirector MIB provides network management functionality to DistributedDirector.

Restrictions

The DistributedDirector MIB implementation for Cisco IOS Release 12.2(8)T supports read-only capability to the objects defined in the MIB.

Related Features and Technologies

- Event MIB
- SNMP
- Network management

Related Documents

- The "Configuring SNMP Support" chapter of *Cisco IOS Configuration Fundamentals Configuration Guide,* Release 12.2
- The "SNMP Commands" chapter of *Cisco IOS Configuration Fundamentals Command Reference,* Release 12.2
- RFC 1157, "Simple Network Management Protocol"
- Event MIB: RFC 2981, *Event MIB*

Supported Platforms

- Cisco 2600 series
- Cisco 3620 series
- Cisco 3640 series
- Cisco 3660 series

- Cisco 7200 series

Determining Platform Support Through Cisco Feature Navigator

Cisco IOS software is packaged in feature sets that support specific platforms. To get updated information regarding platform support for this feature, access Cisco Feature Navigator. Cisco Feature Navigator dynamically updates the list of supported platforms as new platform support is added for the feature.

Cisco Feature Navigator is a web-based tool that enables you to quickly determine which Cisco IOS software images support a specific set of features and which features are supported in a specific Cisco IOS image. You can search by feature or release. Under the release section, you can compare releases side by side to display both the features unique to each software release and the features in common.

Cisco Feature Navigator is updated regularly when major Cisco IOS software releases and technology releases occur. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Supported Standards, MIBs, and RFCs

Standards

No new or modified standards are supported by this feature.

MIBs

- Cisco DistributedDirector MIB (CISCO-DIST-DIRECTOR-MIB.my)
- Event MIB (EVENT-MIB.my)

To obtain lists of supported MIBs by platform and Cisco IOS release, and to download MIB modules, go to the Cisco MIB website on Cisco.com at the following URL:

http://www.cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

- Event MIB: RFC 2981, *Event MIB*

Prerequisites

DistributedDirector must be running on the router.

Configuration Tasks

See the following sections for configuration tasks for the DistributedDirector MIB support feature. Each task in the list is identified as either required or optional.

- Enabling DistributedDirector SNMP Notifications (required)
- Specifying the Recipient of an SNMP Notification (required)

Enabling DistributedDirector SNMP Notifications

To enable DistributedDirector SNMP notifications, use the following command in global configuration mode:

Command	Purpose
`Router(config)# snmp-server enable traps director`	Enables DistributedDirector SNMP notifications.

To disable DistributedDirector SNMP notifications, use the following command in global configuration mode:

Command	Purpose
`Router(config)# no snmp-server enable traps director`	Disables DistributedDirector SNMP notifications.

Specifying the Recipient of an SNMP Notification

To specify the recipient of a DistributedDirector SNMP notification, use the following command in global configuration mode:

Command	Purpose
`Router(config)# snmp-server host 10.0.0.1 public director`	Specifies the recipient of a DistributedDirector SNMP notification, where the host 10.0.0.1 is using the community string defined as “public.”

To remove the specified recipient, use the following command in global configuration mode:

Command	Purpose
`Router(config)# no snmp-server host` *host-address* `director`	Removes the recipient of a DistributedDirector SNMP notification.

Verifying DistributedDirector Notification Information

Enter the **show running-config** command to verify that DistributedDirector SNMP notification information is configured. Both server up and server down information is included, unless you specify one or the other.

```
Router# show running-config

ip host myhost 172.2.2.10 172.2.2.20 172.2.2.30
.
.
.
snmp-server enable traps director server-up server-down
```

Configuration Examples

This section provides the following configuration examples:

- Enabling DistributedDirector SNMP Notifications Example
- Specifying the Recipient of an SNMP Notification Example

Enabling DistributedDirector SNMP Notifications Example

In the following example, both ciscoDistDirEventServerUp and ciscoDistDirEventServerDown notifications are enabled:

```
Router(config)#  snmp-server enable traps director

Router# show running-config

ip host myhost 172.2.2.10 172.2.2.20 172.2.2.30
.
.
.
snmp-server enable traps director server-up server-down
```

Specifying the Recipient of an SNMP Notification Example

In the following example, the ciscoDistDirEventServerUp and ciscoDistDirEventServerDown notifications are to be sent to the host 10.0.0.1 using the community string defined as “public”:

```
Router(config)# snmp-server host 10.0.0.1 public director

Router# show snmp

Chassis:8768490
0 SNMP packets input
    0 Bad SNMP version errors
    0 Unknown community name
    0 Illegal operation for community name supplied
    0 Encoding errors
    0 Number of requested variables
    0 Number of altered variables
    0 Get-request PDUs
    0 Get-next PDUs
    0 Set-request PDUs
0 SNMP packets output
    0 Too big errors (Maximum packet size 1500)
    0 No such name errors
    0 Bad values errors
    0 General errors
    0 Response PDUs
    0 Trap PDUs

SNMP logging:enabled
    Logging to 10.0.0.1.162, 0/10, 0 sent, 0 dropped.
```

Command Reference

The following commands are introduced or modified in the feature or features documented in this module. For information about these commands, see the *Cisco IOS Network Management Command Reference* at http://www.cisco.com/en/US/docs/ios/netmgmt/command/reference/nm_book.html. For information about all Cisco IOS commands, go to the Command Lookup Tool at http://tools.cisco.com/Support/CLILookup or to the *Cisco IOS Master Commands List*.

- **snmp-server enable traps director**
- **snmp-server host**

Embedded Event Manager (EEM)

Embedded Event Manager Overview

First Published: October 31, 2005
Last Update: Novemeber 20, 2009

Embedded Event Manager (EEM) is a distributed and customized approach to event detection and recovery offered directly in a Cisco IOS device. EEM offers the ability to monitor events and take informational, corrective, or any desired EEM action when the monitored events occur or when a threshold is reached. An EEM policy is an entity that defines an event and the actions to be taken when that event occurs.

This module contains a technical overview of EEM. EEM can be used alone, or with other network management technologies to help monitor and maintain your network. Before you begin to implement EEM, it is important that you understand the information presented in this module.

Finding Feature Information in This Module

Your Cisco IOS software release may not support all of the features documented in this module. To reach links to specific feature documentation in this module and to see a list of the releases in which each feature is supported, use the "Feature Information for Embedded Event Manager Overview" section on page 20.

Finding Support Information for Platforms and Cisco IOS and Catalyst OS Software Images

Use Cisco Feature Navigator to find information about platform support and Cisco IOS and Catalyst OS software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Contents

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

Information About Embedded Event Manager

To use EEM in your network, you should understand the following concepts:

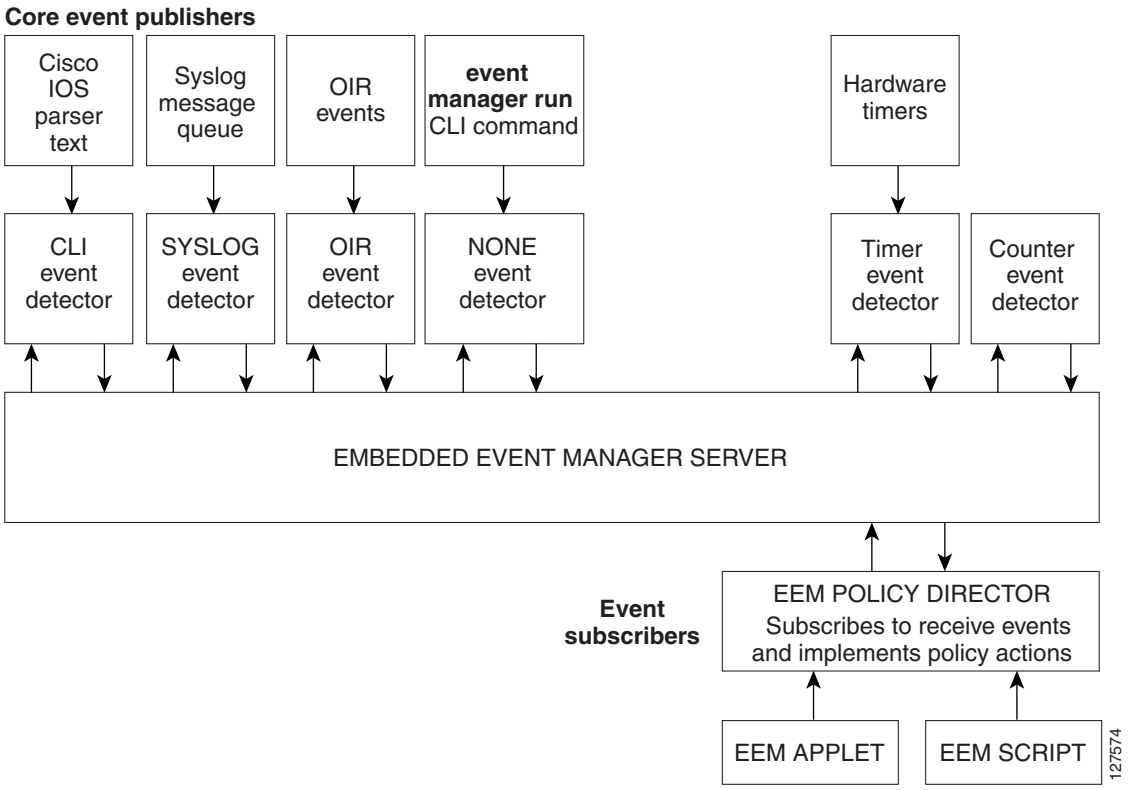
Embedded Event Manager

Event tracking and management has traditionally been performed by devices external to the networking device. Embedded Event Manager (EEM) has been designed to offer event management capability directly in Cisco IOS devices. The on-device, proactive event management capabilities of EEM are useful because not all event management can be done off router because some problems compromise communication between the router and the external network management device. Capturing the state of the router during such situations can be invaluable in taking immediate recovery actions and gathering information to perform root-cause analysis. Network availability is also improved if automatic recovery actions are performed without the need to fully reboot the routing device.

EEM is a flexible, policy-driven framework that supports in-box monitoring of different components of the system with the help of software agents known as event detectors. Figure 1 shows the relationship between the EEM server, core event publishers (event detectors), and the event subscribers (policies). Basically, event publishers screen events and publish them when there is a match on an event specification that is provided by the event subscriber. Event detectors notify the EEM server when an event of interest occurs. The EEM policies that are configured using the Cisco IOS command-line interface (CLI) then implement recovery on the basis of the current state of the system and the actions specified in the policy for the given event.

EEM offers the ability to monitor events and take informational or corrective action when the monitored events occur or when a threshold is reached. An EEM policy is an entity that defines an event and the actions to be taken when that event occurs. There are two types of EEM policies: an applet or a script. An applet is a simple form of policy that is defined within the CLI configuration. A script is a form of policy that is written in Tool Command Language (Tcl).

Figure 1 Embedded Event Manager Core Event Detectors



Embedded Event Manager 1.0

EEM 1.0 is supported in Cisco IOS Releases 12.0(26)S and 12.3(4)T and later releases and introduced Embedded Event Manager. EEM 1.0 introduced the following event detectors:

- SNMP—The Simple Network Management Protocol (SNMP) event detector allows a standard SNMP MIB object to be monitored and an event to be generated when the object matches specified values or crosses specified thresholds.
- Syslog—The syslog event detector allows for screening syslog messages for a regular expression pattern match.

EEM 1.0 introduced the following actions:

- Generating prioritized syslog messages.
- Generating a Cisco Networking Services (CNS) event for upstream processing by CNS devices.
- Reloading the Cisco IOS software.
- Switching to a secondary processor in a fully redundant hardware configuration.

Embedded Event Manager 2.0

EEM 2.0 is supported in Cisco IOS Release 12.2(25)S and later releases and introduced some new features. EEM 2.0 introduced the following event detectors:

- Application-Specific—The application-specific event detector allows any Embedded Event Manager policy to publish an event.
- Counter—The counter event detector publishes an event when a named counter crosses a specified threshold.
- Interface Counter—The interface counter event detector publishes an event when a generic Cisco IOS interface counter for a specified interface crosses a defined threshold.
- Timer—The timer event detector publishes events for the following four different types of timers: absolute-time-of-day, countdown, watchdog, and CRON.
- Watchdog System Monitor (IOSWDSysMon)—The Cisco IOS watchdog system monitor event detector publishes an event when CPU or memory utilization for a Cisco IOS process crosses a threshold.

EEM 2.0 introduced the following actions:

- Setting or modifying a named counter.
- Publishing an application-specific event
- Generating an SNMP trap.

The ability to run a Cisco defined sample policy written using Tool Command Language (Tcl) was introduced. A sample policy was provided that could be stored in the system policy directory.

Embedded Event Manager 2.1

EEM 2.1 is supported in Cisco IOS Release 12.3(14)T, 12.2(18)SXF5, 12.2(28)SB, 12.2(33)SRA, and later releases, and introduced some new features. EEM 2.1 introduced the following new event detectors:

- CLI—The CLI event detector screens command-line interface (CLI) commands for a regular expression match.
- None—The none event detector publishes an event when the Cisco IOS **event manager run** command executes an EEM policy.
- OIR—The online insertion and removal (OIR) event detector publishes an event when a particular hardware insertion or removal event occurs.

EEM 2.1 introduced the following actions:

- Executing a Cisco IOS CLI command.
- Requesting system information when an event occurs.
- Sending a short e-mail.
- Manually running an EEM policy.

EEM 2.1 also permits multiple concurrent policies to be run using the new **event manager scheduler script** command. Support for SNMP event detector rate-based events is provided as is the ability to create policies using Tool Command Language (Tcl).

Embedded Event Manager 2.1 (Software Modularity)

EEM 2.1 (Software Modularity) is supported in Cisco IOS Release 12.2(18)SXF4 and later releases on Cisco IOS Software Modularity images. EEM 2.1 (Software Modularity) introduced the following event detectors:

- GOLD—The Generic Online Diagnostic (GOLD) event detector publishes an event when a GOLD failure event is detected on a specified card and subcard.
- System Manager—The system manager event detector generates events for Cisco IOS Software Modularity process start, normal or abnormal stop, and restart events. The events generated by the system manager allows policies to change the default behavior of the process restart.
- Watchdog System Monitor (WDSysMon)—The Cisco IOS Software Modularity watchdog system monitor event detector detects infinite loops, deadlocks, and memory leaks in Cisco IOS Software Modularity processes.

EEM 2.1 for Software Modularity introduced the ability to display EEM reliability metric data for processes.

Note EEM 2.1 for Software Modularity images also supports the resource and RF event detectors introduced in EEM 2.2, but it does not support the enhanced object tracking event detector or the actions to read and set tracked objects.

Embedded Event Manager 2.2

EEM 2.2 is supported in Cisco IOS Release 12.4(2)T, 12.2(31)SB3, 12.2(33)SRB, and later releases, and introduced some new features. EEM 2.2 introduced the following event detectors:

- Enhanced Object Tracking—The enhanced object tracking event detector publishes an event when the tracked object changes. Enhanced object tracking provides complete separation between the objects to be tracked and the action to be taken by a client when a tracked object changes.
- Resource—The resource event detector publishes an event when the Embedded Resource Manager (ERM) reports an event for the specified policy.
- RF—The redundancy framework (RF) event detector publishes an event when one or more RF events occur during synchronization in a dual Route Processor (RP) system. The RF event detector can also detect an event when a dual RP system continuously switches from one RP to another RP (referred to as a ping-pong situation).

EEM 2.2 introduced the following actions:

- Reading the state of a tracked object.
- Setting the state of a tracked object.

Embedded Event Manager 2.3

EEM 2.3 is supported in Cisco IOS Release 12.2(33)SXH and later releases for the Cisco Catalyst 6500 Series switches and introduces enhancements to the Generic Online Diagnostics (GOLD) Event Detector on that product.

- The **event gold** command was enhanced with the addition of the **action-notify**, **testing-type**, **test-name**, **test-id**, **consecutive-failure**, **platform-action**, and **maxrun** keywords for improved reaction to GOLD test failures and conditions.
- The following platform-wide GOLD Event Detector information can be accessed through new read-only EEM built-in environment variables:
  - Boot-up diagnostic level
  - Card index, name, serial number
  - Port counts
  - Test counts
- The following test-specific GOLD Event Detector information can be accessed through new read-only EEM built-in environment variables (available to EEM applets only):
  - Test name, attribute, total run count
  - Test result per test, port, or device
  - Total failure count, last fail time
  - Error code
  - Occurrence of consecutive failures

These enhancements result in reduced mean time to recovery (MTTR) and higher availability through improved automation and fault detection.

Embedded Event Manager 2.4

EEM 2.4 is supported in Cisco IOS Release 12.4(20)T, 12.2(33)SXI, 12.2(33)SRE and later releases, and introduced several new features. EEM 2.4 introduced the following event detectors:

- SNMP Notification—The SNMP notification event detector provides the ability to intercept SNMP trap and inform messages coming into the router. An SNMP notification event is generated when an incoming SNMP trap or inform message matches specified values or crosses specified thresholds.
- RPC—The remote procedure call (RPC) event detector provides the ability to invoke EEM policies from outside the router over an encrypted connection using Secure Shell (SSH). The RPC event detector uses Simple Object Access Protocol (SOAP) data encoding for exchanging XML-based messages. This event detector can be used to run EEM policies and then receive output in a SOAP XML-formatted reply.

EEM 2.4 added enhancements to the following event detectors:

- Interface counter rate-based trigger—This feature adds the ability for an interface event to be triggered based on a rate of change over a period of time. A rate can be specified both for the entry value and the exit value. This feature copies the rate-based functionality that currently exists for the SNMP event detector.

- SNMP delta value—The difference between the monitored Object Identifier (OID) value at the beginning of the monitored period and the actual OID value when the event is published will be provided in the **event_reqinfo** data for both the SNMP event detector and the Interface Counter event detector.

EEM 2.4 introduced the following actions:

- Multiple event support—The ability to run multiple events was introduced. and **show event manager** commands were enhanced to show multiple events.
- Support for parameters—The *parameter* argument has been added to the **event manager run** command. A maximum of 15 parameters can be used.
- Display of Job IDs and completion status—Some of the **show event manager** commands were enhanced to display Job IDs and completion status.
- Bytecode support—Tcl 8 defines a specialized bytecode language (BCL) and includes a just-in-time compiler that translates Tcl scripts to BCL. Byte sequence is executed by a "virtual machine," Tcl_ExecuteByteCode(), or TEBC for short, as often as needed. Currently EEM accepts file extensions, such as *.tcl for user policies and *.tm for system policies. Tcl standard extension for bytecode scripts are *.tbc. Now EEM will accept *.tbc as valid EEM policies.
- Registration substitution enhancement—Supports replacing multiple parameters in the event registration statement lines with a single environment variable.
- Tcl package support

Embedded Event Manager 3.0

EEM 3.0 is supported in Cisco IOS Release 12.4(22)T, 12.2(33)SRE, and later releases.

EEM 3.0 introduces the following new event detectors:

- Custom CLI—The custom CLI event detector publishes an event to add and enhance existing CLI command syntax.
- Routing—The Routing event detector publishes an event when route entries change in the Routing Information Base (RIB).
- NetFlow— The NetFlow event detector publishes an event when a NetFlow event is triggered.
- IP SLA—The IP SLA event detector publishes an event when an IP SLA reaction is triggered.

EEM 3.0 introduces the following features.

- Class-based scheduling—The EEM policies will be assigned a class using the **class** keyword when they are registered. EEM policies registered without a class will be assigned to the default class.
- High performance Tcl policies—Three new Tcl commands are introduced **event_completion**, **event_wait**, and **event_completion_with_wait**.
- Interactive cli support—The synchronous applets are enhanced to support interaction with the local console (TTY). Two new IOS commands, **action gets** and **action puts,** are introduced to allow users to enter and display input directly on the console.
- Variable logic for applets—The Variable Logic for EEM Applets feature adds the ability to apply conditional logic within EEM applets. Conditional logic introduces a control structure that can change the flow of actions within applets depending on conditional expressions.
- Digital signature support—A new API performs digital signature verification for a Tcl script to check it the script is signed by Cisco before execution.

- Support authenticating e-mail servers—The **action mail** command is modified to include an optional username and password.
- SMTP IPv6 support—The keyword **sourceaddr** is added in Tcl e-mail templates to specify either an IPv6 or IPv4 address.
- SNMP library extensions—The EEM applet **action info** and Tcl **sys_reqinfo_snmp** commands are enhanced to include functionality for SNMP getid, inform, trap, and set-type operations.
- SNMP Notification IPv6 support—IPv6 address is supported for the source and destination IP addresses.
- CLI Library XML-PI support—Provides a programmable interface which encapsulates IOS command-line interface (CLI) show commands in XML format in a consistent way across different Cisco products. Customers using XML-PI will be able to parse IOS show command output from within Tcl scripts using well-known keywords instead of having to depend on the use of regular expression support.

Embedded Event Manager 3.1

EEM 3.1 is supported in Cisco IOS Release 15.0(1)M and later releases, and introduces sthe following:

EEM 3.1 introduced one new event detector:

- SNMP Object—The Simple Network Management Protocol (SNMP) object trap event detector provides an extension to replace the value when an SNMP trap with the specified SNMP object ID (OID) is encountered on a specific interface or address.

EEM 3.1 added an enhancement to the following event detector:

- SNMP Notification—The SNMP notification event detector now can wait and intercept the outgoing SNMP traps and informs.

EEM 3.1 added enhancement to the following action:

- Specify facility—The **action syslog** command has been enhanced to specify syslog facility.

EEM 3.1 introduces the following features:

- Provides the ability to create a short description for the registered policy—A new **description** command has been introduced to register policies with a brief description in Cisco IOS CLI and Tcl policies. The **show event manager policy available** command and the **show event manager policy registered** command have been enhanced to add the **description** keyword to display the description of the registered applet.
- Enables EEM policies to bypass AAA authorization—The **event manager application** command has been enhanced to provide authorization and bypass keywords to disable AAA.
- Introduces CLI Library enhancements—Provides two new commands in the CLI library: **cli_run** and **cli_run_interactive**.

EEM Event Detectors Available by Cisco IOS Release

EEM uses software programs known as event detectors to determine when an EEM event occurs. Some event detectors are available on every Cisco IOS release, but most event detectors have been introduced in a specific release. Use Table 1 to determine which event detectors are available in your specific Cisco IOS release. A blank entry (—) indicates that the event detector is not available: the text “Yes”

indicates that the event detector is available. The event detectors shown in Table 1 are supported in later releases of the same Cisco IOS release train. For more details on each event detector, see the Event Detectors concept in the "Embedded Event Manager Overview" module.

Table 1 Availability of Event Detectors by Cisco IOS Release

Event Detector	12.2(25)S	12.3(14)T 12.2(18)SXF5 12.2(28)SB 12.2(33)SRA	12.4(2)T 12.2(31)SB3 12.2(33)SRB	12.2(18)SXF4 Cisco IOS Software Modularity	12.2(33)SXH	12.4(20)T 12.2(33)SXI	12.4(22)T 12.2(33)SRE	15.0(1)M
Application-Specific	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
CLI	—	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Counter	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Custom CLI	—	—	—	—	—	—	Yes	Yes
Enhanced Object Tracking	—	—	Yes	—	Yes	Yes	Yes	Yes
GOLD	—	—	—	Yes	Yes	Yes	Yes	Yes
Interface Counter	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
IPSLA	—	—	—	—	—	—	Yes	Yes
NF	—	—	—	—	—	—	Yes	Yes
None	—	Yes	Yes	Yes	Yes	Yes	Yes	Yes
OIR	—	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Resource	—	—	Yes	Yes	Yes	Yes	Yes	Yes
RF	—	—	Yes	Yes	Yes	Yes	Yes	Yes
Routing	—	—	—	—	—	—	Yes	Yes
RPC	—	—	—	—	—	Yes	Yes	Yes
SNMP	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SNMP Notification	—	—	—	—	—	Yes	Yes	Yes
SNMP Object	—	—	—	—	—	—	—	Yes
Syslog	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
System Manager	—	—	—	Yes	Yes	Yes	Yes	Yes
Timer	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
IOSWDSysMon (Cisco IOS watchdog)	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
WDSysMon (Cisco IOS Software Modularity watchdog)	—	—	—	Yes	—	—	—	

Event Detectors

Embedded Event Manager (EEM) uses software programs known as *event detectors* to determine when an EEM event occurs. Event detectors are separate systems that provide an interface between the agent being monitored, for example Simple Network Management Protocol (SNMP), and the EEM policies where an action can be implemented. Some event detectors are available on every Cisco IOS release, but most event detectors have been introduced in a specific release. For details of which event detector is supported in each Cisco IOS release, see the EEM Event Detectors Available by Cisco IOS Release concept in the "Writing Embedded Event Manager Policies Using the Cisco IOS CLI" or the "Writing Embedded Event Manager Policies Using Tcl" modules. EEM contains the following event detectors.

Application-Specific Event Detector

The application-specific event detector allows any Embedded Event Manager policy to publish an event. When an EEM policy publishes an event it must use an EEM subsystem number of 798 with any event type. If an existing policy is registered for subsystem 798 and a specified event type, a second policy of the same event type will trigger the first policy to run when the specified event is published.

CLI Event Detector

The CLI event detector screens command-line interface (CLI) commands for a regular expression match. When a match is found, an event is published. The match logic is performed on the fully expanded CLI command after the command is successfully parsed and before it is executed. The CLI event detector supports three publish modes:

- Synchronous publishing of CLI events—The CLI command is not executed until the EEM policy exits, and the EEM policy can control whether the command is executed. The read/write variable, _exit_status, allows you to set the exit status at policy exit for policies triggered from synchronous events. If _exit_status is 0, the command is skipped, if _exit_status is 1, the command is run.
- Asynchronous publishing of CLI events—The CLI event is published, and then the CLI command is executed.
- Asynchronous publishing of CLI events with command skipping—The CLI event is published, but the CLI command is not executed.

Counter Event Detector

The counter event detector publishes an event when a named counter crosses a specified threshold. There are two or more participants that affect counter processing. The counter event detector can modify the counter, and one or more subscribers define the criteria that cause the event to be published. After a counter event has been published, the counter monitoring logic can be reset to start monitoring the counter immediately or it can be reset when a second threshold—called an exit value—is crossed.

Custom CLI Event Detector

The custom CLI event detector publishes an event to add and enhance existing CLI command syntax. When the special parser characters Tab, ? (question mark), and Enter are entered, the parser sends the input to the custom CLI event detector for processing. The custom CLI event detector then compares this input against registered strings to determine if this is a new or enhanced CLI command. Upon a match the custom CLI event detector takes appropriate actions, such as displaying help for the command if ? is entered, displaying the entire command if Tab is entered, or executing the command if Enter was entered. If a match does not occur, the parser regains control and processes the information as usual.

Enhanced Object Tracking Event Detector

The enhanced object tracking (EOT) event detector publishes an event when the status of a tracked object changes. Object tracking was first introduced into the Hot Standby Router Protocol (HSRP) as a simple tracking mechanism that allowed you to track the interface line-protocol state only. If the line-protocol state of the interface went down, the HSRP priority of the router was reduced, allowing another HSRP router with a higher priority to become active.

Object tracking was enhanced to provide complete separation between the objects to be tracked and the action to be taken by a client when a tracked object changes. Thus, several clients such as HSRP, VRRP, or GLBP can register their interest with the tracking process, track the same object, and each take different action when the object changes. Each tracked object is identified by a unique number that is specified on the tracking command-line interface (CLI). Client processes use this number to track a specific object. The tracking process periodically polls the tracked objects and notes any change of value. The changes in the tracked object are communicated to interested client processes, either immediately or after a specified delay. The object values are reported as either up or down.

Enhanced object tracking is now integrated with EEM to allow EEM to report on a status change of a tracked object and to allow enhanced object tracking to track EEM objects. A new type of tracking object—a stub object—is created. The stub object can be manipulated using the existing CLI commands that already allow tracked objects to be manipulated.

GOLD Event Detector

The GOLD event detector publishes an event when a GOLD failure event is detected on a specified card and subcard.

Interface Counter Event Detector

The interface counter event detector publishes an event when a generic Cisco IOS interface counter for a specified interface crosses a defined threshold. A threshold can be specified as an absolute value or an incremental value. If the incremental value is set to 50, for example, an event would be published when the interface counter increases by 50.

After an interface counter event has been published, the interface counter monitoring logic is reset using two methods. The interface counter is reset either when a second threshold—called an exit value—is crossed or when an elapsed period of time occurs.

IP SLA Event Detector

The IP SLA event detector publishes an event when an IP SLA reaction is triggered.

NetFlow Event Detector

The NetFlow event detector publishes an event when a NetFlow event is triggered.

None Event Detector

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

OIR Event Detector

The online insertion and removal (OIR) event detector publishes an event when one of the following hardware insertion or removal events occurs:

- A card is removed.
- A card is inserted.

Route Processors (RPs), line cards, or feature cards can be monitored for OIR events.

Resource Event Detector

The resource event detector publishes an event when the Embedded Resource Manager (ERM) reports an event for the specified policy. The ERM infrastructure tracks resource depletion and resource dependencies across processes and within a system to handle various error conditions. The error conditions are handled by providing an equitable sharing of resources between various applications. The ERM framework provides a communication mechanism for resource entities and allows communication between these resource entities from numerous locations. The ERM framework also helps in debugging CPU and memory-related issues. The ERM monitors system resource usage to better understand scalability needs by allowing you to configure threshold values for resources such as the CPU, buffers, and memory. The ERM event detector is the preferred method for monitoring resources in Cisco IOS software but the ERM event detector is not supported in Software Modularity images. For more details about ERM, go to "Embedded Resource Manager" module.

RF Event Detector

The redundancy framework (RF) event detector publishes an event when one or more RF events occur during synchronization in a dual Route Processor (RP) system. The RF event detector can also detect an event when a dual RP system continuously switches from one RP to another RP (referred to as a ping-pong situation).

RPC Event Detector

The remote procedure call (RPC) event detector provides the ability to invoke EEM policies from outside the router over an encrypted connection using Secure Shell (SSH). The RPC event detector uses Simple Object Access Protocol (SOAP) data encoding for exchanging XML-based messages. This event detector can be used to run EEM policies and then receive output in a SOAP XML-formatted reply.

Routing Event Detector

The routing event detector publishes an event when a route entry changes in the Routing Information Base (RIB).

SNMP Event Detector

The SNMP event detector allows a standard SNMP MIB object to be monitored and an event to be generated when the object matches specified values or crosses specified thresholds.

SNMP Notification Event Detector

The SNMP notification event detector provides the ability to intercept SNMP trap and inform messages coming into or going out of the router. An SNMP notification event is generated when an incoming or outgoing SNMP trap or inform message matches specified values or crosses specified thresholds. The SNMP event detector can wait and intercept the outgoing SNMP traps and informs.

SNMP Object Event Detector

The Simple Network Management Protocol (SNMP) object trap event detector provides an extension to replace the value when an SNMP trap with the specified SNMP object ID (OID) is encountered on a specific interface or address.

Syslog Event Detector

The syslog event detector allows for screening syslog messages for a regular expression pattern match. The selected messages can be further qualified, requiring that a specific number of occurrences be logged within a specified time. A match on a specified event criteria triggers a configured policy action.

System Manager Event Detector

The system manager event detector generates events for Cisco IOS Software Modularity process start, normal or abnormal stop, and restart events. The events generated by the system manager allows policies to change the default behavior of the process restart.

Timer Event Detector

The timer event detector publishes events for the following four different types of timers:

- An absolute-time-of-day timer publishes an event when a specified absolute date and time occurs.
- A countdown timer publishes an event when a timer counts down to zero.
- A watchdog timer publishes an event when a timer counts down to zero and then the timer automatically resets itself to its initial value and starts to count down again.
- A CRON timer publishes an event using a UNIX standard CRON specification to indicate when the event is to be published. A CRON timer never publishes events more than once per minute.

Watchdog System Monitor (IOSWDSysMon) Event Detector for Cisco IOS

The Cisco IOS watchdog system monitor event detector publishes an event when one of the following occurs:

- CPU utilization for a Cisco IOS task crosses a threshold.
- Memory utilization for a Cisco IOS task crosses a threshold.

Note Cisco IOS processes are now referred to as tasks to distinguish them from Cisco IOS Software Modularity processes.

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

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

The Cisco IOS Software Modularity watchdog system monitor event detector detects infinite loops, deadlocks, and memory leaks in Cisco IOS Software Modularity processes.

EEM Actions Available by Cisco IOS Release

The CLI-based corrective actions that are taken when event detectors report events enable a powerful on-device event management mechanism. Some actions are available in every Cisco IOS release, but most actions have been introduced in a specific release. Use Table 2 to determine which actions are available in your specific Cisco IOS release. A blank entry (—) indicates that the action is not available; the text “Yes” indicates that the action is available. The actions shown in Table 2 are supported in later releases of the same Cisco IOS release train. For more details on each action, see the Embedded Event Manager Actions concept in the “Embedded Event Manager Overview” module.

Table 2 Availability of Actions by Cisco IOS Release

Action	12.2(25)S	12.3(14)T 12.2(18)SXF5 12.2(28)SB 12.2(33)SRA	12.4(2)T 12.2(31)SB3 12.2(33)SRB	12.2(18)SXF4 Cisco IOS Software Modularity	12.2(33)SXH	12.4(20)T	12.4(22)T	15.0(1)M
Execute a CLI command	—	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Generate a CNS event	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Generate a prioritized syslog message	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Generate an SNMP trap	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Manually run an EEM policy	—	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Publish an application-specific event	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Read the state of a tracked object	—	—	Yes	—		Yes	Yes	Yes
Reload the Cisco IOS software	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Request system information	—	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Send a short e-mail	—	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Set or modify a named counter	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Set the state of a tracked object	—	—	Yes	—		Yes	Yes	Yes
Switch to a secondary RP	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Embedded Event Manager Actions

The CLI-based corrective actions that are taken when event detectors report events enable a powerful on-device event management mechanism. Some EEM actions are available on every Cisco IOS release, but most EEM actions have been introduced in a specific release. For details of which EEM action is supported in each Cisco IOS release, see the EEM Actions Available by Cisco IOS Release concept in the "Writing Embedded Event Manager Policies Using the Cisco IOS CLI" or the "Writing Embedded Event Manager Policies Using Tcl" modules. EEM supports the following actions:

- Executing a Cisco IOS command-line interface (CLI) command.
- Generating a CNS event for upstream processing by Cisco CNS devices.
- Setting or modifying a named counter.
- Switching to a secondary processor in a fully redundant hardware configuration.
- Requesting system information when an event occurs.

- Sending a short e-mail.
- Manually running an EEM policy.
- Publishing an application-specific event.
- Reloading the Cisco IOS software.
- Generating an SNMP trap.
- Generating prioritized syslog messages.
- Reading the state of a tracked object.
- Setting the state of a tracked object.

EEM action CLI commands contain an EEM action label that is a unique identifier that can be any string value. Actions are sorted and run in ascending alphanumeric (lexicographical) key sequence using the label as the sort key. If you are using numbers as labels be aware that alphanumerical sorting will sort 10.0 after 1.0, but before 2.0, and in this situation we recommend that you use numbers such as 01.0, 02.0, and so on, or use an initial letter followed by numbers.

Embedded Event Manager Environment Variables

EEM allows environment variables to be used in EEM policies. Tool Command Language (Tcl) allows global variables to be defined that are known to all procedures within a Tcl script. EEM allows environment variables to be defined using a CLI command, the **event manager environment** command, for use within an EEM policy. All EEM environment variables are automatically assigned to Tcl global variables before a Tcl script is run. There are three different types of environment variables associated with Embedded Event Manager:

- User-defined—Defined by you if you create an environment variable in a policy that you have written.
- Cisco-defined—Defined by Cisco for a specific sample policy.
- Cisco built-in (available in EEM applets)—Defined by Cisco and can be read only or read/write. The read only variables are set by the system before an applet starts to execute. The single read/write variable, _exit_status, allows you to set the exit status at policy exit for policies triggered from synchronous events.

Cisco-defined environment variables (see Table 3) and Cisco system-defined environment variables may apply to one specific event detector or to all event detectors. Environment variables that are user-defined or defined by Cisco in a sample policy are set using the **event manager environment** command. Variables that are used in the EEM policy must be defined before you register the policy. A Tcl policy contains a section called “Environment Must Define” that can be defined to check that any required environment variables are defined before the policy runs.

Cisco built-in environment variables are a subset of the Cisco-defined environment variables and the built-in variables are available to EEM applets only. The built-in variables can be read-only or can be read and write, and these variables may apply to one specific event detector or to all event detectors. For more details and a table listing the Cisco system-defined variables, see the ““Writing Embedded Event Manager Policies Using the Cisco IOS CLI” module.

Note Cisco-defined environment variables begin with an underscore character (_). We strongly recommend that customers avoid the same naming convention to prevent naming conflicts.

Table 3 describes the Cisco-defined variables used in the sample EEM policies. Some of the environment variables do not have to be specified for the corresponding sample policy to run and these are marked as optional.

Table 3 Cisco-Defined Environmental Variables and Examples

Environment Variable	Description	Example
_config_cmd1	The first configuration command that is executed.	**interface Ethernet1/0**
_config_cmd2	(Optional) The second configuration command that is executed.	**no shutdown**
_crash_reporter_debug	(Optional) A value that identifies whether debug information for tm_crash_reporter.tcl will be enabled.	1
_crash_reporter_url	The URL location to which the crash report is sent.	http://www.yourdomain.com/fm/interface_tm.cgi
_cron_entry	A CRON specification that determines when the policy will run. See the "Writing Embedded Event Manager Policies Using Tcl" module for more information about how to specify a cron entry.	0-59/1 0-23/1 * * 0-7
_email_server	A Simple Mail Transfer Protocol (SMTP) mail server used to send e-mail.	mailserver.yourdomain.com
_email_to	The address to which e-mail is sent.	engineer@yourdomain.com
_email_from	The address from which e-mail is sent.	devtest@yourdomain.com
_email_cc	The address to which the e-mail is be copied.	manager@yourdomain.com
_email_ipaddr	The source IP address of the recipient.	209.165.201.1 or (IPv6 address) 2001:0DB8::1
_info_snmp_oid	The SNMP object ID.	1.3.6.1.2.1.2 or iso.internet.mgmt.mib-2.interfaces
_info_snmp_value	The value string of the associated SNMP data element.	
_show_cmd	The CLI **show** command to be executed when the policy is run.	**show version**
_syslog_pattern	A regular expression pattern match string that is used to compare syslog messages to determine when the policy runs.	.*UPDOWN.*FastEthernet 0/0.*
_tm_fsys_usage_cron	(Optional) A CRON specification that is used in the **event_register** keyword extension. If unspecified, the _tm_fsys_usage.tcl policy is triggered once per minute.	0-59/1 0-23/1 * * 0-7

Table 3 Cisco-Defined Environmental Variables and Examples (continued)

Environment Variable	Description	Example
_tm_fsys_usage_debug	(Optional) When this variable is set to a value of 1, disk usage information is displayed for all entries in the system.	1
_tm_fsys_usage_ freebytes	(Optional) Free byte threshold for systems or specific prefixes. If free space falls below a given value, a warning is displayed.	disk2:98000000
_tm_fsys_usage_percent	(Optional) Disk usage percentage thresholds for systems or specific prefixes. If disk usage percentage exceeds a given percentage, a warning is displayed. If unspecified, the default disk usage percentage is 80 percent for all systems.	nvram:25 disk2:5

Embedded Event Manager Policy Creation

EEM is a policy driven process in which the EEM policy engine receives notifications when faults and other events occur in the Cisco IOS software system. Embedded Event Manager policies implement recovery based on the current state of the system and the actions specified in the policy for a given event. Recovery actions are triggered when the policy is run.

Although there are some EEM CLI configuration and **show** commands, EEM is implemented through the creation of policies. An EEM policy is an entity that defines an event and the actions to be taken when that event occurs. There are two types of EEM policies: an applet or a script. An applet is a simple form of policy that is defined within the CLI configuration. A script is a form of policy that is written in Tcl.

The creation of an EEM policy involves:

- Selecting the event for which the policy is run.
- Defining the event detector options associated with logging and responding to the event.
- Defining the environment variables, if required.
- Choosing the actions to be performed when the event occurs.

There are two ways to create an EEM policy. The first method is to write applets using CLI commands, and the second method is to write Tcl scripts. Cisco provides enhancements to Tcl in the form of Tcl command extensions that facilitate the development of EEM policies. Scripts are defined off the networking device using an ASCII editor. The script is then copied to the networking device and registered with EEM. When a policy is registered with the Embedded Event Manager, the software examines the policy and registers it to be run when the specified event occurs. Policies can be unregistered or suspended. Both types of policies can be used to implement EEM in your network.

For details on writing EEM policies using the Cisco IOS CLI, go to "Writing Embedded Event Manager Policies Using the Cisco IOS CLI" module.

For details on writing EEM policies using Tcl, go to "Writing Embedded Event Manager Policies Using Tcl" module.

Where to Go Next

- If you want to write EEM policies using the Cisco IOS CLI, see the "Writing Embedded Event Manager Policies Using the Cisco IOS CLI" module.
- If you want to write EEM policies using Tcl, see the "Writing Embedded Event Manager Policies Using Tcl" module.

Additional References

The following sections provide references related to EEM.

Related Documents

Related Topic	Document Title
Cisco IOS commands	*Cisco IOS Master Commands List, All Releases*
Network Management commands (including EEM commands): complete command syntax, defaults, command mode, command history, usage guidelines, and examples	*Cisco IOS Network Management Command Reference*
Embedded Event Manager policy writing using the CLI	Writing Embedded Event Manager Policies Using the Cisco IOS CLI module
Embedded Event Manager policy writing using Tcl	Writing Embedded Event Manager Policies Using Tcl module
Embedded Resource Manager	Embedded Resource Manager module

Standards

Standard	Title
No new or modified standards are supported, and support for existing standards has not been modified.	—

MIBs

MIB	MIBs Link
CISCO-EMBEDDED-EVENT-MGR-MIB	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

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

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for Embedded Event Manager Overview

Table 4 lists the features in this module and provides links to specific configuration information. Only features that were introduced or modified in Cisco IOS Releases 12.3(14)T, 12.2(25)S, 12.0(26)S, 12.2(18)SXF4, 12.2(28)SB, 12.2(33)SRA, 12.2(33)SXH, 12.2(33)SXI, 12.4(20)T, 12.4(22)T, 15.0(1)M, 12.2(33)SRE or a later release appear in the table.

Not all commands may be available in your Cisco IOS software release. For release information about a specific command, see the command reference documentation.

Use Cisco Feature Navigator to find information about platform support and software image support. Cisco Feature Navigator enables you to determine which Cisco IOS and Catalyst OS software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Note Table 4 lists only the Cisco IOS software release that introduced support for a given feature in a given Cisco IOS software release train. Unless noted otherwise, subsequent releases of that Cisco IOS software release train also support that feature.

Table 4 Feature Information for Embedded Event Manager Overview

Feature Name	Releases	Feature Configuration Information
Embedded Event Manager 1.0	12.0(26)S 12.3(4)T	EEM 1.0 introduced Embedded Event Manager applet creation with the SNMP and Syslog event detectors. EEM 1.0 also introduced the following actions: generating prioritized syslog messages, generating a CNS event for upstream processing by Cisco CNS devices, reloading the Cisco IOS software, and switching to a secondary processor in a fully redundant hardware configuration. The following sections provide information about this feature: • Embedded Event Manager 1.0, page 3 • Event Detectors, page 10 • Embedded Event Manager Actions, page 14 • Embedded Event Manager Policy Creation, page 17 The following commands were introduced by this feature: **action cns-event**, **action force-switchover**, **action reload**, **action syslog**, **debug event manager**, **event manager applet**, **event snmp**, **event syslog**, **show event manager policy registered**.

Table 4 Feature Information for Embedded Event Manager Overview (continued)

Feature Name	Releases	Feature Configuration Information
Embedded Event Manager 2.0	12.2(25)S	EEM 2.0 introduced the application-specific event detector, the counter event detector, the interface counter event detector, the timer event detector, and the IOSWDSysMon event detector. New actions include setting and modifying a named counter, publishing an application-specific event, and generating an SNMP trap. The ability to define environment variables and to run a sample EEM policy (included in the software) written using Tcl was introduced. The following sections provide information about this feature: • Embedded Event Manager 2.0, page 4 • Event Detectors, page 10 • Embedded Event Manager Actions, page 14 • Embedded Event Manager Environment Variables, page 15 • Embedded Event Manager Policy Creation, page 17 The following commands were introduced by this feature: **action counter**, **action publish-event**, **action snmp-trap**, **event application**, **event counter**, **event interface**, **event ioswdsysmon**, **event manager environment**, **event manager history size**, **event manager policy**, **event manager scheduler suspend**, **event timer**, **show event manager environment, show event manager history events, show event manager history traps, show event manager policy available**, **show event manager policy pending**.

Table 4 Feature Information for Embedded Event Manager Overview (continued)

Feature Name	Releases	Feature Configuration Information
Embedded Event Manager 2.1	12.3(14)T 12.2(18)SXF5 12.2(28)SB 12.2(33)SRA	EEM 2.1 introduced some new event detectors and actions with new functionality to allow EEM policies to be run manually and the ability to run multiple concurrent policies. Support for Simple Network Management Protocol (SNMP) event detector rate-based events was provided as was the ability to create policies using Tool Command Language (Tcl). The following sections provide information about this feature: • Embedded Event Manager 2.1, page 4 • Event Detectors, page 10 • Embedded Event Manager Actions, page 14 • Embedded Event Manager Environment Variables, page 15 • Embedded Event Manager Policy Creation, page 17 The following commands were introduced or modified by this feature: **action cli**, **action counter**, **action info**, **action mail**, **action policy**, **debug event manager**, **event cli**, **event manager directory user**, **event manager policy**, **event manager run**, **event manager scheduler script**, **event manager session cli username**, **event none**, **event oir**, **event snmp**, **event syslog**, **set** (EEM), **show event manager directory user, show event manager policy registered**, **show event manager session cli username**.

Table 4 Feature Information for Embedded Event Manager Overview (continued)

Feature Name	Releases	Feature Configuration Information
Embedded Event Manager 2.1 (Software Modularity)	12.2(18)SXF4 Cisco IOS Software Modularity images	EEM 2.1 for Software Modularity images introduced the GOLD, system manager, and WDSysMon (Cisco IOS Software Modularity watchdog) event detectors, and the ability to display Cisco IOS Software Modularity processes and process metrics. The following sections provide information about this feature: • Embedded Event Manager 2.1 (Software Modularity), page 5 • Event Detectors, page 10 • Embedded Event Manager Actions, page 14 • Embedded Event Manager Environment Variables, page 15 • Embedded Event Manager Policy Creation, page 17 The following commands were introduced by this feature: **event gold**, **event process**, **show event manager metric process**. **Note** EEM 2.1 for Software Modularity images also supports the resource and RF event detectors introduced in EEM 2.2, but it does not support the enhanced object tracking event detector or the actions to read and set tracked objects.
Embedded Event Manager 2.2	12.4(2)T 12.2(31)SB3 12.2(33)SRB	EEM 2.2 introduced the enhanced object tracking, resource, and RF event detectors. The actions of reading and setting the state of a tracked object were also introduced. The following sections provide information about this feature: • Embedded Event Manager 2.2, page 5 • Event Detectors, page 10 • Embedded Event Manager Actions, page 14 • Embedded Event Manager Environment Variables, page 15 • Embedded Event Manager Policy Creation, page 17 The following commands were introduced or modified by this feature: **action track read**, **action track set**, **default-state**, **event resource**, **event rf**, **event track**, **show track**, **track stub-object**.

Table 4 Feature Information for Embedded Event Manager Overview (continued)

Feature Name	Releases	Feature Configuration Information
Embedded Event Manager 2.3	12.2(33)SXH 12.2(33)SB	EEM 2.3 is supported in Cisco IOS Release 12.2(33)SXH and later releases for the Cisco Catalyst 6500 Series switches and introduces enhancements to the Generic Online Diagnostics (GOLD) Event Detector on that product. The following sections provide information about this feature: • Embedded Event Manager 2.3, page 6 • Event Detectors, page 10 • Embedded Event Manager Actions, page 14 • Embedded Event Manager Environment Variables, page 15 • Embedded Event Manager Policy Creation, page 17 The **event gold** command was enhanced in addition to the Tcl keywords—**action-notify**, **testing-type**, **test-name**, **test-id**, **consecutive-failure**, **platform-action**, and **maxrun**—for improved reaction to GOLD test failures and conditions.
Embedded Event Manager 2.4	12.4(20)T 12.2(33)SXI 2.2(33)SRE	EEM 2.4 is supported in Cisco IOS Release 12.4(20)T and later releases, and introduced several new features. The following sections provide information about this feature: • Embedded Event Manager 2.4, page 6 • Event Detectors, page 10 • Embedded Event Manager Actions, page 14 • Embedded Event Manager Environment Variables, page 15 • Embedded Event Manager Policy Creation, page 17 The following commands were introduced by this feature: **attribute (EEM)**, **correlate**, **event manager detector rpc**, **event manager directory user repository**, **event manager update user policy**, **event manager scheduler clear**, **event manager update user policy**, **event owner**, **event rpc**, **event snmp-notification**, **show event manager detector**, **show event manager version**, **trigger (EEM)**.

Table 4 Feature Information for Embedded Event Manager Overview (continued)

Feature Name	Releases	Feature Configuration Information
Embedded Event Manger 3.0	12.4(22)T 12,2(33)SRE	EEM 3.0 is supported in Cisco IOS Release 12.4(22)T and later releases, and introduced several new features. The following sections provide information about this feature: • Embedded Event Manager 3.0, page 7 • Event Detectors, page 10 • Embedded Event Manager Actions, page 14 • Embedded Event Manager Environment Variables, page 15 • Embedded Event Manager Policy Creation, page 17 The following commands were introduced or modified by this feature: **action add**, **action append**, **action break**, **action comment**, **action context retrieve**, **action context save**, **action continue**, **action decrement**, **action divide**, **action else**, **action elseif**, **action end**, **action exit**, **action foreach**, **action gets**, **action if**, **action if goto**, **action increment**, **action info type interface-names**, **action info type snmp getid**, **action info type snmp inform**, **action info type snmp oid**, **action info type snmp trap**, **action info type snmp var**, **action multiply**, **action puts**, **action regexp**, **action set (EEM)**, **action string compare**, **action string equal**, **action string first**, **action string index**, **action string last**, **action string length**, **action string match**, **action string range**, **action string replace**, **action string tolower**, **action string toupper**, **action string trim**, **action string trimleft**, **action string trimright**, **action subtract**, **action while**, **event cli**, **event ipsla**, **event manager detector routing**, **event manager scheduler**, **event manager scheduler clear**, **event manager scheduler hold**, **event manager scheduler modify**, **event manager scheduler release**, **event nf**, **event routing**, **show event manager policy active**, **show event manager policy pending**, and **show event manager scheduler**.

Table 4 Feature Information for Embedded Event Manager Overview (continued)

Feature Name	Releases	Feature Configuration Information
Embedded Event Manager 3.1	15.0(1)M	EEM 3.1 is supported in Cisco IOS Release 15.0(1)M and later releases, and introduced several new features. The following sections provide information about this feature: • Embedded Event Manager 3.1, page 8 • Event Detectors, page 10 • Embedded Event Manager Actions, page 14 • Embedded Event Manager Environment Variables, page 15 • Embedded Event Manager Policy Creation, page 17 The following commands were introduced or modified by this feature: **action syslog**, **description (EEM)**, **event manager applet**, **event manager policy**, **event snmp-notification**, **event snmp-object**, **show event manager policy registered**, and **show event manager policy available**.

Writing Embedded Event Manager Policies Using the Cisco IOS CLI

First Published: October 31, 2005
Last Updated: December 9, 2009

This module describes how to write Embedded Event Manager (EEM) policies using Cisco IOS command-line interface (CLI) applets to handle Cisco IOS software faults and events. EEM is a distributed and customized approach to event detection and recovery offered directly in a Cisco IOS device. EEM offers the ability to monitor events and take informational, corrective, or any desired action when the monitored events occur or when a threshold is reached. The EEM policy engine receives notifications when faults and other events occur. EEM policies implement recovery on the basis of the current state of the system and the actions specified in the policy for a given event. Recovery actions are triggered when the policy is run.

Finding Feature Information in This Module

Your Cisco IOS software release may not support all of the features documented in this module. To reach links to specific feature documentation in this module and to see a list of the releases in which each feature is supported, use the "Feature Information for Writing EEM Policies Using the Cisco IOS CLI" section on page 76.

Finding Support Information for Platforms and Cisco IOS and Catalyst OS Software Images

Use Cisco Feature Navigator to find information about platform support and Cisco IOS and Catalyst OS software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Contents

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

Prerequisites for Writing EEM Policies Using the Cisco IOS CLI

- Before writing EEM policies, you should be familiar with the concepts explained in the "Embedded Event Manager Overview" module.
- If the **action cns-event** command is used, access to a Cisco Networking Services (CNS) Event gateway must be configured.
- If the **action force-switchover** command is used, a secondary processor must be configured on the device.
- If the **action snmp-trap** command is used, the **snmp-server enable traps event-manager** command must be enabled to permit SNMP traps to be sent from the Cisco IOS device to the SNMP server. Other relevant **snmp-server** commands must also be configured; for details see the **action snmp-trap** command page.

Information About Writing EEM Policies Using the Cisco IOS CLI

To write EEM policies using the Cisco IOS CLI, you should understand the following concepts:

Embedded Event Manager Policies

EEM offers the ability to monitor events and take informational or corrective action when the monitored events occur or a threshold is reached. An EEM policy is an entity that defines an event and the actions to be taken when that event occurs. There are two types of EEM policies: an applet or a script. An applet is a simple form of policy that is defined within the CLI configuration. A script is a form of policy that is written in Tool Command Language (Tcl).

EEM Applet

An EEM applet is a concise method for defining event screening criteria and the actions to be taken when that event occurs. In applet configuration mode, three types of configuration statements are supported. The **event** commands are used to specify the event criteria to trigger the applet to run, the **action** commands are used to specify an action to perform when the EEM applet is triggered, and the **set** command is used to set the value of an EEM applet variable. Currently only the _exit_status variable is supported for the **set** command.

Only one **event** configuration command is allowed within an applet configuration. When applet configuration mode is exited and no **event** command is present, a warning is displayed stating that no event is associated with this applet. If no event is specified, this applet is not considered registered. When

no action is associated with this applet, events are still triggered but no actions are performed. Multiple **action** configuration commands are allowed within an applet configuration. Use the **show event manager policy registered** command to display a list of registered applets.

Before modifying an EEM applet, be aware that the existing applet is not replaced until you exit applet configuration mode. While you are in applet configuration mode modifying the applet, the existing applet may be executing. It is safe to modify the applet without unregistering it. When you exit applet configuration mode, the old applet is unregistered and the new version is registered.

The action configuration commands are uniquely identified using the *label* argument, which can be any string value. Actions are sorted in ascending alphanumeric key sequence using the *label* argument as the sort key, and they are run using this sequence.

The Embedded Event Manager schedules and runs policies on the basis of an event specification that is contained within the policy itself. When applet configuration mode is exited, EEM examines the **event** and **action** commands that are entered and registers the applet to be run when a specified event occurs.

EEM Script

Scripts are defined off the networking device using an ASCII editor. The script is then copied to the networking device and registered with EEM. Tcl scripts are supported by EEM.

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

- Selecting the event for which the policy is run.
- Defining the event detector options associated with logging and responding to the event.
- Choosing the actions to be followed when the event occurs.

Cisco provides enhancements to Tcl in the form of keyword extensions that facilitate the development of EEM policies. The main categories of keywords identify the detected event, the subsequent action, utility information, counter values, and system information. For more details about writing EEM policies using Tcl, see the "Writing Embedded Event Manager Policies Using Tcl" module.

Embedded Event Manager Built-In Environment Variables Used in EEM Applets

EEM built-in environment variables are a subset of the Cisco-defined environment variables and the built-in variables are available to EEM applets only. The built-in variables can be read-only or can be read and write and these variables may apply to one specific event detector or to all event detectors. Table 1 lists the Cisco built-in environment variables that are read-only alphabetically by event detector and subevent.

Table 1 EEM Built-In Environment Variables (Read Only)

Environment Variable	Description
All Events	
_event_id	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
_event_type	Type of event.
_event_type_string	An ASCII string identifier of the event type that triggered the event.
_event_pub_sec **_event_pub_msec**	The time, in seconds and milliseconds, at which the event was published to the EEM.

Table 1 EEM Built-In Environment Variables (Read Only) (continued)

Environment Variable	Description
_event_severity	The severity of the event.
Application-Specific Event Detector	
_application_component_id	The event application component identifier.
_application_data1	The value of an environment variable, character text, or a combination of the two to be passed to an application-specific event when the event is published.
_application_data2	The value of an environment variable, character text, or a combination of the two to be passed to an application-specific event when the event is published.
_application_data3	The value of an environment variable, character text, or a combination of the two to be passed to an application-specific event when the event is published.
_application_data4	The value of an environment variable, character text, or a combination of the two to be passed to an application-specific event when the event is published.
_application_sub_system	The event application subsystem number.
_application_type	The type of application.
CLI Event Detector	
_cli_msg	The fully expanded message that triggered the CLI event.
_cli_msg_count	The number of times that a message match occurred before the event was published.
Counter Event Detector	
_counter_name	The name of the counter.
_counter_value	The value of the counter.
Enhanced Object Tracking Event Detector	
_track_number	The number of the tracked object.
_track_state	The state of the tracked object; down or up.
GOLD Event Detector	
_action_notify	The action notify information in a GOLD event flag; either false or true.
_event_severity	The event severity which can be one of the following; normal, minor, or major.
_gold_bl	The boot diagnostic level, which can be one of the following values: • 0: complete diagnostic • 1: minimal diagnostic • 2: bypass diagnostic
_gold_card	The card on which a GOLD failure event was detected.

Table 1 EEM Built-In Environment Variables (Read Only) (continued)

Environment Variable	Description
_gold_cf*testnum*	Consecutive failure, where *testnum* is the test number. For example, **_gold_cf3** is the EEM built-in environment variable for consecutive failure of test 3.
_gold_ci	Card index.
_gold_cn	Card name.
_gold_ec*testnum*	Test error code, where *testnum* is the test number. For example, **_gold_ec3** is the EEM built-in environment variable for the error code of test 3.
_gold_lf*testnum*	Last fail time, where *testnum* is the test number. For example, **_gold_lf3** is the EEM built-in variable for the last fail time of test 3. The time-stamp format is *mmm dd yyyy hh*:*mm*:*ss*. For example, Mar 11 2005 08:47:00.
_gold_new_failure	The new test failure information in a GOLD event flag; either true or false.
_gold_overall_result	The overall diagnostic result, which can be one of the following values: • 0: OK • 3: minor error • 4: major error • 14: unknown result
_gold_pc	Port counts.
_gold_rc*testnum*	Test total run count, where *testnum* is the test number. For example, **_gold_rc3** is the EEM built-in variable for the total run count of test 3.
_gold_sn	Card serial number.
_gold_sub_card	The subcard on which a GOLD failure event was detected.
_gold_ta*testnum*	Test attribute, where *testnum* is the test number. For example, **_gold_ta3** is the EEM built-in variable for the test attribute of test 3.
_gold_tc	Test counts.
_gold_tf*testnum*	Total failure count, where *testnum* is the test number. For example, **_gold_tf3** is the EEM built-in variable for the total failure count of test 3.
_gold_tn*testnum*	Test name, where *testnum* is the test number. For example, **_gold_tn3** is the EEM built-in variable for the name of test 3.

Table 1 EEM Built-In Environment Variables (Read Only) (continued)

Environment Variable	Description
_gold_tr*testnum*	Test result, where *testnum* is the test number. For example, **_gold_tr6** is the EEM built-in variable for test 6, where test 6 is not a per-port test and not a per-device test. The test result is one of the following values: • P: diagnostic result Pass • F: diagnostic result Fail • U: diagnostic result Unknown
_gold_tr*testnum***d***devnum*	Per-device test result, where *testnum* is the test number and *devnum* is the device number. For example, **_gold_tr3d20** is the EEM built-in variable for the test result for test 3, device 20. The test result is one of the following values: • P: diagnostic result Pass • F: diagnostic result Fail • U: diagnostic result Unknown
_gold_tr*testnum***p***portnum*	Per-port test result, where *testnum* is the test number and *portnum* is the port number. For example, **_gold_tr5p20** is the EEM built-in variable for the test result for test 5, port 20. The test result is one of the following values: • P: diagnostic result Pass • F: diagnostic result Fail • U: diagnostic result Unknown
_gold_tt	The testing type, which can be one of the following: • 1: a boot diagnostic • 2: an on-demand diagnostic • 3: a schedule diagnostic • 4: a monitoring diagnostic
Interface Counter Event Detector	
_interface_is_increment	A value to indicate whether the current interface counter value is an absolute value (0) or an increment value (1).
_interface_name	The name of the interface to be monitored.
_interface_parameter	The name of the interface counter to be monitored.
_interface_value	A value with which the current interface counter value is compared.
None Event Detector	
_event_id	A value of 1 indicates an insertion event; a value of 2 indicates a removal event.

Table 1 EEM Built-In Environment Variables (Read Only) (continued)

Environment Variable	Description
_none_argc **_none_arg1** **_none_arg2** **_none_arg3** **_none_arg4** **_none_arg5** **_none_arg6** **_none_arg7** **_none_arg8** **_none_arg9** **_none_arg10** **_none_arg11** **_none_arg12** **_none_arg13** **_none_arg14** **_none_arg15**	The parameters that are passed from the XML SOAP command to the script.
OIR Event Detector	
_oir_event	A value of 1 indicates an insertion event; a value of 2 indicates a removal event.
_oir_slot	The slot number for the OIR event.
Resource Event Detector	
_resource_configured_threshold	The configured ERM threshold.
_resource_current_value	The current value reported by ERM.
_resource_dampen_time	The ERM dampen time, in nanoseconds.
_resource_direction	The ERM event direction. The event direction can be one of the following: up, down, or no change.
_resource_level	The ERM event level. The four event levels are normal, minor, major, and critical.
_resource_notify_data_flag	The ERM notify data flag.
_resource_owner_id	The ERM resource owner ID.
_resource_policy_id	The ERM policy ID.
_resource_policy_violation_flag	The ERM policy violation flag; either false or true.
_resource_time_sent	The ERM event time, in nanoseconds.
_resource_user_id	The ERM resource user ID.
RF Event Detector	
_rf_event	A value of 0 indicates that this is not an RF event; a value of 1 indicates an RF event.

Table 1 EEM Built-In Environment Variables (Read Only) (continued)

Environment Variable	Description
RPC Event Detector	
_rpc_event	A value of 0 indicates that there is no error; a value of 1 to 83 indicates error.
_rpc_argc **_rpc_arg0** **_rpc_arg1** **_rpc_arg2** **_rpc_arg3** **_rpc_arg4** **_rpc_arg5** **_rpc_arg6** **_rpc_arg7** **_rpc_arg8** **_rpc_arg9** **_rpc_arg10** **_rpc_arg11** **_rpc_arg12** **_rpc_arg13** **_rpc_arg14**	The parameters that are passed from the XML SOAP command to the applet.
SNMP Event Detector	
_snmp_exit_event	A value of 0 indicates that this is not an exit event; a value of 1 indicates an exit event.
_snmp_oid	The SNMP object ID that caused the event to be published.
_snmp_oid_delta_val	The actual incremental difference between the value of the current SNMP object ID and the value when the event was last triggered.
_snmp_oid_val	The SNMP object ID value when the event was published.
SNMP Notification Event Detector	
_snmp_notif_oid	A user specified object ID.
_snmp_notif_oid_val	A user specified object ID value.
_snmp_notif_src_ip_addr	The source IP address of the SNMP Protocol Data Unit (PDU).
_snmp_notif_dest_ip_addr	The destination IP address of the SNMP PDU.
_x_x_x_x_x_x_x(varbinds)	The SNMP PDU varbind information.
_snmp_notif_trunc_vb_buf	Indicates whether the varbind information has been truncated due to the lack of space in the buffer.

Table 1 EEM Built-In Environment Variables (Read Only) (continued)

Environment Variable	Description
Syslog Event Detector	
_syslog_msg	The syslog message that caused the event to be published.
System Manager (Process) Event Detector	
_process_dump_count	The number of times that a Posix process was dumped.
_process_exit_status	The status of the Posix process at exit.
_process_fail_count	The number of times that a Posix process failed.
_process_instance	The instance number of the Posix process.
_process_last_respawn	The Posix process that was last respawned.
_process_node_name	The node name of the Posix process.
_process_path	The path of the Posix process.
_process_process_name	The name of the Posix process.
_process_respawn_count	The number of times that a Posix process was respawned.
Timer Event Detector	
_timer_remain	The time available before the timer expires. **Note** This environment variable is not available for the CRON timer.
_timer_time	The time at which the last event was triggered.
_timer_type	The type of timer.
Watchdog System Monitor (IOSWDSysMon) Event Detector	
_ioswd_node	The slot number for the Route Processor (RP) reporting node.
_ioswd_num_subs	The number of subevents present.
All Watchdog System Monitor (IOSWDSysMon) Subevents	
_ioswd_sub1_present **_ioswd_sub2_present**	A value to indicate whether subevent 1 or subevent 2 is present. A value of 1 means that the subevent is present; a value of 0 means that the subevent is not present.
_ioswd_sub1_type **_ioswd_sub2_type**	The event type, either cpu_proc or mem_proc.
Watchdog System Monitor (IOSWDSysMon) cpu_proc Subevents	
_ioswd_sub1_path **_ioswd_sub2_path**	A process name of subevents.
_ioswd_sub1_period **_ioswd_sub2_period**	The time period, in seconds and optional milliseconds, used for measurement in subevents.
_ioswd_sub1_pid **_ioswd_sub2_pid**	The process identifier of subevents.
_ioswd_sub1_taskname **_ioswd_sub2_taskname**	The task name of subevents.

Table 1 EEM Built-In Environment Variables (Read Only) (continued)

Environment Variable	Description
_ioswd_sub1_value **_ioswd_sub2_value**	The CPU utilization of subevents measured as a percentage.
Watchdog System Monitor (IOSWDSysMon) mem_proc Subevents	
_ioswd_sub1_diff **_ioswd_sub2_diff**	A percentage value of the difference that triggered the event. **Note** This variable is set only when the **_ioswd_sub1_is_percent** or **_ioswd_sub2_is_percent** variable contains a value of 1.
_ioswd_sub1_is_percent **_ioswd_sub2_is_percent**	A number that identifies whether the value is a percentage. A value of 0 means that the value is not a percentage; a value of 1 means that the value is a percentage.
_ioswd_sub1_path **_ioswd_sub2_path**	The process name of subevents.
_ioswd_sub1_pid **_ioswd_sub2_pid**	The process identifier of subevents.
_ioswd_sub1_taskname **_ioswd_sub2_taskname**	The task name of subevents.
_ioswd_sub1_value **_ioswd_sub2_value**	The CPU utilization of subevents measured as a percentage.
Watchdog System Monitor (WDSysMon) Event Detector	
_wd_sub1_present **_wd_sub2_present**	A value to indicate whether subevent 1 or subevent 2 is present. A value of 1 means that the subevent is present; a value of 0 means that the subevent is not present.
_wd_num_subs	The number of subevents present.
_wd_sub1_type **_wd_sub2_type**	The event type: cpu_proc, cpu_tot, deadlock, dispatch_mgr, mem_proc, mem_tot_avail, or mem_tot_used.
Watchdog System Monitor (WDSysMon) cpu_proc Subevents	
_wd_sub1_node **_wd_sub2_node**	The slot number for the subevent RP reporting node.
_wd_sub1_period **_wd_sub2_period**	The time period, in seconds and optional milliseconds, used for measurement in subevents.
_wd_sub1_procname **_wd_sub2_procname**	The process name of subevents.
_wd_sub1_value **_wd_sub2_value**	The CPU utilization of subevents measured as a percentage.
Watchdog System Monitor (WDSysMon) cpu_tot Subevents	
_wd_sub1_node **_wd_sub2_node**	The slot number for the subevent RP reporting node.

Table 1 EEM Built-In Environment Variables (Read Only) (continued)

Environment Variable	Description
_wd_sub1_period **_wd_sub2_period**	The time period, in seconds and optional milliseconds, used for measurement in subevents.
_wd_sub1_value **_wd_sub2_value**	The CPU utilization of subevents measured as a percentage.
Watchdog System Monitor (WDSysMon) deadlock Subevents	
_wd_sub1_entry_[1-N]_**b_node** **_wd_sub2_entry**_[1-N]_**b_node**	The slot number for the subevent RP reporting node.
_wd_sub1_entry_[1-N]_**b_pid** **_wd_sub2_entry**_[1-N]_**b_pid**	The process identifier of subevents.
_wd_sub1_entry_[1-N]_**b_procname** **_wd_sub2_entry**_[1-N]_**b_procname**	The process name of subevents.
_wd_sub1_entry_[1-N]_**b_tid** **_wd_sub2_entry**_[1-N]_**b_tid**	The time identifier of subevents.
_wd_sub1_entry_[1-N]_**node** **_wd_sub2_entry**_[1-N]_**node**	The slot number for the subevent RP reporting node.
_wd_sub1_entry_[1-N]_**pid** **_wd_sub2_entry**_[1-N]_**pid**	The process identifier of subevents.
_wd_sub1_entry_[1-N]_**procname** **_wd_sub2_entry**_[1-N]_**procname**	The process name of subevents.
_wd_sub1_entry_[1-N]_**state** **_wd_sub2_entry**_[1-N]_**state**	The time identifier of subevents.
_wd_sub1_entry_[1-N]_**tid** **_wd_sub2_entry**_[1-N]_**tid**	The time identifier of subevents.
_wd_sub1_num_entries **_wd_sub2_num_entries**	The number of subevents.
Watchdog System Monitor (WDSysMon) dispatch manager Subevents	
_wd_sub1_node **_wd_sub2_node**	The slot number for the subevent RP reporting node.
_wd_sub1_period **_wd_sub2_period**	The time period, in seconds and optional milliseconds, used for measurement in subevents.
_wd_sub1_procname **_wd_sub2_procname**	The process name of subevents.
_wd_sub1_value **_wd_sub2_value**	The CPU utilization of subevents measured as a percentage.

Table 1 EEM Built-In Environment Variables (Read Only) (continued)

Environment Variable	Description
Watchdog System Monitor (WDSysMon) mem_proc Subevents	
_wd_sub1_diff **_wd_sub2_diff**	A percentage value of the difference that triggered the event. **Note** This variable is set only when the **_wd_sub1_is_percent** or **_wd_sub2_is_percent** variable contains a value of 1.
_wd_sub1_is_percent **_wd_sub2_is_percent**	A number that identifies whether the value is a percentage. A value of 0 means that the value is not a percentage; a value of 1 means that the value is a percentage.
_wd_sub1_node **_wd_sub2_node**	The slot number for the subevent RP reporting node.
_wd_sub1_period **_wd_sub2_period**	The time period, in seconds and optional milliseconds, used for measurement in subevents.
_wd_sub1_pid **_wd_sub2_pid**	The process identifier of subevents.
_wd_sub1_procname **_wd_sub2_procname**	The process name of subevents.
_wd_sub1_value **_wd_sub2_value**	The CPU utilization of subevents measured as a percentage.
Watchdog System Monitor (WDSysMon) mem_tot_avail and mem_tot_used Subevents	
_wd_sub1_avail **_wd_sub2_avail**	The memory available for subevents.
_wd_sub1_diff **_wd_sub2_diff**	A percentage value of the difference that triggered the event. **Note** This variable is set only when the **_wd_sub1_is_percent** or **_wd_sub2_is_percent** variable contains a value of 1.
_wd_sub1_is_percent **_wd_sub2_is_percent**	A number that identifies whether the value is a percentage. A value of 0 means that the value is not a percentage; a value of 1 means that the value is a percentage.
_wd_sub1_node **_wd_sub2_node**	The slot number for the subevent RP reporting node.
_wd_sub1_period **_wd_sub2_period**	The time period, in seconds and optional milliseconds, used for measurement in subevents.
_wd_sub1_value **_wd_sub2_value**	The CPU utilization of subevents measured as a percentage.
_wd_sub1_used **_wd_sub2_used**	The memory used by subevents.

How to Write EEM Policies Using the Cisco IOS CLI

This section contains the following tasks:

Registering and Defining an Embedded Event Manager Applet

Perform this task to register an applet with Embedded Event Manager and to define the EEM applet using the Cisco IOS CLI **event** and **action** commands. Only one **event** command is allowed in an EEM applet. Multiple **action** commands are permitted. If no **event** and no **action** commands are specified, the applet is removed when you exit configuration mode.

The SNMP event detector and the syslog **action** commands used in this task are just representing any event detector and **action** commands. For examples using other event detectors and **action** commands, see the "Embedded Event Manager Applet Configuration: Examples" section on page 60.

EEM Environment Variables

EEM environment variables for EEM policies are defined using the EEM **event manager environment** configuration command. By convention, all Cisco EEM environment variables begin with "_". In order to avoid future conflict, customers are urged not to define new variables that start with "_".

You can display the EEM environment variables set on your system by using the **show event manager environment** privileged EXEC command.

For example, you can create EEM policies that can send e-mails when an event occurs. Table 2 describes the e-mail-specific environment variables that can be used in EEM policies.

Table 2 EEM E-mail-Specific Environmental Variables

Environment Variable	Description	Example
_email_server	A Simple Mail Transfer Protocol (SMTP) mail server used to send e-mail.	The e-mail server name—Mailservername— can be in any one of the following template formats: • username:password@host • username@host • host
_email_to	The address to which e-mail is sent.	engineering@example.com
_email_from	The address from which e-mail is sent.	devtest@example.com
_email_cc	The address to which the e-mail is copied.	manager@example.com

Alphabetical Order of EEM Action Labels

An EEM action label is a unique identifier that can be any string value. Actions are sorted and run in ascending alphanumeric (lexicographical) key sequence using the label as the sort key. If you are using numbers as labels be aware that alphanumerical sorting will sort 10.0 after 1.0, but before 2.0, and in this situation we recommend that you use numbers such as 01.0, 02.0, and so on, or use an initial letter followed by numbers.

SUMMARY STEPS

1. **enable**
2. **show event manager environment** [**all** | *variable-name*]
3. **configure terminal**
4. **event manager environment** *variable-name string*
5. Repeat Step 4 for all the required environment variables.
6. **event manager applet** *applet-name*
7. **event snmp oid** *oid-value* **get-type** {**exact** | **next**} **entry-op** *operator* **entry-val** *entry-value* [**exit-comb** {**or** | **and**}] [**exit-op** *operator*] [**exit-val** *exit-value*] [**exit-time** *exit-time-value*] **poll-interval** *poll-int-value*
8. **action** *label* **cli command** *cli-string* [**pattern** *pattern-string*]
9. **action** *label* **syslog** [**priority** *priority-level*] **msg** *msg-text* **facility** *string*
10. **action** *label* **mail server** *server-address* **to** *to-address* **from** *from-address* [**cc** *cc-address*] **subject** *subject* **body** *body-text*
11. Add more action commands as required.
12. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show event manager environment** [**all** | *variable-name*] **Example:** `Router# show event manager environment all`	(Optional) Displays the name and value of EEM environment variables. • The optional **all** keyword displays all the EEM environment variables. • The optional *variable-name* argument displays information about the specified environment variable.
Step 3	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 4	**event manager environment** *variable-name string* **Example:** `Router(config)# event manager environment _email_to engineering@example.com`	Configures the value of the specified EEM environment variable. • In this example, the environment variable that holds the e-mail address to which e-mail is sent is set to engineering@example.com.
Step 5	Repeat Step 4 for all the required environment variables.	Repeat Step 4 to configure all the environment variables required by the policy to be registered in Step 6.
Step 6	**event manager applet** *applet-name* **Example:** `Router(config)# event manager applet memory-fail`	Registers the applet with the Embedded Event Manager (EEM) and enters applet configuration mode.
Step 7	**event snmp oid** *oid-value* **get-type** {**exact** | **next**} **entry-op** *operator* **entry-val** *entry-value* [**exit-comb** {**or** | **and**}] [**exit-op** *operator*] [**exit-val** *exit-value*] [**exit-time** *exit-time-value*] **poll-interval** *poll-int-value* **Example:** `Router(config-applet)# event snmp oid 1.3.6.1.4.1.9.9.48.1.1.1.6.1 get-type exact entry-op lt entry-val 5120000 poll-interval 90`	Specifies the event criteria that cause the EEM applet to run. • In this example, an EEM event is triggered when free memory falls below the value of 5120000. • Exit criteria are optional, and if not specified, event monitoring is reenabled immediately.

	Command or Action	Purpose

Step 8	**action** *label* **cli command** *cli-string* [**pattern** *pattern-string*]
	
	Example:
	`Router(config-applet)# action 1.0 cli command "enable"`
	`Router(config-applet)# action 2.0 cli command "clear counters Ethernet0/1" pattern "confirm"`
	`Router(config-applet)# action 3.0 cli command "y"`
		Specifies the action of executing a Cisco IOS CLI command when an EEM applet is triggered.
		
		The **pattern** keyword is optional and is used only when the command string solicits input. The **action cli** command ends when the solicited prompt as specified in the optional **pattern** keyword is received. You are required to specify a regular expression pattern that will match the next solicited prompt. Specification of an incorrect pattern will cause the **action cli** command to wait forever until the applet execution times out due to the maxrun timer expiration.
		
		• The action taken is to specify an EEM applet to run when the **pattern** keyword specifies the *confirm* argument for the **clear counters Ethernet0/1** command. In this case the command string solicits input, such as "confirm," which has to be completed with a "yes" or a "no" input.

Step 9	**action** *label* **syslog** [**priority** *priority-level*] **msg** *msg-text* **facility** *string*
	
	Example:
	`Router(config-applet)# action 1.0 syslog priority critical msg "Memory exhausted; current available memory is $_snmp_oid_val bytes"`
	
	Example:
	`Router(config-applet)# action 1.0 syslog priority errors facility EEM-FAC message "TEST MSG"`
		Specifies the action to be taken when an EEM applet is triggered.
		
		In this example, the action taken is to write a message to syslog.
		
		• The optional **priority** keyword specifies the priority level of the syslog messages. If selected, the *priority-level* argument must be defined.
		
		• The *msg-text* argument can be character text, an environment variable, or a combination of the two.
		
		• The **facility** keyword specifies the location of generated message
		
		• The *string* argument can be character text, an environment variable, or a combination of the two.

Step 10	**action** *label* **mail server** *server-address* **to** *to-address* **from** *from-address* [**cc** *cc-address*] **subject** *subject* **body** *body-text* **Example:** `Router(config-applet)# action 2.0 mail server 192.168.1.10 to engineering@example.com from devtest@example.com subject "Memory failure" body "Memory exhausted; current available memory is $_snmp_oid_val bytes"`	Specifies the action of sending a short e-mail when an EEM applet is triggered. • The *server-address* argument specifies the fully qualified domain name of the e-mail server to be used to forward the e-mail. • The *to-address* argument specifies the e-mail address where the e-mail is to be sent. • The *from-address* argument specifies the e-mail address from which the e-mail is sent. • The *subject* argument specifies the subject line content of the e-mail as an alphanumeric string. • The *body-text* argument specifies the text content of the e-mail as an alphanumeric string.

	Command or Action	Purpose
Step 11	Add more action commands as required.	—
Step 12	**end** **Example:** `Router(config-applet)# end`	Exits applet configuration mode and returns to privileged EXEC mode.

Troubleshooting Tips

Use the **debug event manager** command in privileged EXEC mode to troubleshoot EEM command operations. Use any debugging command with caution as the volume of generated output can slow or stop the router operations. We recommend that this command be used only under the supervision of a Cisco engineer.

Registering and Defining an Embedded Event Manager Policy to Run Manually

There are two ways to manually run an EEM policy. EEM usually schedules and runs policies on the basis of an event specification that is contained within the policy itself. The **event none** command allows EEM to identify an EEM policy that can be manually triggered. To run the policy, use either the **action policy** command in applet configuration mode or the **event manager run** command in privileged EXEC mode.

Perform this task to register an EEM policy to be run manually using the **event manager run** command. For an example of how to manually run a policy using the **action policy** command, see the "Embedded Event Manager Manual Policy Execution: Examples" section on page 64.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **event manager applet** *applet-name*
4. **event none**
5. **action** *label* **syslog** [**priority** *priority-level*] **msg** *msg-text* **facility** *string*
6. **end**
7. **event manager run** *applet-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**event manager applet** *applet-name* **Example:** `Router(config)# event manager applet manual-policy`	Registers the applet with the Embedded Event Manager and enters applet configuration mode.
Step 4	**event none** **Example:** `Router(config-applet)# event none`	Specifies that an EEM policy is to be registered with the EEM and can be run manually.
Step 5	**action** *label* **syslog** [**priority** *priority-level*] **msg** *msg-text* **facility** *string* **Example:** `Router(config-applet)# action 1.0 syslog msg "Manual-policy triggered"`	Specifies the action to be taken when an EEM applet is triggered. In this example, the action to be taken is to write a message to syslog. • The optional **priority** keyword specifies the priority level of the syslog messages. If selected, the *priority-level* argument must be defined. • The *msg-text* argument can be character text, an environment variable, or a combination of the two. • The **facility** keyword specifies the location of generated message. • The *string* argument can be character text, an environment variable, or a combination of the two.
Step 6	**end** **Example:** `Router(config-applet)# end`	Exits applet configuration mode and returns to privileged EXEC mode.
Step 7	**event manager run** *applet-name* **Example:** `Router# event manager run manual-policy`	Manually runs a registered EEM policy.

Unregistering Embedded Event Manager Policies

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

SUMMARY STEPS

1. **enable**
2. **show event manager policy registered** [**description** [*policy-name*] | **detailed** *policy-filename* [**system** | **user**] | [**event-type** *event-name*] [**system** | **user**] [**time-ordered** | **name-ordered**]]
3. **configure terminal**
4. **no event manager policy** *policy-filename*
5. **exit**
6. Repeat Step 2 to ensure that the policy has been removed.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show event manager policy registered** [**description** [*policy-name*] | **detailed** *policy-filename* [**system** | **user**] | [**event-type** *event-name*] [**system** | **user**] [**time-ordered** | **name-ordered**]] **Example:** `Router# show event manager policy registered`	(Optional) Displays the EEM policies that are currently registered. • The optional **system** and **user** keywords display the registered system and user policies. • If no keywords are specified, EEM registered policies for all event types are displayed in time order.
Step 3	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 4	**no event manager policy** *policy-filename* **Example:** `Router(config)# no event manager policy IPSLAping1`	Removes the EEM policy from the configuration, causing the policy to be unregistered.

	Command or Action	Purpose
Step 5	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode and returns to privileged EXEC mode.
Step 6	Repeat Step 2 to ensure that the policy has been removed. **Example:** `Router# show event manager policy registered`	—

Examples

In the following example, the **show event manager policy registered** privileged EXEC command is used to display the two EEM applets that are currently registered:

```
Router# show event manager policy registered

No.  Class   Type    Event Type          Trap  Time Registered           Name
1    applet  system  snmp                Off   Fri Aug 12 17:42:52 2005  IPSLAping1
 oid {1.3.6.1.4.1.9.9.42.1.2.9.1.6.4} get-type exact entry-op eq entry-val {1}
 exit-op eq exit-val {2} poll-interval 90.000
 action 1.0 syslog priority critical msg "Server IPecho Failed: OID=$_snmp_oid_val"
 action 1.1 snmp-trap strdata "EEM detected server reachability failure to 10.1.88.9"
 action 1.2 publish-event sub-system 88000101 type 1 arg1 "10.1.88.9" arg2 "IPSLAEcho"
arg3 "fail"
 action 1.3 counter name _IPSLA1F op inc value 1
2    applet  system  snmp                Off   Thu Sep 15 05:57:16 2005  memory-fail
 oid {1.3.6.1.4.1.9.9.48.1.1.1.6.1} get-type exact entry-op lt entry-val {5120000}
poll-interval 90
 action 1.0 syslog priority critical msg Memory exhausted; current available memory is
$_snmp_oid_val bytes
 action 2.0 force-switchover
```

In the following example, the **show event manager policy registered** privileged EXEC command is used to show that applet IPSLAping1 has been removed after entering the **no event manager policy** command:

```
Router# show event manager policy registered

No.  Class   Type    Event Type          Trap  Time Registered           Name
1    applet  system  snmp                Off   Thu Sep 15 05:57:16 2005  memory-fail
 oid {1.3.6.1.4.1.9.9.48.1.1.1.6.1} get-type exact entry-op lt entry-val {5120000}
poll-interval 90
 action 1.0 syslog priority critical msg Memory exhausted; current available memory is
$_snmp_oid_val bytes
 action 2.0 force-switchover
```

Suspending All Embedded Event Manager Policy Execution

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

SUMMARY STEPS

1. **enable**
2. **show event manager policy registered** [**description** [*policy-name*] | **detailed** *policy-filename* [**system** | **user**] | [**event-type** *event-name*] [**system** | **user**] [**time-ordered** | **name-ordered**]]
3. **configure terminal**
4. **event manager scheduler suspend**
5. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show event manager policy registered** [**description** [*policy-name*] | **detailed** *policy-filename* [**system** | **user**] | [**event-type** *event-name*] [**system** | **user**] [**time-ordered** | **name-ordered**]] **Example:** `Router# show event manager policy registered`	(Optional) Displays the EEM policies that are currently registered. • The optional **system** and **user** keywords display the registered system and user policies. • If no keywords are specified, EEM registered policies for all event types are displayed in time order.
Step 3	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 4	**event manager scheduler suspend** **Example:** `Router(config)# event manager scheduler suspend`	Immediately suspends the execution of all EEM policies.
Step 5	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode and returns to privileged EXEC mode.

Configuring and Tracking a Stub Object Using Embedded Event Manager

Perform this task to create a stub object, set the state of the stub object, and configure an EEM applet to be run when the tracked object changes. Actions are specified within the EEM applet to both set and read the state of the object. This task allows EEM to define an enhanced object tracking (EOT) object that may be manipulated by other EOT clients. An EEM policy can be a trigger for any EOT object including objects defined for other EOT clients or for an object defined by EEM.

Enhanced Object Tracking

Object tracking was first introduced into the Hot Standby Router Protocol (HSRP) as a simple tracking mechanism that allowed you to track the interface line-protocol state only. Enhanced object tracking provides complete separation between the objects to be tracked and the action to be taken by a client when a tracked object changes. Thus, several clients such as EEM, VRRP, or GLBP can register their interest with the tracking process, track the same object, and each take different action when the object changes.

Each tracked object is identified by a unique number that is specified on the tracking command-line interface (CLI). Client processes use this number to track a specific object. The tracking process periodically polls the tracked objects and notes any change of value. The changes in the tracked object are communicated to interested client processes, either immediately or after a specified delay. The object values are reported as either up or down.

The EOT event detector publishes an event when the tracked object changes.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **track** *object-number* **stub-object**
4. **default-state** {**up** | **down**}
5. **exit**
6. **event manager applet** *applet-name*
7. **event** [*label*] **track** *object-number* [**state** {**up** | **down** | **any**}]
8. **action** *label* **track set** *object-number* **state** {**up** | **down**}
9. **action** *label* **track read** *object-number*
10. **end**
11. **show track** [*object-number* [**brief**]]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**track** *object-number* **stub-object** **Example:** `Router(config)# track 2 stub-object`	Creates a stub object to be tracked using EEM and enters tracking configuration mode. • Use the *object-number* argument to assign a number to the tracked object.

	Command or Action	Purpose
Step 4	**default-state** {**up** | **down**} **Example:** `Router(config-track)# default-state up`	Sets the default state for a stub object. • In this example, the default state of the object is set to up.
Step 5	**exit** **Example:** `Router(config-track)# exit`	Exits tracking configuration mode and returns to global configuration mode.
Step 6	**event manager applet** *applet-name* **Example:** `Router(config)# event manager applet track-two`	Registers an applet with EEM and enters applet configuration mode.
Step 7	**event** [*label*] **track** *object-number* [**state** {**up** | **down** | **any**}] **Example:** `Router(config-applet)# event track 2 state down`	Specifies the event criteria that cause the EEM applet to run. • In this example, an EEM event is triggered when the Cisco IOS Object Tracking subsystem reports that tracked object number 2 transitions from an up state to a down state.
Step 8	**action** *label* **track set** *object-number* **state** {**up** | **down**} **Example:** `Router(config-applet)# action 1.0 track set 2 state up`	Specifies the action to be taken when an EEM applet is triggered. • In this example, the action to be taken is to set the state of tracked object number 2 to up.
Step 9	**action** *label* **track read** *object-number* **Example:** `Router(config-applet)# action 2.0 track read 2`	Specifies the action to be taken when an EEM applet is triggered. • In this example, the action to be taken is to read the state of tracked object number 2. • The _track_state read-only variable gets set when this command is run.
Step 10	**end** **Example:** `Router(config-applet)# end`	Exits applet configuration mode and returns to privileged EXEC mode.
Step 11	**show track** [*object-number* [**brief**]] **Example:** `Router# show track 2`	(Optional) Displays information about objects that are tracked by the tracking process. • The optional *object-number* argument displays tracking information for a specified object. • The optional **brief** keyword displays a single line of information.

Examples

In the following example, the **show track** privileged EXEC command is used to display information about tracked object number 2.

```
Router# show track 2
Track 2
  Stub-object
  State is Up
    1 change, last change 00:00:04, by Undefined
```

Displaying Embedded Event Manager History Data

Perform this optional task to change the size of the history tables and to display EEM history data.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **event manager history size** {**events** | **traps**} [*size*]
4. **exit**
5. **show event manager history events** [**detailed**] [**maximum** *number*]
6. **show event manager history traps** {**server** | **policy**}

DETAILED STEPS

Step 1 **enable**

Enables privileged EXEC mode. Enter your password if prompted.

```
Router> enable
```

Step 2 **configure terminal**

Enters global configuration mode.

```
Router# configure terminal
```

Step 3 **event manager history size** {**events** | **traps**} [*size*]

Use this command to change the size of the EEM event history table or the size of the EEM SNMP trap history table. In the following example, the size of the EEM event history table is changed to 30 entries:

```
Router(config)# event manager history size events 30
```

Step 4 **exit**

Exits global configuration mode and returns to privileged EXEC mode.

```
Router(config)# exit
```

Step 5 **show event manager history events** [**detailed**] [**maximum** *number*]

Use this command to display detailed information about each EEM event, for example:

```
Router# show event manager history events

No.  Time of Event            Event Type          Name
```

```
1    Fri Aug13  21:42:57 2004  snmp               applet: SAAping1
2    Fri Aug13  22:20:29 2004  snmp               applet: SAAping1
3    Wed Aug18  21:54:48 2004  snmp               applet: SAAping1
4    Wed Aug18  22:06:38 2004  snmp               applet: SAAping1
5    Wed Aug18  22:30:58 2004  snmp               applet: SAAping1
6    Wed Aug18  22:34:58 2004  snmp               applet: SAAping1
7    Wed Aug18  22:51:18 2004  snmp               applet: SAAping1
8    Wed Aug18  22:51:18 2004  application        applet: CustApp1
```

Step 6 **show event manager history traps** {**server** | **policy**}

Use this command to display the EEM SNMP traps that have been sent either from the EEM server or from an EEM policy. In the following example, the EEM SNMP traps that were triggered from within an EEM policy are displayed.

```
Router# show event manager history traps policy

No.  Time                      Trap Type          Name
1    Wed Aug18  22:30:58 2004  policy             EEM Policy Director
2    Wed Aug18  22:34:58 2004  policy             EEM Policy Director
3    Wed Aug18  22:51:18 2004  policy             EEM Policy Director
```

Displaying Embedded Event Manager Registered Policies

Perform this optional task to display registered EEM policies.

SUMMARY STEPS

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

DETAILED STEPS

Step 1 **enable**

Enables privileged EXEC mode. Enter your password if prompted.

```
Router> enable
```

Step 2 **show event manager policy registered** [**event-type** *event-name*] [**time-ordered** | **name-ordered**]

Use this command with the **time-ordered** keyword to display information about currently registered policies sorted by time, for example:

```
Router# show event manager policy registered time-ordered

No.  Type    Event Type          Time                    Registered Name
1    applet  snmp                Thu May30 05:57:16 2004 memory-fail
 oid {1.3.6.1.4.1.9.9.48.1.1.1.6.1} get-type exact entry-op lt entry-val
{5120000} poll-interval 90
action 1.0 syslog priority critical msg "Memory exhausted; current available memory
is $_snmp_oid_val bytes"
 action 2.0 force-switchover
2    applet  syslog              Wed Jul16 00:05:17 2004 intf-down
 pattern {.*UPDOWN.*Ethernet1/0.*}
 action 1.0 cns-event msg "Interface state change: $_syslog_msg"
```

Use this command with the **name-ordered** keyword to display information about currently registered policies sorted by name, for example:

```
Router# show event manager policy registered name-ordered

No.  Type    Event Type          Time Registered          Name
1    applet  syslog              Wed Jul16  00:05:17 2004 intf-down
 pattern {.*UPDOWN.*Ethernet1/0.*}
 action 1.0 cns-event msg "Interface state change: $_syslog_msg"
2    applet  snmp                Thu May30 05:57:16 2004   memory-fail
 oid {1.3.6.1.4.1.9.9.48.1.1.1.6.1} get-type exact entry-op lt entry-val
{5120000} poll-interval 90
 action 1.0 syslog priority critical msg "Memory exhausted; current available memory
is $_snmp_oid_val bytes"
 action 2.0 force-switchover
```

Use this command with the **event-type** keyword to display information about currently registered policies for the event type specified in the *event-name* argument, for example:

```
Router# show event manager policy registered event-type syslog

No.  Type    Event Type          Time Registered           Name
1    applet  syslog              Wed Jul16  00:05:17 2004 intf-down
 pattern {.*UPDOWN.*Ethernet1/0.*}
 action 1.0 cns-event msg "Interface state change: $_syslog_msg"
```

Configuring Event SNMP Notification

Perform this task to configure SNMP notifications.

Prerequisites

- You must be running Cisco IOS Release 12.4(20)T, 12.2(33)SXI or later release.
- SNMP event manager must be configured using the **snmp-server manager** command.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **event manager applet** *applet-name*
4. **event** [**tag** *event-tag*] **snmp-notification oid** *oid-string* **oid-val** *comparison-value* **op** *operator* [**maxrun** *maxruntime-number*] [**src-ip-address** *ip-address*] [**dest-ip-address** *ip-address*] [**default** *seconds*] [**direction** {**incoming** | **outgoing**}] [**msg-op** {**drop** | **send**}]
5. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**event manager applet** *applet-name* **Example:** `Router(config)# event manager applet snmp`	Registers the applet with the event manager server and enters applet configuration mode.
Step 4	**event** [**tag** *event-tag*] **snmp-notification oid** *oid-string* **oid-val** *comparison-value* **op** *operator* [**maxrun** *maxruntime-number*] [**src-ip-address** *ip-address*] [**dest-ip-address** *ip-address*] [**default** *seconds*] [**direction** {**incoming** | **outgoing**}] [**msg-op** {**drop** | **send**}] **Example:** `Router(config-applet)# event snmp-notification dest-ip-address 192.168.1.1 oid 1 op eq oid-val 10`	Specifies the event criteria for an Embedded Event Manager (EEM) applet that is run by sampling Simple Network Management Protocol (SNMP) notification.
Step 5	**end** **Example:** `Router(config-applet)# end`	Exits applet configuration mode and returns to privileged EXEC mode.

Configuring Multiple Event Support

The multiple event support feature introduced in Cisco IOS Release 12.4(20)T and later releases, adds the ability to register multiple events in the EEM server. The multiple event support involves one or more event occurrences, one or more tracked object states, and a time period for the event to occur. The event parameters are specified in the CLI commands. The data structure to handle multiple events contains multiple event identifiers and correlation logic. This data is used to register multiple events in the EEM Server.

Setting the Event Configuration Parameters

The **trigger** command enters the trigger applet configuration mode and specifies the multiple event configuration statements for EEM applets. The trigger statement is used to relate multiple event statement using the *tag* argument specified in each event statement. The events are raised based on the specified parameters.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **event manager applet** *applet-name*
4. **event** [**tag** *event-tag*] **cli pattern** *regular-expression* **sync** {**yes** | **no skip** {**yes** | **no**}} [**occurs** *num-occurrences*] [**period** *period-value*] [**maxrun** *maxruntime-number*]
5. **trigger** [**occurs** *occurs-value*] [**period** *period-value*] [**period-start** *period-start-value*] [**delay** *delay-value*]
6. **correlate** {**event** *event-tag* | **track** *object-number*} [*boolean-operator* { **event** *event-tag* | **track** *tracked-object*} …]
7. **attribute tag** *event-tag* [**occurs** *occurs-value*]
8. **action** *label* **cli command** *cli-string*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**event manager applet** *applet-name* **Example:** `Router(config)# event manager applet EventInterface`	Registers an applet with EEM and enters applet configuration mode.
Step 4	**event** [**tag** *event-tag*] **cli pattern** *regular-expression* **sync** {**yes** | **no skip** {**yes** | **no**}} [**occurs** *num-occurrences*] [**period** *period-value*] [**maxrun** *maxruntime-number*] **Example:** `Router(config-applet)# event tag 1.0 cli pattern "show bgp all" sync yes occurs 32 period 60 maxrun 60`	Specifies the event criteria for an EEM applet that is run by matching a Cisco IOS command-line interface (CLI) command.
Step 5	**trigger** [**occurs** *occurs-value*] [**period** *period-value*] [**period-start** *period-start-value*] [**delay** *delay-value*] **Example:** `Router(config-applet)# trigger occurs 1 period-start "0 8 * * 1-5" period 60`	Specifies the complex event configuration parameters for an EEM applet.

	Command or Action	Purpose
Step 6	**correlate** {**event** *event-tag* | **track** *object-number*} [*boolean-operator* {**event** *event-tag* | **track** *tracked-object*} …] **Example:** `Router(config-applet)# correlate event 1.0 or event 2.0 and track 10`	Specifies a complex event correlation in the trigger mode for an EEM applet.
Step 7	**attribute tag** *event-tag* [**occurs** *occurs-value*] **Example:** `Router(config-applet)# attribute tag 1.0 occurs 1`	Specifies up to eight attribute statements to build a complex event for an EEM applet.
Step 8	**action** *label* **cli command** *cli-string* **Example:** `Router(config-applet)# action 1.0 cli command "show mwmory"`	Specifies the action of executing a CLI command when an EEM applet is triggered.

Examples

In the following example, applet is run if the **show bgp all** CLI command and any syslog message that contains the string "COUNT" occurred within a period 60 seconds.

```
event manager applet delay_50
 event tag 1.0 cli pattern "show bgp all" sync yes occurs 32 period 60 maxrun 60
 event tag 2.0 syslog pattern "COUNT"
 trigger occurs 1 delay 50
  correlate event 1.0 or event 2.0
  attribute tag 1.0 occurs 1
  attribute tag 2.0 occurs 1
 action 1.0 cli command "show memory"
 action 2.0 cli command "enable"
 action 3.0 cli command "config terminal"
 action 4.0 cli command " ip route 192.0.2.0 255.255.255.224 192.0.2.12"
 action 91.0 cli command "exit"
 action 99.0 cli command "show ip route | incl 192.0.2.5"
```

Configuring EEM Class-Based Scheduling

To schedule Embedded Event Manager (EEM) policies and set policy scheduling options, perform this task. In this task, two EEM execution threads are created to run applets assigned to the default class.

The EEM policies will be assigned a class using the **class** keyword when they are registered. EEM policies registered without a class will be assigned to the default class. Threads that have default class, will service the default class when the thread is available for work. Threads that are assigned specific class letters will service any policy with a matching class letter when the thread is available for work.

If there is no EEM execution thread available to run the policy in the specified class and a scheduler rule for the class is configured, the policy will wait until a thread of that class is available for execution. Synchronous policies that are triggered from the same input event should be scheduled in the same execution thread.

Prerequisites

To use this feature, you must be running Cisco IOS Release 12.4(22)T or a later release.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **event manager scheduler** {**applet** | **axp** | **call-home**} **thread class** *class-options* **number** *thread-number*
4. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**event manager scheduler** {**applet** | **axp** | **call-home**} **thread class** *class-options* **number** *thread-number* **Example:** `Router(config)# event manager scheduler applet thread class default number 2`	Schedules EEM policies and sets policy scheduling options. • In this example, two EEM execution threads are created to run applets assigned to the default class.
Step 4	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode and returns to privileged EXEC mode.

Holding a Scheduled EEM Policy Event or Event Queue

To hold a scheduled EEM policy event or event queue in the EEM scheduler, perform this task. In this task, all pending EEM policies are displayed. A policy identified using a job ID of 2 is held in the EEM scheduler, and the final step shows that the policy with a job ID of 2 has changed status from pending to held.

Prerequisites

To use this feature, you must be running Cisco IOS Release 12.4(22)T or a later release.

SUMMARY STEPS

1. **enable**

2. **show event manager policy pending** [**queue-type** {**applet** | **call-home** | **axp** | **script**} **class** *class-options* | **detailed**]
3. **event manager scheduler hold** {**policy** *job-id* | **queue-type** {**applet** | **call-home** | **axp** | **script**} **class** *class-options* | **all**} [**processor** {**rp_primary** | **rp_standby**}]
4. **show event manager policy pending** [**queue-type** {**applet** | **call-home** | **axp** | **script**} **class** *class-options* | **detailed**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show event manager policy pending** [**queue-type** {**applet** | **call-home** | **axp** | **script**} **class** *class-options* | **detailed**] **Example:** `Router# show event manager policy pending`	Displays the pending EEM policies.
Step 3	**event manager scheduler hold** {**all** | **policy** *job-id* | **queue-type** {**applet** | **call-home** | **axp** | **script**} **class** *class-options*} [**processor** {**rp_primary** | **rp_standby**}] **Example:** `Router# event manager scheduler hold policy 2`	Holds a scheduled EEM policy event or event queue in the EEM scheduler. • In this example, a policy with a job ID of 2 is put on hold.
Step 4	**show event manager policy pending** [**queue-type** {**applet** | **call-home** | **axp** | **script**} **class** *class-options* | **detailed**] **Example:** `Router# show event manager policy pending`	Displays the status of EEM policy put on hold in Step 3 as held, along with other pending policies.

Examples

The following example shows how to view all pending EEM policies and to hold the EEM policy with a job ID of 2.

```
Router# show event manager policy pending

no. job id status time of event             event type     name
1   1      pend   Thu Sep 7  02:54:04 2006  syslog         applet: one
2   2      pend   Thu Sep 7  02:54:04 2006  syslog         applet: two
3   3      pend   Thu Sep 7  02:54:04 2006  syslog         applet: three

Router# event manager scheduler hold policy 2

Router# show event manager policy pending

no. job id status time of event             event type     name
1   1      pend   Thu Sep 7  02:54:04 2006  syslog         applet: one
2   2      held   Thu Sep 7  02:54:04 2006  syslog         applet: two
```

```
3    3       pend    Thu Sep 7  02:54:04 2006  syslog          applet: three
```

Resuming Execution of EEM Policy Events or Event Queues

To resume the execution of specified EEM policies, perform this task. In this task, the policy that was put on hold in the "Holding a Scheduled EEM Policy Event or Event Queue" section on page 30 is now allowed to resume execution.

Prerequisites

To use this feature, you must be running Cisco IOS Release 12.4(22)T or a later release.

SUMMARY STEPS

1. **enable**
2. **show event manager policy pending**
3. **event manager scheduler release** {**all** | **policy** *policy-id* | **queue-type** {**applet** | **call-home** | **axp** | **script**}} **class** *class-options* [**processor** {**rp_primary** | **rp_standby**}]
4. **show event manager policy pending**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show event manager policy pending** **Example:** `Router# show event manager policy pending`	Displays the pending and held EEM policies. **Note** Only the syntax applicable to this task is used in this example. For more details, see the *Cisco IOS Network Management Command Reference*.
Step 3	**event manager scheduler release** {**all** | **policy** *policy-id* | **queue-type** {**applet** | **call-home** | **axp** | **script**}} **class** *class-options* [**processor** {*rp_primary* | *rp_standby*}] **Example:** `Router# event manager scheduler release policy 2`	Resumes execution of specified EEM policies. • The example shows how to resume the execution of the policy with job ID of 2.
Step 4	**show event manager policy pending** **Example:** `Router# show event manager policy pending`	Displays the status of the EEM policy resumed in Step 3 as pending, along with other pending policies. **Note** Only the syntax applicable to this task is used in this example. For more details, see the *Cisco IOS Network Management Command Reference*.

Examples

The following example shows how to view all pending EEM policies, to specify the policy that will resume execution, and to see that the policy is now back in a pending status.

```
Router# show event manager policy pending

no. job id status time of event             event type     name
1   1      pend   Thu Sep 7  02:54:04 2006  syslog         applet: one
2   2      held   Thu Sep 7  02:54:04 2006  syslog         applet: two
3   3      pend   Thu Sep 7  02:54:04 2006  syslog         applet: three

Rotuer# event manager scheduler release policy 2

Rotuer# show event manager policy pending

no. job id status time of event             event type     name
1   1      pend   Thu Sep 7  02:54:04 2006  syslog         applet: one
2   2      pend   Thu Sep 7  02:54:04 2006  syslog         applet: two
3   3      pend   Thu Sep 7  02:54:04 2006  syslog         applet: three
```

Clearing Pending EEM Policy Events or Event Queues

Perform this task to clear EEM policies that are executing or pending execution. In this task, the EEM policy with a job ID of 2 is cleared from the pending queue. The **show event manager policy pending** command is used to display the policies that are pending before and after the policy is cleared.

Prerequisites

To use this feature, you must be running Cisco IOS Release 12.4(22)T or a later release.

SUMMARY STEPS

1. **enable**
2. **show event manager policy pending**
3. **event manager scheduler clear** {**all** | **policy** *policy-id* | **queue-type** {**applet** | **call-home** | **axp** | **script**}} **class** *class-options* [**processor** {**rp_primary** | **rp_standby**}
4. **show event manager policy pending**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show event manager policy pending** **Example:** `Router# show event manager policy pending`	Displays the pending EEM policies. **Note** Only the syntax applicable to this task is used in this example. For more details, see the *Cisco IOS Network Management Command Reference*.
Step 3	**event manager scheduler clear** {**all** | **policy** *job-id* | **queue-type** {**applet** | **call-home** | **axp** | **script**} **class** *class-options*} [**processor** {**rp_primary** | **rp_standby**}] **Example:** `Router# event manager scheduler clear policy 2`	Clears EEM policies that are executing or pending execution. • In this example, the EEM policy with a job ID of 2 is cleared from the pending queue.
Step 4	**show event manager policy pending** **Example:** `Router# show event manager policy pending`	Displays all the pending EEM policies except the policy cleared in Step 3. **Note** Only the syntax applicable to this task is used in this example. For more details, see the *Cisco IOS Network Management Command Reference*.

Examples

The following example shows how to clear the EEM policy with a job ID of 2 that was pending execution. The **show** commands are used to display the policies that are pending before and after the policy is cleared.

```
Router# show event manager policy pending

no. job id status time of event             event type     name
1   1      pend   Thu Sep 7  02:54:04 2006  syslog         applet: one
2   2      pend   Thu Sep 7  02:54:04 2006  syslog         applet: two
3   3      pend   Thu Sep 7  02:54:04 2006  syslog         applet: three

Router# event manager scheduler clear policy 2

Router# show event manager policy pending

no. job id status time of event             event type     name
1   1      pend   Thu Sep 7  02:54:04 2006  syslog         applet: one
3   3      pend   Thu Sep 7  02:54:04 2006  syslog         applet: three
```

Modifying the Scheduling Parameters of EEM Policy Events or Event Queues

To modify the scheduling parameters of the EEM policies, perform this task. The **show event manager policy pending** command displays policies that are assigned to the B or default class. All the currently pending policies are then changed to class A. After the configuration modification, the **show event manager policy pending** command shows all policies assigned as class A.

Prerequisites

To use this feature, you must be running Cisco IOS Release 12.4(22)T or a later release.

SUMMARY STEPS

1. **enable**
2. **show event manager policy pending**
3. **event manager scheduler modify** {**all** | **policy** *job-id* | **queue-type** {**applet** | **call-home** | **axp** | **script**} | **class** *class-options*} [**queue-priority** {**high** | **last** | **low** | **normal**}] [**processor** {**rp_primary** | **rp_standby**}]
4. **show event manager policy pending**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show event manager policy pending** **Example:** `Router# show event manager policy pending`	Displays the pending EEM policies. **Note** Only the syntax applicable to this task is used in this example. For more details, see the *Cisco IOS Network Management Command Reference*.
Step 3	**event manager scheduler modify** {**all** | **policy** *job-id* | **queue-type** {**applet** | **call-home** | **axp** | **script**} | **class** *class-options*} [**queue-priority** {**high** | **last** | **low** | **normal**}] [**processor** {**rp_primary** | **rp_standby**}] **Example:** `Router# event manager scheduler modify all class A`	Modifies the scheduling parameters of the EEM policies. • In this example, all currently pending EEM policies are assigned to class A.
Step 4	**show event manager policy pending** **Example:** `Router# show event manager policy pending`	Displays the EEM policies modified in Step 3 along with other pending policies. **Note** Only the syntax applicable to this task is used in this example. For more details, see the *Cisco IOS Network Management Command Reference*.

Examples

The following example shows how to modify the scheduling parameters of the EEM policies. In this example, the **show event manager policy pending** command displays policies that are assigned to the B or default class. All the currently pending policies are then changed to class A. After the configuration modification, the **show event manager policy pending** command verifies that all policies are now assigned as class A.

```
Router# show event manager policy pending

no. class   status time of event             event type     name
1   default pend   Thu Sep 7  02:54:04 2006  syslog         applet: one
```

```
2   default pend   Thu Sep 7  02:54:04 2006  syslog         applet: two
3   B       pend   Thu Sep 7  02:54:04 2006  syslog         applet: three

Router# event manager scheduler modify all class A

Router# show event manager policy pending

no. class status time of event             event type     name
1   A     pend   Thu Sep 7  02:54:04 2006  syslog         applet: one
2   A     pend   Thu Sep 7  02:54:04 2006  syslog         applet: two
3   A     pend   Thu Sep 7  02:54:04 2006  syslog         applet: three
```

Verifying Class-Based Scheduled Activities of EEM Policies

To verify the scheduled activities of the EEM policies, use the **show event manager scheduler** command.

Prerequisites

To use this feature, you must be running Cisco IOS Release 12.4(22)T or a later release.

SUMMARY STEPS

1. **show event manager scheduler thread** [**queue-type** {**applet** | **call-home** | **axp** | **script**} **class** *class-options* | **detailed**]

DETAILED STEPS

Step 1 **show event manager scheduler thread** [**queue-type** {**applet** | **call-home** | **axp** | **script**} **class** *class-options* | **detailed**]

This command displays all the EEM execution threads from the scheduler perspective and the details of the running policies. This command includes **detailed** and **queue-type** optional keywords. The following is sample output from this command:

```
Router# show event manager scheduler thread

1 Script threads service class default
  total: 1 running: 1 idle: 0
2 Script threads service class range A-D
  total: 3 running: 0 idle: 3
3 Applet threads service class default
  total: 32 running: 0 idle: 32
4 Applet threads service class W X
  total: 5 running: 0 idle: 5
```

To display the details of the running policies using the scheduler threads use the **detailed** keyword. The following is sample output for this keyword:

```
Router# show event manager scheduler thread detailed

1 Script threads service class default
total: 5 running: 5 idle: 0
1 job id: 12341, pid: 101, name: loop.tcl
2 job id: 12352, pid: 52, name: loop.tcl
3 job id: 12363, pid: 55, name: loop.tcl
4 job id: 12395, pid: 53, name: loop.tcl
5 job id: 12588, pid: 102, name: loop.tcl
```

```
2 Applet threads service class default
total: 32 running: 5 idle: 27
1 job id: 15585, pid: 104, name: WDOG_SYSLG_CNTR_TRACK_INTF_APPL
2 job id: 15586, pid: 105, name: WDOG_SYSLG_CNTR_TRACK_INTF_APPL
3 job id: 15587, pid: 106, name: WDOG_SYSLG_CNTR_TRACK_INTF_APPL
4 job id: 15589, pid: 107, name: WDOG_SYSLG_CNTR_TRACK_INTF_APPL
5 job id: 15590, pid: 80, name: WDOG_SYSLG_CNTR_TRACK_INTF_APPL
```

To display the scheduler threads of a queue-type use the **queue-type** keyword. The following are the sample output for this keyword:

```
Router# show event manager sched thread queue-type applet

1 Applet threads service class default
total: 32 running: 7 idle: 25

Router# show event manager sched thread queue-type applet detailed

1 Applet threads service class default
total: 32 running: 5 idle: 27
1 job id: 15700, pid: 103, name: WDOG_SYSLG_CNTR_TRACK_INTF_APPL
2 job id: 15701, pid: 104, name: WDOG_SYSLG_CNTR_TRACK_INTF_APPL
3 job id: 15703, pid: 106, name: WDOG_SYSLG_CNTR_TRACK_INTF_APPL
4 job id: 15704, pid: 107, name: WDOG_SYSLG_CNTR_TRACK_INTF_APPL
5 job id: 15706, pid: 55, name: WDOG_SYSLG_CNTR_TRACK_INTF_APPL
```

Verifying Class-Based Active EEM Policies

To verify the active or the running EEM policies, use the **show event manager policy active** command.

Prerequisites

To use this feature, you must be running Cisco IOS Release 12.4(22)T or a later release.

SUMMARY STEPS

1. **show event manager policy active** [**queue-type** {**applet** | **call-home** | **axp** | **script**} **class** *class-options* | **detailed**]

DETAILED STEPS

Step 1 **show event manager policy active** [**queue-type** {**applet** | **call-home** | **axp** | **script**} **class** *class-options* | **detailed**]

This command displays only the running EEM policies. This command includes **class**, **detailed** and **queue-type** optional keywords. The following is sample output from this command:

```
Router# show event manager policy active

no. job id p s status time of event event type name
1 12598 N A running Mon Oct29 20:49:37 2007 timer watchdog loop.tcl
2 12609 N A running Mon Oct29 20:49:42 2007 timer watchdog loop.tcl
3 12620 N A running Mon Oct29 20:49:46 2007 timer watchdog loop.tcl
4 12650 N A running Mon Oct29 20:49:59 2007 timer watchdog loop.tcl
5 12842 N A running Mon Oct29 20:51:13 2007 timer watchdog loop.tcl

default class - 6 applet events
no. job id p s status time of event event type name
```

```
1 15852 N A running Mon Oct29 21:11:09 2007 counter WDOG_SYSLG_CNTR_TRACK_INTF_APPL
2 15853 N A running Mon Oct29 21:11:09 2007 counter WDOG_SYSLG_CNTR_TRACK_INTF_APPL
3 15854 N A running Mon Oct29 21:11:10 2007 counter WDOG_SYSLG_CNTR_TRACK_INTF_APPL
4 15855 N A running Mon Oct29 21:11:10 2007 timer watchdog WDOG_SYSLG_CNTR_TRACK_INTF_APPL
5 15856 N A running Mon Oct29 21:11:11 2007 counter WDOG_SYSLG_CNTR_TRACK_INTF_APPL
6 15858 N A running Mon Oct29 21:11:11 2007 counter WDOG_SYSLG_CNTR_TRACK_INTF_APPL
```

Verifying Pending EEM Policies

To verify the EEM policies that are pending for execution, use the **show event manager policy pending** command. In Cisco IOS Release 12.4(22)T, optional keywords were added to this command to specify EEM class-based scheduling options.

Prerequisites

To use this feature, you must be running Cisco IOS Release 12.4(22)T or later release.

SUMMARY STEPS

1. **show event manager policy pending** [**queue-type** {**applet** | **call-home** | **axp** | **script**} **class** *class-options* | **detailed**]

DETAILED STEPS

Step 1 **show event manager policy pending** [**queue-type** {**applet** | **call-home** | **axp** | **script**} **class** *class-options* | **detailed**]

This command displays only the pending policies. This command includes **class**, **detailed** and **queue-type** optional keywords. The following is sample output from this command:

```
Router# show event manager policy pending

no. job id p s status time of event event type name
1    12851   N A    pend    Mon Oct29   20:51:18 2007   timer watchdog   loop.tcl
2    12868   N A    pend    Mon Oct29   20:51:24 2007   timer watchdog   loop.tcl
3    12873   N A    pend    Mon Oct29   20:51:27 2007   timer watchdog   loop.tcl
4    12907   N A    pend    Mon Oct29   20:51:41 2007   timer watchdog   loop.tcl
5    13100   N A    pend    Mon Oct29   20:52:55 2007   timer watchdog   loop.tcl
```

Configring EEM Applet (Interactive CLI) Support

The synchronous applets are enhanced to support interaction with the local console (tty) in Cisco IOS Release 12.4(22)T. Two new commands **action gets** and **action puts** are introduced to allow users to enter and display input directly on the console. The output for synchronous applets will bypass the system logger. The local console will be opened by the applets and serviced by the corresponding synchronous Event Detector pty. Synchronous output will be directed to the opened console.

For details on configuring EEM applet interactive CLI support, see the "Reading and Writing Input from the Active Console for Synchronous EEM Applets" section on page 39.

Reading and Writing Input from the Active Console for Synchronous EEM Applets

The synchronous applets are enhanced to support interaction with the local console (tty) in Cisco IOS Release 12.4(22)T.

Use the following tasks to implement EEM applet interactive CLI support:

- Reading Input from the Active Console, page 39
- Writing Input to the Active Console, page 40

Reading Input from the Active Console

When a synchronous policy is triggered, the related console is stored in the publish information specification. The policy director will query this information in an event_reqinfo call, and store the given console information for use by the **action gets** command.

The **action gets** command reads a line of the input from the active console and stores the input in the variable. The trailing new line will not be returned.

Prerequisites

To use this feature, you must be running Cisco IOS Release 12.4(22)T, or later releases.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **event manager applet** *applet-name*
4. **event none**
5. **action** *label* **gets** *variable*
6. **action** *label* **syslog** [**priority** *priority-level*] **msg** *msg-text*
7. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**event manager applet** *applet-name* **Example:** `Router(config)# event manager applet action`	Registers the applet with the EEM and enters applet configuration mode.

	Command or Action	Purpose
Step 4	**event none** **Example:** `Router(config-applet)# event none`	Specifies that an EEM policy is to be registered with the EEM and can be run manually.
Step 5	**action** *label* **gets** *variable* **Example:** `Router(config-applet)# action label2 gets input`	Gets input from the local console in a synchronous applet and stores the value in the given variable when an EEM applet is triggered.
Step 6	**action** *label* **syslog** [**priority** *priority-level*] **msg** *msg-text* **Example:** `Router(config-applet)# action label3 syslog msg "Input entered was \"$input\""`	Specifies the action to be taken when an EEM applet is triggered. • In this example, the action to be taken is to write the value of the variable specified in Step 5, to syslog.
Step 7	**exit** **Example:** `Router(config-applet)# exit`	Exits applet configuration mode and returns to privileged EXEC mode.

Example

The following example shows how to get the input from the local tty in a synchronous applet and store the value

```
Router(config)# event manager applet action
Router(config-applet)# event none
Router(config-applet)# action label2 gets input
Router(config-applet)# action label3 syslog msg "Input entered was \"$input\""
```

Writing Input to the Active Console

When a synchronous policy is triggered, the related console is stored in the publish information specification. The policy director will query this information in an event_reqinfo call, and store the given console information for use by the **action puts** command.

The **action puts** command will write the string to the active console. A new line will be displayed unless the **nonewline** keyword is specified. The output from the **action puts** command for a synchronous applet is displayed directly to the console, bypassing the system logger. The output of the **action puts** command for an asynchronous applet is directed to the system logger.

Prerequisites

To use this feature, you must be running Cisco IOS Release 12.4(22)T, or a later release.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **event manager applet** *applet-name*
4. **event none**

5. **action** *label* **regexp** *string-pattern string-input* [*string-match* [*string-submatch1*] [*string-submatch2*] [*string-submatch3*]]
6. **action** *label* **puts** [**nonewline**] *string*
7. **exit**
8. **event manager run** *applet-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**event manager applet** *applet-name* **Example:** `Router(config)# event manager applet action`	Registers the applet with the EEM and enters applet configuration mode.
Step 4	**event none** **Example:** `Router(config-applet)# event none`	Specifies that an EEM policy is to be registered with the EEM and can be run manually.
Step 5	**action** *label* **regexp** *string-pattern string-input* [*string-match* [*string-submatch1*] [*string-submatch2*] [*string-submatch3*]] **Example:** `Router(config-applet)# action 1 regexp "(.*) (.*) (.*)" "one two three" _match _sub1`	Specifies the action to match the regular expression pattern on an input string when an EEM applet is triggered.
Step 6	**action** *label* **puts** [**nonewline**] *string* **Example:** `Router(config-applet)# action 2 puts "match is $_match"`	Specifies the action of printing data directly to the local console when an EEM applet is triggered. • The **nonewline** keyword is optional and is used to suppress the display of the new line character.
Step 7	**exit** **Example:** `Router(config-applet)# exit`	Exits applet configuration mode and returns to privileged EXEC mode.
Step 8	**event manager run** *applet-name* **Example:** `Router# event manager run action`	Manually runs a registered EEM policy. • In this example, the policy registered in Step 3 is triggered and the associated actions specified in Step 5 and Step 6 are executed.

Example

The following example shows how the **action puts** command prints data directly to the local console:

```
Router(config-applet)# event manager applet puts
Router(config-applet)# event none
Router(config-applet)# action 1 regexp "(.*) (.*) (.*)" "one two three" _match _sub1
Router(config-applet)# action 2 puts "match is $_match"
Router(config-applet)# action 3 puts "submatch 1 is $_sub1"
Router# event manager run puts
match is one two three
submatch 1 is one
```

Configuring SNMP Library Extensions

To configure SNMP Library Extensions for an EEM policy, you must be familiar with the following concepts:

- SNMP Get and Set Operations
- SNMP Traps and Inform Requests

Prerequisites

To use this feature, you must be running Cisco IOS Release 12.4(22)T or a later release.

SNMP Get and Set Operations

With the Cisco IOS Release 12.4(22)T, the SNMP Library Extensions feature extends the EEM applet **action info** and Tcl **sys_reqinfo_snmp** commands to include functionality for SNMP get-one, get-next, getid and set-any operations.

SNMP Get Operation

The SNMP event manager performs the SNMP get operation to retrieve one or more variables for the managed objects. Using the **action info type snmp oid get-type** and **action info type snmp getid** commands, you can configure the SNMP event manager to send an SNMP get request by specifying the variables to retrieve, and the IP address of the agent.

For example, if you want to retrieve the variable with the OID value of 1.3.6.1.2.1.1.1, you should specify the variable value, that is 1.3.6.1.2.1.1.1. If the specified values do not match, a trap will be generated and an error message will be written to the syslog history.

The **action info type snmp oid get-type** command specifies the type of the get operation to be performed. To retrieve the exact variable, the get operation type should be specified as **exact**. To retrieve a lexicographical successor of the specified OID value, the get operation type should be set to **next**.

Table 3 shows the built-in variables, in which the values retrieved from SNMP get operation are stored.

Table 3 Built-in Variables for action info type snmp oid Command

Built-in Variable	Description
_info_snmp_oid	The SNMP object ID.
_info_snmp_value	The value string of the associated SNMP data element.

GetID Operation

The **action info type snmp getid** command retrieves the following variables from the SNMP entity:

- sysDescr.0
- sysObjectID.0
- sysUpTime.0
- sysContact.0
- sysName.0
- sysLocation.0

Table 4 shows the built-in variables, in which the values retrieved from the SNMP getID operation are stored.

Table 4 Built-in Variables for action info type snmp getid Command

Built-in Variable	Description
_info_snmp_syslocation_oid	The OID value of the sysLocation variable.
_info_snmp_syslocation_value	The value string for the sysLocation variable.
_info_snmp_sysdescr_oid	The OID value of the sysDescr variable.
_info_snmp_sysdescr_value	The value string for the sysDescr variable.
_info_snmp_sysobjectid_oid	The OID value of the sysObjectID variable.
_info_snmp_sysobjectid_value	The value string for the sysObjectID variable.
_info_snmp_sysuptime_oid	The OID value of the sysUptime variable.
_info_snmp_sysuptime_value	The value string for the sysUptime variable.
_info_snmp_syscontact_oid	The OID value of the sysContact variable.
_info_snmp_syscontact_value	The value string for the sysContact variable.

The get operation requests can be sent to both local and remote hosts.

SNMP Set Operation

All SNMP variables are assigned a default value in the MIB view. The SNMP event manager can modify the value of these MIB variables through set operation. The set operation can be performed only on the system that allows read-write access.

To perform a set operation, you must specify the type of the variable and the value associated with it.

Table 5 shows the valid OID types and values for each OID type.

Table 5 OID Type and Value for Set Operation

OID Type	Description
counter32	A 32-bit number with a minimum value of 0. When the maximum value is reached, the counter resets to 0. Integer value in the range from 0 to 4294967295 is valid.
gauge	A 32-bit number with a minimum value of 0. For example, the interface speed on a router is measured using a gauge object type. Integer value in the range from 0 to 4294967295 is valid.
integer	A 32-bit number used to specify a numbered type within the context of a managed object. For example, to set the operational status of a router interface, 1 represents up and 2 represents down. Integer value in the range from 0 to 4294967295 is valid.
ipv4	IP version 4 address. IPv4 address in dotted decimal notation is valid.
octet string	An octet string in hexadecimal notation used to represent physical addresses. Text strings are valid.
string	An octet string in text notation used to represent text strings. Text strings are valid.
unsigned32	A 32-bit number used to represent decimal value.Unsigned integer value in the range from 0 to 4294967295 is valid.

The set operation can carried out on both local and remote hosts.

SNMP Traps and Inform Requests

Traps are SNMP notifications that alert the SNMP manager or the NMS to a network condition.

SNMP inform requests refer to the SNMP notifications that alert the SNMP manager to a network condition and request for confirmation of receipt from the SNMP manager.

An SNMP event occurs when SNMP MIB object ID values are sampled, or when the SNMP counter crosses a defined threshold. If the notifications are enabled and configured for such events, the SNMP traps or inform messages generated. An SNMP notification event is triggered when an SNMP trap or inform message is received by the event manager server.

To send an SNMP trap or inform message when an Embedded Event Manager (EEM) applet is triggered, the **action info type snmp trap** and **action info type snmp inform** commands are used. The CISCO-EMBEDDED-EVENT-MGR-MIB.my is used to define the trap and inform messages.

How to Configure SNMP Library Extensions

This section contains the following tasks:

Configuring EEM Applet for SNMP Get and Set Operations

While registering a policy with the event manager server, the actions associated with an SNMP event can be configured.

Perform this task to configure EEM applet for SNMP set and get operations.

Prerequisites

- You must be running Cisco IOS Release 12.4(22)T or a later release.
- SNMP event manager must be configured using the **snmp-server manager** command.
- The SNMP community string should be set by using the **snmp-server community** command to enable access to the SNMP entity.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **event manager applet** *applet-name*
4. **event snmp oid** *oid-value* **get-type** {**exact** | **next**} **entry-op** *operator* **entry-val** *entry-value* [**exit-comb** {**or** | **and**}] [**exit-op** *operator*] [**exit-val** *exit-value*] [**exit-time** *exit-time-value*] **poll-interval** *poll-int-value*
5. **action label info type snmp oid** *oid-value* **get-type** {**exact** | **next**} [**community** *community-string*] [**ipaddr** *ip-address*]
6. **action label info type snmp oid** *oid-value* **set-type** *oid-type oid-type-value* **community** *community-string* [**ipaddr** *ip-address*]
7. **action label info type snmp getid** *oid-value* [**community** *community-string*] [**ipaddr** *ip-address*]
8. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**event manager applet** *applet-name* **Example:** `Router(config)# event manager applet snmp`	Registers the applet with the event manager server and enters applet configuration mode.

	Command or Action	Purpose
Step 4	**event snmp oid** *oid-value* **get-type** {**exact** | **next**} **entry-op** *operator* **entry-val** *entry-value* [**exit-comb** {**or** | **and**}] [**exit-op** *operator*] [**exit-val** *exit-value*] [**exit-time** *exit-time-value*] **poll-interval** *poll-int-value* **Example:** `Router(config-applet)# event snmp oid 1.3.6.1.4.1.9.9.48.1.1.1.6.1 get-type exact entry-op lt entry-val 5120000 poll-interval 90`	Specifies the event criteria that cause the EEM applet to run. • In this example, an EEM event is triggered when free memory falls below the value of 5120000. • Exit criteria are optional, and if not specified, event monitoring is reenabled immediately.
Step 5	**action label info type snmp oid** *oid-value* **get-type** {**exact** | **next**} [**community** *community-string*][**ipaddr** *ip-address*] **Example:** `Router(config-applet)# action 1.3 info type snmp oid 1.3.6.1.4.1.9.9.48.1.1.1.6.1 get-type exact community public ipaddr 172.17.16.69`	Specifies the type of get operation to perform. • In this example, the type of get operation is specified as exact and community string is specified as public.
Step 6	**action label info type snmp oid** *oid-value* **set-type** *oid-type oid-type-value* **community** *community-string* [**ipaddr** *ip-address*] **Example:** `Router(config-applet)# action 1.4 info type snmp oid 1.3.6.1.4.1.9.9.48.1.1.1.6.1 set-type integer 42220 sysName.0 community rw ipaddr 172.17.16.69`	(Optional) Specifies the variable to be set. • In this example, the sysName.0 variable is specified for the set operation and community string is specified as rw. **Note** For set operation, you must specify the SNMP community string.
Step 7	**action label info type snmp getid** *oid-value* [**community** *community-string*] [**ipaddr** *ip-address*] **Example:** `Router(config-applet)# action 1.3 info type snmp getid community public ipaddr 172.17.16.69`	(Optional) Specifies if the individual variables should be retrieved by the getid operation.
Step 8	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode and enters global configuration mode.

Configuring EEM Applet for SNMP OID Notifications

Perform this task to configure SNMP notifications.

Prerequisites

- You must be running Cisco IOS Release 12.4(22)T or later release.
- SNMP event manager must be configured using the **snmp-server manager** command and SNMP agents must be configured to send and receive SNMP traps generated for an EEM policy.

- SNMP traps and informs must be enabled by using the **snmp-server enable traps event-manager** and **snmp-server enable traps** commands, to allow traps and inform requests to be sent from the Cisco IOS device to the event manager server.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **event manager applet** *applet-name*
4. **event snmp oid** *oid-value* **get-type** {**exact** | **next**} **entry-op** *operator* **entry-val** *entry-value* [**exit-comb** {**or** | **and**}] [**exit-op** *operator*] [**exit-val** *exit-value*] [**exit-time** *exit-time-value*] **poll-interval** *poll-int-value*
5. **action label info type snmp var** *variable-name* **oid** *oid-value oid-type oid-type-value*
6. **action label info type snmp trap enterprise-oid** *enterprise-oid-value* **generic-trapnum** *generic-trap-number* **specific-trapnum** *specific-trap-number* **trap-oid** *trap-oid-value* **trap-var** *trap-variable*
7. **action label info type snmp inform trap-oid** *trap-oid-value* **trap-var** *trap-variable* **community** *community-string* **ipaddr** *ip-address*
8. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**event manager applet** *applet-name* **Example:** `Router(config)# event manager applet snmp`	Registers the applet with the event manager server and enters applet configuration mode.
Step 4	**event snmp oid** *oid-value* **get-type** {**exact** | **next**} **entry-op** *operator* **entry-val** *entry-value* [**exit-comb** {**or** | **and**}] [**exit-op** *operator*] [**exit-val** *exit-value*] [**exit-time** *exit-time-value*] **poll-interval** *poll-int-value* **Example:** `Router(config-applet)# event snmp oid 1.3.6.1.4.1.9.9.48.1.1.1.6.1 get-type exact entry-op lt entry-val 5120000 poll-interval 90`	Specifies the event criteria that cause the EEM applet to run. • In this example, an EEM event is triggered when free memory falls below the value of 5120000. • Exit criteria are optional, and if not specified, event monitoring is reenabled immediately.
Step 5	**action label info type snmp var** *variable-name* **oid** *oid-value oid-type oid-type-value* **Example:** `Router(config-applet)# action 1.3 info type snmp var sysDescr.0 oid 1.3.6.1.4.1.9.9.48.1.1.1.6.1 integer 4220`	Specifies the instance of a managed object and its value. • In this example, the sysDescr.0 variable is used.
Step 6	**action label info type snmp trap enterprise-oid** *enterprise-oid-value* **generic-trapnum** *generic-trap-number* **specific-trapnum** *specific-trap-number* **trap-oid** *trap-oid-value* **trap-var** *trap-variable* **Example:** `Router(config-applet)# action 1.4 info type snmp trap enterprise-oid 1.3.6.1.4.1.1 generic-trapnum 4 specific-trapnum 7 trap-oid 1.3.6.1.4.1.1.226.0.2.1 trap-var sysUpTime.0`	Generates an SNMP trap when the EEM applet is triggered. • In this example, the authenticationFailure trap is generated. **Note** The specific trap number refers to the enterprise-specific trap, which is generated when an enterprise event occurs. If the generic trap number is not set to 6, the specific trap number you specify will be used to generate traps.

	Command or Action	Purpose
Step 7	**action label info type snmp inform trap-oid** *trap-oid-value* **trap-var** *trap-variable* **community** *community-string* **ipaddr** *ip-address* **Example:** `Router(config-applet)# action 1.4 info type snmp inform trap-oid 1.3.6.1.4.1.1.226.0.2.1 trap-var sysUpTime.0 community public ipaddr 172.69.16.2`	Generates an SNMP inform request when the EEM applet is triggered. • In this example, the inform request is generated for the sysUpTime.0 variable.
Step 8	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode and enters global configuration mode.

Configuring Variable Logic for EEM Applets

The Variable Logic for EEM Applets feature introduced in Cisco IOS Release 12.4(22)T and later releases, adds the ability to apply conditional logic within EEM applets. Before variable logic is introduced, applets have a linear structure where each action is executed in the order in which they are configured when the event is triggered. Conditional logic introduces a control structure that can change the flow of actions within applets depending on conditional expressions. Each control structure can contain a list of applet actions including looping and if/else actions which determine if the structure is executed or not.

The information in applet configuration mode is presented as background to set the context for the action commands.

To provide a consistent user interface between the Tool Command Language (Tcl) and the applet (CLI) based EEM policies, the following criteria are followed:

- Event specification criteria are written in Tcl in the Tcl based implementation.
- Event specification data is written using the CLI applet submode configuration statements in the applet-based implementation.

Applet configuration mode is entered using the event manager applet command. In applet configuration mode the config prompt changes to (config-applet)#. In applet configuration mode two types of config statements are supported:

- event - used to specify the event criteria to cause this applet to run.
- action - used to specify a built-in action to perform.

Multiple **action** applet config commands are allowed within an applet configuration. If no **action** applet config command is present, a warning is displayed, upon exit, stating no statements are associated with this applet. When no statements are associated with this applet, events get triggered but no action is taken. If no commands are specified in applet configuration mode, the applet will be removed upon exit. The exit applet config command is used to exit from applet configuration mode.

Prerequisites

To use this feature, you must be running Cisco IOS Release 12.4(22)T or a later release.

Configuring Variable Logic for EEM Applets

EEM 3.0 adds new applet action commands to permit simple variable logic within applets.

To configure the variable logic using action commands perform the followng tasks.

Specifying a Loop of Conditional Blocks

To specify a loop of a conditional block when an EEM applet is triggered, perform this task. In this task, a conditional loop is set to check if the value of the variable is less than 10. If the value of the variable is less than 10, then the message ‘i is $_i’ is written to the syslog.

Note Effective with Cisco IOS Release 12.4(22)T, the **set** (EEM) command is replaced by the **action set** command. See the **action** *label* **set** command for more information. If the set (EEM) command is entered in 12.4(22)T and later releases, the IOS parser translates the **set** command to the **action** *label* **set** command.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **event manager applet** *applet-name*
4. **action** *label* **set**
5. **action** *label* **while** *string_op1 operator string_op2*
6. Add any action as required.
7. **action** *label* **end**

DETAILED STEPS

	Command or Action	Purpose

Step 1	`enable`
	
	Example:
	`Router> enable`
		Enables privileged EXEC mode.
		• Enter your password if prompted.

Step 2	`configure terminal`
	
	Example:
	`Router# configure terminal`
		Enters global configuration mode.

	Command or Action	Purpose
Step 3	**event manager applet** *applet-name* **Example:** `Router(config)# event manager applet condition`	Registers the applet with the Embedded Event Manager (EEM) and enters applet configuration mode.
Step 4	**action** *label* **set** **Example:** `Router(config-applet)#action 1.0 set i 2`	Sets an action for the event. • In this example, the value of the variable i is set to 2.
Step 5	**action** *label* **while** *string_op1 operator string_op2* **Example:** `Router(config-applet)# action 2 while $i lt 10`	Specifies a loop of a conditional block. • In this example, a loop is set to check if the value of the variable i is less than 10.
Step 6	Add any action as required. **Example:** `Router(config-applet)# action 3 action syslog msg "i is $i"`	Performs the action as indicated by the action command. • In this example, the message 'i is $_i' is written to the syslog.
Step 7	**action** *label* **end** **Example:** `Router(config-applet)# action 3 end`	Exits from the running action.

Specifying if else Conditional Blocks

To specify the beginning of an if conditional statement followed by an else conditional statement, perform this task. The if or else conditional statements can be used in conjunction with each other or separately. In this task, the value of a variable is set to 5. An if conditional block is then specified, to check if the value of the variable is less than 10. Provided the if conditional block is satisfied, an action command to output the message 'x is less than 10' is specified.

Following the if conditional block, an else conditional block is specified. Provided the if conditional block is not satisfied, an action command to output the message 'x is greater than 10' is specified.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **event manager applet** *applet-name*
4. **action** *label* **set**
5. **action** *label* **if** [*stringop1*] {**eq** | **gt** | **ge** | **lt** | **le** | **ne**} [*stringop2*]
6. Add any action as required.
7. **action** *label* **else**
8. Add any action as required.

9. **action** *label* **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**event manager applet** *applet-name* **Example:** `Router(config)# event manager applet ifcondition`	Registers the applet with the Embedded Event Manager (EEM) and enters applet configuration mode.
Step 4	**action** *label* **set** **Example:** `Router(config-applet)#action 1.0 set x 5`	Sets an action for the event. • In this example, the value of the variable i is set to 5.
Step 5	**action** *label* **if** [*stringop1*] {**eq** | **gt** | **ge** | **lt** | **le** | **ne**} [*stringop2*] **Example:** `Router(config-applet)#action 2.0 if $x lt 10`	Specifies an if conditional statement. • In this example, an if conditional statement to check if the value of the variable is less than 10.
Step 6	Add any action as required. **Example:** `Router(config-applet)# action 3.0 puts "$x is less than 10"`	Performs the action as indicated by the action command. • In this example, the message '5 is less than 10' is displayed on the screen.
Step 7	**action** *label* **else** **Example:** `Router(config-applet)# action 4.0 else`	Specifies an else conditional satement
Step 8	Add any action as required. **Example:** `Router(config-applet)# action 5.0`	Performs the action as indicated by the action command. • In this example, the message '5 is greater than 10' is displayed on the screen.
Step 9	**end** **Example:** `Router(config-applet)# end`	Exits from the running action.

Specifying foreach Iterating Statements

To specify a conditional statement that iterates over an input string using the delimiter as a tokenizing pattern, perform this task. The foreach iteration statement is used to iterate through a collection to get the desired information. The delimiter is a regular expression pattern string. The token found in each iteration is assigned to the given iterator variable. All arithmetic calculations are performed as long integers with out any checks for overflow. In this task, the value of the variable x is set to 5. An iteration statement is set to run through the input string red, blue, green, orange. For every element in the input string, a corresponding message is displayed on the screen.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **event manager applet** *applet-name*
4. **action** *label* **foreach** [*string-iterator*] [*string-input*] [*string-delimiter*]
5. Specify any action command.
6. **action** *label* **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**event manager applet** *applet-name* **Example:** `Router(config)# event manager applet iteration`	Registers the applet with the Embedded Event Manager (EEM) and enters applet configuration mode.
Step 4	**action** *label* **foreach** [*string-iterator*] [*string-input*] [*string-delimiter*] **Example:** `Router(config-applet)#action 2.0 foreach iterator "red blue green orange"`	Iterates over an input string using the delimter as a tokenizing pattern. • In this example, the iteration is run through the elements of the input string - red, blue, green and orange.

	Command or Action	Purpose
Step 5	Specify any action command **Example:** `Router(config-applet)#action 3.0 puts "Iterator is $iterator"`	Performs the action as indicated by the action command. • In this example, the following message is displayed on the screen: Iterator is red Iterator is blue Iterator is green Iterator is orange
Step 6	**action** *label* **end** **Example:** `Router(config-applet)# action 4.0 end`	Exits from the running action.

Using Regular Expressions

To match a regular expression pattern with an input string, perform this task. Using regular expressions, you can specify the rules for a set of possible strings to be matched. In this task,

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **event manager applet** *applet-name*
4. **action** *label* **regexp** *string-pattern string-input* [*string-match* [*string-submatch1*] [*string-submatch2*] [*string-submatch3*]]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**event manager applet** *applet-name* **Example:** `Router(config)# event manager applet regexp`	Registers the applet with the Embedded Event Manager (EEM) and enters applet configuration mode.
Step 4	**action** *label* **regexp** *string-pattern string-input* [*string-match* [*string-submatch1*] [*string-submatch2*] [*string-submatch3*]] **Example:** `Router(config-applet)#action 2.0 regexp "(.*) (.*) (.*)" "red blue green" _match _sub1`	Specifies an expression pattern to match with an input string. • In this example, an input string of 'red blue green' is specified. When the expression pattern matches the input string the entire result **red blue green** is stored in the variable **_match** and the submatch **red** is stored in the variable **_sub1**.

Incrementing the Values of Variables

To increment the value of variables, perform this task. In this task, the value of a variable is set to 20 and then the value is incremented by 12.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **event manager applet** *applet-name*
4. **action** *label* **set**
5. **action** *label* **increment** *variable-name long-integer*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**event manager applet** *applet-name* **Example:** `Router(config)# event manager applet increment`	Registers the applet with the Embedded Event Manager (EEM) and enters applet configuration mode.

	Command or Action	Purpose
Step 4	**action** *label* **set** **Example:** `Router(config-applet)#action 1.0 set varname 20`	Sets an action for the event. • In this example, the value of the variable is set to 20.
Step 5	**action** *label* **increment** *variable-name long-integer* **Example:** `Router(config-applet)#action 2.0 increment varname 12`	Increments the value of variable by the specified long integer. • In this example, the value of the variable is incremented by 12.

Configuring Event SNMP Object

Perform this task to register the Simple Network Management Protocol (SNMP) object event for an Embedded Event Manager (EEM) applet that is run by sampling SNMP object.

Prerequisites

To use this feature, you must be running Cisco IOS Release 15.0(1)M or a later release.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **event manager applet** *applet-name*
4. **event snmp-object oid** *oid-value* **type** *value* **sync** {**yes** | **no**} **skip** {**yes** | **no**} **istable** {**yes** | **no**} [**default** *seconds*] [**maxrun** *maxruntime-number*]
5. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**event manager applet** *applet-name* **Example:** `Router(config)# event manager applet manual-policy`	Registers the applet with the Embedded Event Manager and enters applet configuration mode.

	Command or Action	Purpose

Step 4	**event snmp-object oid** *oid-value* **type** *value* **sync** {**yes** | **no**} **skip** {**yes** | **no**} **istable** {**yes** | **no**} [**default** *seconds*] [**maxrun** *maxruntime-number*]
	
	Example:
	`Router(config-applet)# event snmp-object oid 1.9.9.9.9 type gauge sync yes`
	`action 1 syslog msg "oid = $_snmp_oid"`
	`action 2 syslog msg "request = $_snmp_request"`
	`action 3 syslog msg "request_type = $_snmp_request_type"`
	
	Example:
	`Router(config-applet)# event manager applet snmp-obj1`
	`description "APPLET SNMP-OBJ-1"`
	`event snmp-object oid 1.3.6.1.2.1.2.2.1.2 istable yes type int sync no skip no default 0`
	`action 1 syslog msg "SNMP-OBJ1:TRIGGERED"`
	`facility "SNMP_OBJ"`
		Registers the Simple Network Management Protocol (SNMP) object event for an Embedded Event Manager (EEM) applet that is run by sampling SNMP object.
		
		The default for this command is that it is not configured. If this command is configured the defaults are the same as in the description of the syntax options,
		• The **oid** keyword specifies the SNMP object identifier (object ID).
		• The *oid-value* argument can be the Object ID value of the data element, in SNMP dotted notation. An OID is defined as a type in the associated MIB, CISCO-EMBEDDED-EVENT-MGR-MIB, and each type has an object value.
		• The **istable** keyword specifies whether the OID is an SNMP table.
		• The **sync** keyword specifies the SNMP and EEM policy execution.
		• The **type** keyword specifies the type of object.
		• The *value* argument is the value of the object.
		• The **skip** keyword specifies whether to skip CLI command execution.
		• The **default** keyword specifies the action to process the set or get request normally by the SNMP subsystem. If the **default** keyword is not specified, the default time period is set to 30 seconds.
		• The *milliseconds* argument is the time period during which the SNMP Object event detector waits for the policy to exit.
		• The **maxrun** keyword specifies the maximum runtime of the applet. If the **maxrun** keyword is specified, the *maxruntime-number* value must be specified. If the **maxrun** keyword is not specified, the default applet run time is 20 seconds.
		• The *milliseconds* argument is the maximum runtime of the apple in milliseconds. If the argument is not specified, the default 20-second run-time limit is used.

Step 5	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode and returns to privileged EXEC mode.

Disabling AAA Authorization

Perform this task to allow EEM policies to bypass AAA authorization when triggered.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **event manager applet** *applet-name* [**authorization bypass**] [**class** *class-options*] [**trap**]
4. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**event manager applet** *applet-name* [**authorization bypass**] [**class** *class-options*] [**trap**] **Example:** `Router(config-applet)# event manager applet one class A authorization bypass`	Registers the applet with the Embedded Event Manager (EEM) and enters applet configuration mode.
Step 4	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode and returns to privileged EXEC mode.

Configuring Description of an Embedded Event Manager Applet

Perform this task to describe an EEM applet. The description of an applet can be added in any order, before or after any other applet configuration. Configuring a new description for an applet that already has a description overwrites the current description. An applet description is optional.

Perform this task to configure a new description for an applet.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **event manager applet** *applet-name*
4. **description** *line*
5. **event syslog pattern** *regular-expression*
6. **action** *label* **syslog msg** *msg-text*
7. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**event manager applet** *applet-name* **Example:** `Router(config)# event manager applet increment`	Registers the applet with the EEM and enters applet configuration mode.
Step 4	**description** *line* **Example:** `Router(config-applet)# description "This applet looks for the word count in syslog messages"`	Adds or modifies the description of an EEM applet that is run by sampling Simple Network Management Protocol (SNMP).
Step 5	**event syslog pattern** *regular-expression* **Example:** `Router(config-applet)# event syslog pattern "count"`	Specifies the event criteria for an Embedded Event Manager (EEM) applet that is run by matching syslog messages.
Step 6	**action** *label* **syslog msg** *msg-text* **Example:** `Router(config-applet)# action 1 syslog msg hi`	Specifies the action to be taken when an EEM applet is triggered. • In this example, the action taken is to write a message to syslog. • The *msg-text* argument can be character text, an environment variable, or a combination of the two.
Step 7	**end** **Example:** `Router(config-applet)# end`	Exits applet configuration mode and returns to privileged EXEC mode.

Configuration Examples for Writing EEM Policies Using the Cisco IOS CLI

This section provides the following configuration examples:

Embedded Event Manager Applet Configuration: Examples

The following examples show how to create an EEM applet for some of the EEM event detectors. These examples follow steps outlined in the "Registering and Defining an Embedded Event Manager Applet" section on page 13.

Application-Specific Event Detector

The following example shows how a policy named EventPublish_A runs every 20 seconds and publishes an event type numbered 1 to an EEM subsystem numbered 798. The subsystem value of 798 specifies that a publish event has occurred from an EEM policy. A second policy named EventPublish_B is registered to run when the EEM event type 1 occurs with subsystem 798. When the EventPublish_B policy runs, it sends a message to syslog containing data passed as an argument from the EventPublish_A policy.

```
event manager applet EventPublish_A
 event timer watchdog time 20.0
 action 1.0 syslog msg "Applet EventPublish_A"
 action 2.0 publish-event sub-system 798 type 1 arg1 twenty
 exit
event manager applet EventPublish_B
 event application sub-system 798 type 1
 action 1.0 syslog msg "Applet EventPublish_B arg1 $_application_data1"
```

CLI Event Detector

The following example shows how to specify an EEM applet to run when the Cisco IOS **write memory** CLI command is run. The applet provides a notification that this event has occurred via a syslog message. In the example, the **sync** keyword is configured with the yes argument, and this means that the event detector is notified when this policy completes running. The exit status of the policy determines whether the CLI command will be executed. In this example, the policy exit status is set to one and the CLI command runs.

```
event manager applet cli-match
 event cli pattern "write mem.*" sync yes
 action 1.0 syslog msg "$_cli_msg Command Executed"
 set 2.0 _exit_status 1
```

The following example shows an applet which matches the **cli pattern** with the test argument. When **show access-list test** is entered, the CLI event detector matches the test argument, and the applet is triggered. The **debug event manager detector cli** output is added to show num_matches is set to one.

```
!
event manager applet EEM-PIPE-TEST
 event cli pattern "test" sync yes
 action 1.0 syslog msg "Pattern matched!"
!
*Aug 23 23:19:59.827: check_eem_cli_policy_handler: command_string=show access-lists test
*Aug 23 23:19:59.827: check_eem_cli_policy_handler: num_matches = 1, response_code = 4
*Aug 23 23:19:59.843: %HA_EM-6-LOG: EEM-PIPE-TEST: Pattern matched!
```

Note The functionality provided in the CLI event detector only allows a regular expression pattern match on a valid IOS CLI command itself. This does not include text after a pipe (|) character when redirection is used.

The following example shows that when **show version | include test** is entered, the applet fails to trigger because the CLI event detector does not match on characters entered after the pipe (|) character and the **debug event manager detector cli** output shows num_matches is set to zero.

```
*Aug 23 23:20:16.827: check_eem_cli_policy_handler: command_string=show version
*Aug 23 23:20:16.827: check_eem_cli_policy_handler: num_matches = 0, response_code = 1
```

Counter Event Detector and Timer Event Detector

The following example shows that the EventCounter_A policy is configured to run once a minute and to increment a well-known counter called critical_errors. A second policy—EventCounter_B—is registered to be triggered when the well-known counter called critical_errors exceeds a threshold of 3. When the EventCounter_B policy runs, it resets the counter to 0.

```
event manager applet EventCounter_A
 event timer watchdog time 60.0
 action 1.0 syslog msg "EventCounter_A"
 action 2.0 counter name critical_errors op inc value 1
 exit
event manager applet EventCounter_B
 event counter name critical_errors entry-op gt entry-val 3 exit-op lt exit-val 3
 action 1.0 syslog msg "EventCounter_B"
 action 2.0 counter name critical_errors op set value 0
```

Interface Counter Event Detector

The following example shows how a policy named EventInterface is triggered every time the receive_throttle counter for Fast Ethernet interface 0/0 is incremented by 5. The polling interval to check the counter is specified to run once every 90 seconds.

```
event manager applet EventInterface
 event interface name FastEthernet0/0 parameter receive_throttle entry-op ge entry-val 5
 entry-val-is-increment true poll-interval 90
 action 1.0 syslog msg "Applet EventInterface"
```

Resource Event Detector

The following example shows how to specify event criteria based on an ERM event report for a policy defined to report high CPU usage:

```
event manager applet policy-one
 event resource policy cpu-high
 action 1.0 syslog msg "CPU high at $_resource_current_value percent"
```

RF Event Detector

The RF event detector is only available on networking devices that contain dual Route Processors (RPs). The following example shows how to specify event criteria based on an RF state change notification:

```
event manager applet start-rf
 event rf event rf_prog_initialization
 action 1.0 syslog msg "rf state rf_prog_initialization reached"
```

RPC Event Detector

The RPC event detector allows an outside entity to make a Simple Object Access Protocol (SOAP) request to the router or a switch and invokes a defined EEM policy or script. The following example shows how an EEM applet called Event_RPC is being registered to run an EEM script:

```
event manager applet Event_RPC
 event rpc
 action print puts "hello there"
```

The following example shows the format of the SOAP request and reply message:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope xmlns:SOAP="http://www.cisco.com/eem.xsd">
  <SOAP:Body>
    <run_eemscript>
      <script_name>Event_RPC</script_name>
    </run_eemscript>
  </SOAP:Body>
</SOAP:Envelope>


]]>]]>

<?xml version="1.0" encoding="UTF-8"?><SOAP:Envelope
xmlns:SOAP="http://www.cisco.com/eem.xsd"><SOAP:Body><run_eemscript_response><return_code>
0</return_code><output></output></run_eemscript_response></SOAP:Body></SOAP:Envelope>]]>]]
>
```

SNMP Event Detector

The following example shows how to specify an EEM applet to run when the CPU usage is greater than 75 percent. When the EEM applet runs, the CLI commands **enable** and **show cpu processes** are run, and an e-mail containing the result of the **show cpu processes** command is sent to an engineer.

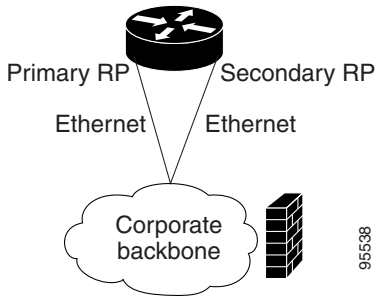
```
event manager applet snmpcpuge75
 event snmp oid 1.3.6.1.4.1.9.9.109.1.1.1.1.3.1 get-type exact entry-op ge entry-val 75
poll-interval 10
 action 1.0 cli command "enable"
 action 2.0 cli command "show process cpu"
 action 3.0 mail server "192.168.1.146" to "engineer@cisco.com" from "devtest@cisco.com"
subject "B25 PBX Alert" body "$_cli_result"
```

The next example is more complex and shows how to configure an EEM applet that causes a switch to the secondary (redundant) Route Processor (RP) when the primary RP runs low on memory.

This example illustrates a method for taking preventative action against a software fault that causes a memory leak. The action taken here is designed to reduce downtime by switching over to a redundant RP when a possible memory leak is detected.

Figure 1 shows a dual RP router that is running an EEM image. An EEM applet has been registered through the CLI using the **event manager applet** command. The applet will run when the available memory on the primary RP falls below the specified threshold of 5,120,000 bytes. The applet actions are to write a message to syslog that indicates the number of bytes of memory available and to switch to the secondary RP.

Figure 1 Dual RP Topology



The commands used to register the policy are shown below.

```
event manager applet memory-demo
 event snmp oid 1.3.6.1.4.1.9.9.48.1.1.1.6.1 get-type exact entry-op lt entry-val 5120000
poll-interval 90
 action 1.0 syslog priority critical msg "Memory exhausted; current available memory is
$_snmp_oid_val bytes"
 action 2.0 force-switchover
```

The registered applet is displayed using the **show event manager policy registered** command:

```
Router# show event manager policy registered

No.  Type    Event Type          Time Registered           Name
1    applet  snmp                Thu Jan30  05:57:16 2003  memory-demo
 oid {1.3.6.1.4.1.9.9.48.1.1.1.6.1} get-type exact entry-op lt entry-val {5120000}
poll-interval 90
 action 1.0 syslog priority critical msg "Memory exhausted; current available memory is
$_snmp_oid_val bytes"
 action 2.0 force-switchover
```

For the purpose of this example, a memory depletion is forced on the router, and a series of **show memory** commands are executed to watch the memory deplete:

```
Router# show memory

                Head    Total(b)     Used(b)     Free(b)   Lowest(b)  Largest(b)
Processor   53585260   212348444   119523060    92825384    92825384    92365916
Fast        53565260      131080       70360       60720       60720       60668

Router# show memory

                Head    Total(b)     Used(b)     Free(b)   Lowest(b)  Largest(b)
Processor   53585260   212364664   164509492    47855172    47855172    47169340
Fast        53565260      131080       70360       60720       60720       60668

Router# show memory

                Head    Total(b)     Used(b)     Free(b)   Lowest(b)  Largest(b)
Processor   53585260   212369492   179488300    32881192    32881192    32127556
Fast        53565260      131080       70360       60720       60720       60668
```

When the threshold is reached, an EEM event is triggered. The applet named memory-demo runs, causing a syslog message to be written to the console and a switch to be made to the secondary RP. The following messages are logged:

```
00:08:31: %HA_EM-2-LOG: memory-demo: Memory exhausted; current available memory is
4484196 bytes
00:08:31: %HA_EM-6-FMS_SWITCH_HARDWARE: fh_io_msg: Policy has requested a hardware
switchover
```

The following is partial output from the **show running-config** command on both the primary RP and the secondary (redundant) RP:

```
redundancy
 mode sso
.
.
!
event manager applet memory-demo
 event snmp oid 1.3.6.1.4.1.9.9.48.1.1.1.6.1 get-type exact entry-op lt entry-val
5120000 poll-interval 90
 action 1.0 syslog priority critical msg "Memory exhausted; current available memory
is $_snmp_oid_val bytes"
 action 2.0 force-switchover
```

SNMP Notification Event Detector

The following example shows how to configure the **snmp-server community** public RW and **snmp-server manager** commands before **event snmp-notification** is configured.

```
snmp-server community public RW
 snmp-server manager
```

The following example shows how an EEM applet called SNMP_Notification is being registered to run an EEM script when the router receives an SNMP notification on destination IP address 192.168.1.1 for object ID 1 whose value equals 10.

```
event manager applet SNMP_Notification
 event snmp-notification dest_ip_address 192.168.1.1 oid 1 op eq oid-value 10
 action 1 policy eem_script
```

Syslog Event Detector

The following example shows how to specify an EEM applet to run when syslog identifies that Ethernet interface 1/0 is down. The applet sends a message about the interface to syslog.

```
event manager applet interface-down
 event syslog pattern ".*UPDOWN.*Ethernet1/0.*" occurs 4
 action 1.0 syslog msg "Ethernet interface 1/0 changed state 4 times"
```

Embedded Event Manager Manual Policy Execution: Examples

The following examples show how to use the none event detector to configure an EEM policy (applet or script) to be run manually.

Using the event manager run Command

This example shows how to run a policy manually using the **event manager run** command. The policy is registered using the **event none** command under applet configuration mode and then run from global configuration mode using the **event manager run** command.

```
event manager applet manual-policy
 event none
 action 1.0 syslog msg "Manual-policy triggered"
 end
!
event manager run manual-policy
```

Using the action policy Command

This example shows how to run a policy manually using the **action policy** command. The policy is registered using the **event none** command under applet configuration mode, and then the policy is executed using the **action policy** command in applet configuration mode.

```
event manager applet manual-policy
 event none
 action 1.0 syslog msg "Manual-policy triggered"
 exit
!
event manager applet manual-policy-two
 event none
 action 1.0 policy manual-policy
 end
!
event manager run manual-policy-two
```

Configuring and Tracking a Stub Object Using Embedded Event Manager: Example

This example shows how to create a stub object, set the state of the stub object, and configure an EEM applet to be run when the tracked object changes. The enhanced object tracking (EOT) event detector is used, and actions are specified to both set and read the state of the object. This example allows EEM to define an EOT object that may be manipulated by other EOT clients. An EEM policy can be a trigger for any EOT object including objects defined for other EOT clients or for an object defined by EEM.

```
track 10 stub-object
 default-state down
!
event manager applet track-ten
 event track 10 state any
 action 1.0 track set 10 state up
 action 2.0 track read 10
```

Embedded Event Manager Watchdog System Monitor (Cisco IOS) Event Detector Configuration: Example

The following example shows how to configure three EEM applets to demonstrate how the Cisco IOS watchdog system monitor (IOSWDSysMon) event detector works.

Watchdog System Monitor Sample1 Policy

The first policy triggers an applet when the average CPU usage for the process named IP Input is greater than or equal to 1 percent for 10 seconds:

```
event manager applet IOSWD_Sample1
 event ioswdsysmon sub1 cpu-proc taskname "IP Input" op ge val 1 period 10
 action 1.0 syslog msg "IOSWD_Sample1 Policy Triggered"
```

Watchdog System Monitor Sample2 Policy

The second policy triggers an applet when the total amount of memory used by the process named Net Input is greater than 100 kb:

```
event manager applet IOSWD_Sample2
 event ioswdsysmon sub1 mem-proc taskname "Net Input" op gt val 100 is-percent false
 action 1.0 syslog msg "IOSWD_Sample2 Policy Triggered"
```

Watchdog System Monitor Sample3 Policy

The third policy triggers an applet when the total amount of memory used by the process named IP RIB Update has increased by more than 50 percent over the sample period of 60 seconds:

```
event manager applet IOSWD_Sample3
 event ioswdsysmon sub1 mem-proc taskname "IP RIB Update" op gt val 50 is-percent true
period 60
 action 1.0 syslog msg "IOSWD_Sample3 Policy Triggered"
```

The three policies are configured, and then repetitive large pings are made to the networking device from several workstations, causing the networking device to register some usage. This will trigger policies 1 and 2, and the console will display the following messages:

```
00:42:23: %HA_EM-6-LOG: IOSWD_Sample1: IOSWD_Sample1 Policy Triggered
00:42:47: %HA_EM-6-LOG: IOSWD_Sample2: IOSWD_Sample2 Policy Triggered
```

To view the policies that are registered, use the **show event manager policy registered** command:

```
Router# show event manager policy registered

No.  Class   Type    Event Type          Trap  Time Registered           Name
1    applet  system  ioswdsysmon         Off   Fri Jul 23 02:27:28 2004  IOSWD_Sample1
 sub1 cpu_util {taskname {IP Input} op ge val 1 period 10.000 }
 action 1.0 syslog msg "IOSWD_Sample1 Policy Triggered"

2    applet  system  ioswdsysmon         Off   Fri Jul 23 02:23:52 2004  IOSWD_Sample2
 sub1 mem_used {taskname {Net Input} op gt val 100 is_percent FALSE}
 action 1.0 syslog msg "IOSWD_Sample2 Policy Triggered"

3    applet  system  ioswdsysmon         Off   Fri Jul 23 03:07:38 2004  IOSWD_Sample3
 sub1 mem_used {taskname {IP RIB Update} op gt val 50 is_percent TRUE period 60.000 }
 action 1.0 syslog msg "IOSWD_Sample3 Policy Triggered"
```

Configuration SNMP Library Extensions: Examples

This section provides the following configuration examples:

- SNMP Get Operations: Examples
- SNMP GetID Operations: Examples
- Set Operations: Examples
- Generating SNMP Notifications: Examples

SNMP Get Operations: Examples

The following example shows how to send a get request to the local host.

```
Router(config)# event manager applet snmp

Router(config-applet)# event snmp oid 1.3.6.1.2.1.1.1.0 get-type exact entry-op lt
entry-val 5120000 poll-interval 90

Router(config-applet)# action 1.3 info type snmp oid 1.3.6.1.2.1.1.1.0 get-type exact
community public

Router(config-applet)# action 1.3 info type snmp oid 1.3.6.1.2.1.1.4.0 get-type next
community public
```

The following log message will be written to the SNMP event manager log:

```
1d03h:%HA_EM-6-LOG: lg: 1.3.6.1.2.1.1.1.0
1d04h:%HA_EM-6-LOG: lgn: 1.3.6.1.2.1.1.5.0
```

The following example shows how to send a get request to a remote host.

```
Router(config)# event manager applet snmp

Router(config-applet)# event snmp oid 1.3.6.1.2.1.1.1.0 get-type exact entry-op lt
entry-val 5120000 poll-interval 90

Router(config-applet)# action 1.3 info type snmp oid 1.3.6.1.2.1.1.4.0 get-type next
community public ipaddr 172.17.16.69

Router(config-applet)# action 1.3 info type snmp getid 1.3.6.1.2.1.1.1.0 community public
ipaddr 172.17.16.69
```

The following log message is written to the SNMP event manager log:

```
1d03h:%HA_EM-6-LOG: lg: 1.3.6.1.2.1.1.1.0
1d04h:%HA_EM-6-LOG: lgn: 1.3.6.1.2.1.1.5.0
```

SNMP GetID Operations: Examples

The following example shows how to send a getid request to the local host.

```
Router(config)# event manager applet snmp

Router(config-applet)# event snmp oid 1.3.6.1.2.1.1.1.0 get-type exact entry-op lt
entry-val 5120000 poll-interval 90

Router(config-applet)# action 1.3 info type snmp getid community public
```

The following log message is written to the SNMP event manager log:

```
1d04h:%HA_EM-6-LOG: lgid: _info_snmp_sysname_oid=1.3.6.1.2.1.1.5.0
1d04h:%HA_EM-6-LOG: lgid: _info_snmp_sysname_value=jubjub.cisco.com
1d04h:%HA_EM-6-LOG: lgid: _info_snmp_syslocation_oid=1.3.6.1.2.1.1.6.0
1d04h:%HA_EM-6-LOG: lgid: _info_snmp_syslocation_value=
1d04h:%HA_EM-6-LOG: lgid: _info_snmp_sysdescr_oid=1.3.6.1.2.1.1.1.0
1d04h:%HA_EM-6-LOG: lgid: _info_snmp_sysobjectid_oid=1.3.6.1.2.1.1.2.0
1d04h:%HA_EM-6-LOG: lgid: _info_snmp_sysobjectid_value=products.222
1d04h:%HA_EM-6-LOG: lgid: _info_snmp_sysuptime_oid=1.3.6.1.2.1.1.3.0
1d04h:%HA_EM-6-LOG: lgid: _info_snmp_sysuptime_oid=10131676
1d04h:%HA_EM-6-LOG: lgid: _info_snmp_syscontact_oid=1.3.6.1.2.1.1.4.0
1d04h:%HA_EM-6-LOG: lgid: _info_snmp_syscontact_value=YYY
```

The following example shows how to send a getid request to a remote host.

```
Router(config)# event manager applet snmp

Router(config-applet)# event snmp oid 1.3.6.1.2.1.1.1.0 get-type exact entry-op lt
entry-val 5120000 poll-interval 90

Router(config-applet)# action 1.3 info type snmp getid 1.3.6.1.2.1.1.1.0 community public
ipaddr 172.17.16.69
```

The following log message is written to the SNMP event manager log:

```
1d04h:%HA_EM-6-LOG: lgid: _info_snmp_sysname_oid=1.3.6.1.2.1.1.5.0
1d04h:%HA_EM-6-LOG: lgid: _info_snmp_sysname_value=jubjub.cisco.com
1d04h:%HA_EM-6-LOG: lgid: _info_snmp_syslocation_oid=1.3.6.1.2.1.1.6.0
1d04h:%HA_EM-6-LOG: lgid: _info_snmp_syslocation_value=
1d04h:%HA_EM-6-LOG: lgid: _info_snmp_sysdescr_oid=1.3.6.1.2.1.1.1.0
```

```
1d04h:%HA_EM-6-LOG: lgid: _info_snmp_sysobjectid_oid=1.3.6.1.2.1.1.2.0
1d04h:%HA_EM-6-LOG: lgid: _info_snmp_sysobjectid_value=products.222
1d04h:%HA_EM-6-LOG: lgid: _info_snmp_sysuptime_oid=1.3.6.1.2.1.1.3.0
1d04h:%HA_EM-6-LOG: lgid: _info_snmp_sysuptime_oid=10131676
1d04h:%HA_EM-6-LOG: lgid: _info_snmp_syscontact_oid=1.3.6.1.2.1.1.4.0
1d04h:%HA_EM-6-LOG: lgid: _info_snmp_syscontact_value=YYY
```

Set Operations: Examples

The following example shows how to perform a set operation on the local host.

```
Router(config)# event manager applet snmp

Router(config-applet)# event snmp oid 1.3.6.1.2.1.1.1.0 get-type exact entry-op lt
entry-val 5120000 poll-interval 90

Router(config-applet)# action 1.3 info type snmp oid 1.3.6.1.2.1.1.4.0 set-type integer 5
sysName.0 community public
```

The following log message is written to the SNMP event manager log:

```
1d04h:%HA_EM-6-LOG: lset: 1.3.6.1.2.1.1.4.0
1d04h:%HA_EM-6-LOG: lset: XXX
```

The following example shows how to perform a set operation on a remote host.

```
Router(config)# event manager applet snmp

Router(config-applet)# event snmp oid 1.3.6.1.2.1.1.1.0 get-type exact entry-op lt
entry-val 5120000 poll-interval 90

Router(config-applet)# action 1.3 info type snmp oid 1.3.6.1.2.1.1.4.0 set-type integer 5
sysName.0 community public ipaddr 172.17.16.69
```

The following log message is written to the SNMP event manager log:

```
1d04h:%HA_EM-6-LOG: lset: 1.3.6.1.2.1.1.4.0
1d04h:%HA_EM-6-LOG: lset: XXX
```

Generating SNMP Notifications: Examples

The following example shows how to configure SNMP traps for the sysUpTime.0 variable:

```
Router(config)# event manager applet snmp

Router(config-applet)# event snmp oid 1.3.6.1.4.1.9.9.48.1.1.1.6.1 get-type exact entry-op
lt entry-val 5120000 poll-interval 90

Router(config-applet)# action 1.3 info type snmp var sysUpTime.0 oid
1.3.6.1.4.1.9.9.43.1.1.6.1.3.41 integer 2

Router(config-applet)# action 1.4 info type snmp trap enterprise-oid ciscoSyslogMIB.2
generic-trapnum 6 specific-trapnum 1 trap-oid 1.3.6.1.4.1.9.9.41.2.0.1 trap-var
sysUpTime.0
```

The following output is generated if the debug snmp packets command is enabled:

```
Router# debug snmp packets

1d04h: SNMP: Queuing packet to 172.69.16.2
1d04h: SNMP: V1 Trap, ent ciscoSyslogMIB.2, addr 172.19.rap 1
```

```
clogHistoryEntry.3 = 4
clogHistoryEntry.6 = 9999
1d04h: SNMP: Queuing packet to 172.19.208.130
1d04h: SNMP: V1 Trap, ent ciscoSyslogMIB.2, addr 172.19.rap 1
clogHistoryEntry.3 = 4
clogHistoryEntry.6 = 9999
1d04h: SNMP: Packet sent via UDP to 172.69.16.2
1d04h: SNMP: Packet sent via UDP to 172.69.16.2
infra-view10:
Packet Dump:
30 53 02 01 00 04 04 63 6f 6d 6d a4 48 06 09 2b
06 01 04 01 09 09 29 02 40 04 ac 13 d1 17 02 01
06 02 01 01 43 04 00 9b 82 5d 30 29 30 12 06 0d
2b 06 01 04 01 09 09 29 01 02 03 01 03 02 01 04
30 13 06 0d 2b 06 01 04 01 09 09 29 01 02 03 01
06 02 02 27 0f
Received SNMPv1 Trap:
Community: comm
Enterprise: ciscoSyslogMIBNotificationPrefix
Agent-addr: 172.19.209.23
Enterprise Specific trap.
Enterprise Specific trap: 1
Time Ticks: 10191453
clogHistSeverity = error(4)
clogHistTimestamp = 9999
```

The following example shows how to configure SNMP inform requests for the sysUpTime.0 variable:

```
Router(config)# event manager applet snmp

Router(config-applet)# event snmp oid 1.3.6.1.4.1.9.9.48.1.1.1.6.1 get-type exact entry-op
lt entry-val 5120000 poll-interval 90

Router(config-applet)# action 1.3 info type snmp var sysUpTime.0 oid
1.3.6.1.4.1.9.9.43.1.1.6.1.3.41 integer 2

Router(config-applet)# action 1.4 info type snmp inform trap-oid 1.3.6.1.4.1.9.9.43.2.0.1
trap-var sysUpTime.0 community public ipaddr 172.19.209.24
```

The following output is generated if the debug snmp packets command is enabled:

```
Router# debug snmp packets

1d04h: SNMP: Inform request, reqid 24, errstat 0, erridx 0
sysUpTime.0 = 10244391
snmpTrapOID.0 = ciscoConfigManMIB.2.0.1
ccmHistoryEventEntry.3.40 = 1
1d04h: SNMP: Packet sent via UDP to 172.19.209.24.162
1d04h: SNMP: Packet received via UDP from 172.19.209.24 on FastEthernet0/0
1d04h: SNMP: Response, reqid 24, errstat 0, erridx 0
1d04h: SNMP: Response, reqid 24, errstat 0, erridx 0
1d04h: SNMP: Inform request, reqid 25, errstat 0, erridx 0
sysUpTime.0 = 10244396
snmpTrapOID.0 = ciscoConfigManMIB.2.0.1
ccmHistoryEventEntry.3.41 = 2
1d04h: SNMP: Packet sent via UDP to 172.19.209.24.162
1d04h: SNMP: Packet received via UDP from 172.19.209.24 on FastEthernet0/0
1d04h: SNMP: Response, reqid 25, errstat 0, erridx 0
1d04h: SNMP: Response, reqid 25, errstat 0, erridx 0
Router# debug snmp packets
5d04h: SNMP: Packet received via UDP from 172.19.209.23 on FastEthernet0/0
5d04h: SNMP: Inform request, reqid 24, errstat 0, erridx 0
sysUpTime.0 = 10244391
snmpTrapOID.0 = ciscoConfigManMIB.2.0.1
```

```
ccmHistoryEventEntry.3.40 = 1
5d04h: dest if_index = 1
5d04h: dest ip addr= 172.19.209.24
5d04h: SNMP: Response, reqid 24, errstat 0, erridx 0
5d04h: SNMP: Packet sent via UDP to 172.19.209.23.57748
5d04h: SNMP: Packet received via UDP from 172.19.209.23 on FastEthernet0/0
5d04h: SNMP: Inform request, reqid 25, errstat 0, erridx 0
```

Configuring Variable Logic for EEM Applets: Examples

The following sections provide examples on some selected action commands. For information on all the action commands supporting variable logic within applets, see Table 6.

In this example, conditional loops **while**, **if** and **foreach** are used to print data. Other action commands such as **action divide**, **action increment** and **action puts** are used to define the actions to be performed when the conditions are met.

```
event manager applet printdata
event none
action 100 set colors "red green blue"
action 101 set shapes "square triangle rectange"
action 102 set i "1"
action 103 while $i lt 6
action 104  divide $i 2
action 105  if $_remainder eq 1
action 106   foreach _iterator "$colors"
action 107    puts nonewline "$_iterator "
action 108   end
action 109   puts ""
action 110  else
action 111   foreach _iterator "$shapes"
action 112    puts nonewline "$_iterator "
action 113   end
action 114   puts ""
action 115  end
action 116  increment i
action 117 end
```

When the event manager applet ex is run, the following output is obtained:

```
event manager run printdata
red green blue
square triangle rectange
red green blue
square triangle rectange
red green blue
```

In this example, two environment variables poll_interface and max_rx_rate are set to F0/0 and 3 respectively. Every 30 seconds there is a poll on an interface for rx rate. If the rx rate is greater than the threshold, a syslog message is displayed.

This applet makes use of the foreach conditional statement to poll the interface, the if conditional block to compare the value under RXPS with max_rx_rate that was set in the EEM environment variable.

```
event manager environment poll_interfaces F0/0
event manager environment max_rx_rate 3

ev man app check_rx_rate
ev timer watchdog name rx_timer time 30
action 100 foreach int $poll_interfaces
action 101  cli command "en"
```

```
action 102  cli command "show int $int summ | beg ------"
action 103  foreach line $_cli_result "\n"
action 105   regexp ".*[0-9]+\s+[0-9]+\s+[0-9]+\s+[0-9]+\s+[0-9]+\s+([0-9]+)\s+.*" $line
junk rxps
action 106   if $_regexp_result eq 1
action 107    if $rxps gt $max_rx_rate
action 108     syslog msg "Warning rx rate for $int is > than threshold. Current value is
$rxps  (threshold is $max_rx_rate)"
action 109    end
action 110   end
action 111  end
action 112 end

Example syslog message:

Oct 16 09:29:26.153: %HA_EM-6-LOG: c: Warning rx rate for F0/0 is > than threshold.
Current value is 4  (threshold is 3)

The output of show int F0/0 summ is of the format:

#show int f0/0 summ

 *: interface is up
 IHQ: pkts in input hold queue     IQD: pkts dropped from input queue
 OHQ: pkts in output hold queue    OQD: pkts dropped from output queue
 RXBS: rx rate (bits/sec)          RXPS: rx rate (pkts/sec)
 TXBS: tx rate (bits/sec)          TXPS: tx rate (pkts/sec)
 TRTL: throttle count

  Interface               IHQ   IQD  OHQ   OQD  RXBS RXPS  TXBS TXPS TRTL
------------------------------------------------------------------------
* FastEthernet0/0          0 87283    0     0     0    0     0    0    0
```

Note To use other action commands supporting variable logic within applets, use the commands listed in Table 6.

Table 6 Available action commands

Action Commands	Purpose
action add	Adds the value of two variables when an EEM applet is triggered.
action append	Appends the given value to the current value of a variable when an EEM applet is triggered.
action break	Causes an immediate exit from a loop of actions when an EEM applet is triggered
action comment	Adds comments to an applet when an EEM applet is triggered
action context retrieve	Retrieves variables identified by a given set of context name keys when an EEM applet is triggered.
action context save	Saves information across multiple policy triggers when an EEM applet is triggered.
action continue	Continues with a loop of actions when an EEM applet is triggered.
action decrement	Decrements the value of a variable when an EEM applet is triggered.
action divide	Divides the dividend value by the given divisor value when an EEM applet is triggered.

Table 6 Available action commands (continued)

Action Commands	Purpose
action else	Specifies the beginning of else conditional action block in if / else conditional action block when an EEM applet is triggered.
action elseif	Identifies the beginning of the else conditional action block in the else / if conditional action block when an EEM applet is triggered.
action end	Specifies the identification of the end of an conditional action block in the if / else and while conditional action block when an EEM applet is triggered.
action exit	Specifies an immediate exit from the running applet configuration when an EEM applet is triggered.
action foreach	Specifies the iteration of an input string using the delimiter as a tokenizing pattern, when an EEM applet is triggered.
action gets	Gets an input from the local TTY in a synchronous applet and store the value in the given variable when an EEM applet is triggered.
action if	Specifies the identification of the beginning of an if conditional block when an EEM applet is triggered.
action if goto	Instructs the applet to jump to a given label if the specified condition is true when an EEM applet is triggered.
action increment	Increments the value of a variable when an EEM applet is triggered.
action info type interface-names	Specifies the action of obtaining interface names when an EEM applet is triggered.
action info type snmp getid	Retrieves the individual variables from a Simple Network Management Protocol (SNMP) entity during the SNMP get operation.
action info type snmp inform	Sends an SNMP inform requests when an EEM applet is triggered.
action info type snmp oid	Specifies the type of SNMP get operation and the object to retrieve during the SNMP set operation, when an EEM applet is triggered.
action info type snmp trap	Sends SNMP trap requests when an EEM applet is triggered.
action info type snmp var	Creates a variable for an SNMP object identifier (OID) and its value from an EEM applet
action multiply	Specifies the action of multiplying the variable value with a specified given integer value when an EEM applet is triggered.
action puts	Enables the action of printing data directly to the local tty when an EEM applet is triggered.
action regexp	Specifies the action of matching a regular expression pattern on an input string when an EEM applet is triggered.
action set (EEM)	Specifies the action of setting the value of a variable when an EEM applet is triggered.
action string compare	Specifies the action of comparing two unequal strings when an EEM applet is triggered.
action string equal	Specifies the action of verifying whether or not two strings are equal when an EEM applet is triggered

Table 6 Available action commands (continued)

Action Commands	Purpose
action string first	Specifies the action of returning the index on the first occurrence of string1 within string2 when an EEM applet is triggered.
action string index	Specifies the action of returning the characters specified at a given index value when an EEM applet is triggered.
action string last	Specifies the action of returning the index on the last occurrence of string1 within string 2 when an EEM applet is triggered.
action string length	Specifies the action of returning the number of characters in a string when the EEM applet is triggered.
action string match	Specifies the action of returning 1 to the $_string_result, if the string matches the pattern when an EEM applet is triggered.
action string range	Specifies the action of storing a range of characters in a string when an EEM applet is triggered.
action string replace	Specifies the action of storing a new string by replacing range of characters in the specified string when an EEM applet is triggered.
action string tolower	Specifies the action of storing specific range of characters of a string in lowercase when an EEM applet is triggered.
action string toupper	Specifies the action of storing specific range of characters of a string in uppercase when an EEM applet is triggered.
action string trim	Specifies the action to trim a string when an EEM applet is triggered.
action string trimleft	Specifies the action to trim the characters of one string from the left end of another string when an EEM applet is triggered.
action string trimright	Specifies the action to trim the characters one string from the right end of another string when an EEM applet is triggered.
action subtract	Subtracts the value of a variable from another value when an EEM applet is triggered.
action while	Specifies the action of identifying the beginning of a loop of conditional block when an EEM applet is triggered.

Configuring Event SNMP-Object: Examples

The following example shows the SET operation and the value to set is in $_snmp_value and it is managed by the script. The example below saves the oid and its value as contexts to be retrieved later.

```
event manager applet snmp-object1
 description "APPLET SNMP-OBJ-1"
 event snmp-object oid 1.3.6.1.2.1.31.1.1.1.18 type string sync no skip no istable yes
default 0  action 1 syslog msg "SNMP-OBJ1:TRIGGERED" facility "SNMP_OBJ"
 action 2 context save key myoid variable "_snmp_oid"
 action 3 context save key myvalue variable "_snmp_value"
```

Configuring Description of an EEM Applet: Examples

The following example shows how to add or modify the description for an Embedded Event Manager (EEM) applet that is run by sampling Simple Network Management Protocol (SNMP):

```
event manager applet test
 description "This applet looks for the word count in syslog messages"
 event syslog pattern "count"
 action 1 syslog msg hi
```

Where to Go Next

- For information about EEM overview, go to "Embedded Event Manager Overview" module.
- For information about writing EEM policies using Tcl, go to "Writing Embedded Event Manager Policies Using Tcl" module.

Additional References

The following sections provide references related to writing EEM policies Using the Cisco IOS CLI.

Related Documents

Related Topic	Document Title
Cisco IOS commands	*Cisco IOS Master Commands List, All Releases*
Network Management commands (including EEM commands): complete command syntax, defaults, command mode, command history, usage guidelines, and examples	*Cisco IOS Network Management Command Reference*
Embedded Event Manager overview	Embedded Event Manager Overview module
Embedded Event Manager policy writing using Tcl	Writing Embedded Event Manager Policies Using Tcl module
Configuring enhanced object tracking	Configuring Enhanced Object Tracking module

Standards

Standard	Title
No new or modified standards are supported, and support for existing standards has not been modified.	—

MIBs

MIB	MIBs Link
CISCO-EMBEDDED-EVENT-MGR-MIB	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

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

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for Writing EEM Policies Using the Cisco IOS CLI

Table 7 lists the features in this module and provides links to specific configuration information. Only features that were introduced or modified in Cisco IOS Releases 12.3(14)T, 12.2(25)S, 12.0(26)S, 12.2(18)SXF4, 12.2(28)SB, 12.2(33)SRA, 12.2(33)SXH, 12.2(33)SXI, 12.4(20)T, 12.4(22)T, 15.0(1)M, 12.2(33)SRE or a later release appear in the table.

Not all commands may be available in your Cisco IOS software release. For release information about a specific command, see the command reference documentation.

Use Cisco Feature Navigator to find information about platform support and software image support. Cisco Feature Navigator enables you to determine which Cisco IOS and Catalyst OS software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Note Table 7 lists only the Cisco IOS software release that introduced support for a given feature in a given Cisco IOS software release train. Unless noted otherwise, subsequent releases of that Cisco IOS software release train also support that feature.

Table 7 Feature Information for Writing EEM Policies Using the Cisco IOS CLI

Feature Name	Releases	Feature Information
Embedded Event Manager 1.0	12.0(26)S 12.3(4)T	EEM 1.0 introduced Embedded Event Manager applet creation with the SNMP and syslog event detectors. EEM 1.0 also introduced the following actions: generating prioritized syslog messages, generating a CNS event for upstream processing by Cisco CNS devices, reloading the Cisco IOS software, and switching to a secondary processor in a fully redundant hardware configuration. The following sections provide information about this feature: • Embedded Event Manager Policies, page 2 • Embedded Event Manager Built-In Environment Variables Used in EEM Applets, page 3 • Registering and Defining an Embedded Event Manager Applet, page 13 • Displaying Embedded Event Manager Registered Policies, page 25 The following commands were introduced by this feature: **action cns-event**, **action force-switchover**, **action reload**, **action syslog**, **debug event manager**, **event manager applet**, **event snmp**, **event syslog**, **show event manager policy registered**.

Table 7 Feature Information for Writing EEM Policies Using the Cisco IOS CLI (continued)

Feature Name	Releases	Feature Information
Embedded Event Manager 2.0	12.2(25)S	EEM 2.0 introduced the application-specific event detector, the counter event detector, the interface counter event detector, the timer event detector, and the watchdog event detector. New actions included modifying a named counter, publishing an application-specific event, and generating an SNMP trap. The ability to define environment variables and to run EEM policies written using Tcl was introduced, and two sample policies were included with the software. The following sections provide information about this feature: • Embedded Event Manager Policies, page 2 • Embedded Event Manager Built-In Environment Variables Used in EEM Applets, page 3 • Registering and Defining an Embedded Event Manager Policy to Run Manually, page 17 • Unregistering Embedded Event Manager Policies, page 19 • Suspending All Embedded Event Manager Policy Execution, page 20 • Displaying Embedded Event Manager History Data, page 24 • Embedded Event Manager Applet Configuration: Examples, page 60 • Embedded Event Manager Watchdog System Monitor (Cisco IOS) Event Detector Configuration: Example, page 65 The following commands were introduced by this feature: **action counter**, **action publish-event**, **action snmp-trap**, **event application**, **event counter**, **event interface**, **event ioswdsysmon**, **event manager environment**, **event manager history size**, **event manager policy**, **event manager scheduler suspend**, **event timer**, **show event manager environment**, **show event manager history events**, **show event manager history traps**, **show event manager policy available**, **show event manager policy pending**.

Table 7 Feature Information for Writing EEM Policies Using the Cisco IOS CLI (continued)

Feature Name	Releases	Feature Information
Embedded Event Manager 2.1	12.3(14)T 12.2(18)SXF5 12.2(28)SB 12.2(33)SRA	EEM 2.1 introduced some new event detectors and actions with new functionality to allow EEM policies to be run manually and the ability to run multiple concurrent policies. Support for Simple Network Management Protocol (SNMP) event detector rate-based events was provided as was the ability to create policies using Tool Command Language (Tcl). The following sections provide information about this feature: • Embedded Event Manager Policies, page 2 • Embedded Event Manager Built-In Environment Variables Used in EEM Applets, page 3 • Registering and Defining an Embedded Event Manager Policy to Run Manually, page 17 • Embedded Event Manager Applet Configuration: Examples, page 60 The following commands were introduced or modified by this feature: **action cli**, **action counter**, **action info**, **action mail**, **action policy**, **debug event manager**, **event cli**, **event manager directory user**, **event manager policy**, **event manager run**, **event manager scheduler script**, **event manager session cli username**, **event none**, **event oir**, **event snmp**, **event syslog**, **set** (EEM), **show event manager directory user**, **show event manager policy registered**, **show event manager session cli username**.

Table 7 Feature Information for Writing EEM Policies Using the Cisco IOS CLI (continued)

Feature Name	Releases	Feature Information
Embedded Event Manager 2.1 (Software Modularity)	12.2(18)SXF4 Cisco IOS Software Modularity images	EEM 2.1 for Software Modularity images introduced the GOLD, system manager, and WDSysMon (Cisco IOS Software Modularity watchdog) event detectors, and the ability to display Cisco IOS Software Modularity processes and process metrics. The following sections provide information about this, feature: • Embedded Event Manager Policies, page 2 • Embedded Event Manager Built-In Environment Variables Used in EEM Applets, page 3 • Embedded Event Manager Applet Configuration: Examples, page 60 The following commands were introduced by this feature: **event gold**, **event process**, **show event manager metric process**. **Note** EEM 2.1 for Software Modularity images also supports the resource and RF event detectors introduced in EEM 2.2, but it does not support the enhanced object tracking event detector or the actions to read and set tracked objects.
Embedded Event Manager 2.2	12.4(2)T 12.2(31)SB3 12.2(33)SRB	EEM 2.2 introduced the enhanced object tracking, resource, and RF event detectors. The actions of reading and setting the state of a tracked object were also introduced. The following sections provide information about this feature: • Embedded Event Manager Policies, page 2 • Embedded Event Manager Built-In Environment Variables Used in EEM Applets, page 3 • Configuring and Tracking a Stub Object Using Embedded Event Manager, page 21 • Configuring and Tracking a Stub Object Using Embedded Event Manager: Example, page 65 The following commands were introduced or modified by this feature: **action track read**, **action track set**, **default-state**, **event resource**, **event rf**, **event track**, **show track**, **track stub-object**.
SNMP event detector delta environment variable[1]	12.4(11)T	A new SNMP event detector environment variable, _snmp_oid_delta_val, was introduced.

Table 7 Feature Information for Writing EEM Policies Using the Cisco IOS CLI (continued)

Feature Name	Releases	Feature Information
Embedded Event Manager 2.3	12.2(33)SXH 12.2(33)SB	EEM 2.3 introduced some new features relative to the Generic Online Diagnostics (GOLD) Event Detector on the Cisco Catalyst 6500 Series switches. The **event gold** command was enhanced in addition to the Tcl keywords—**action-notify**, **testing-type**, **test-name**, **test-id**, **consecutive-failure**, **platform-action**, and **maxrun**—for improved reaction to GOLD test failures and conditions The following section was updated to describe the enhanced functionality of the **event gold** command: • Embedded Event Manager Built-In Environment Variables Used in EEM Applets, page 3 Read-only variables were added under the **GOLD Event Detector** category to provide access to platform-wide and test-specific GOLD event detector information for a detected event.
Embedded Event Manager 2.4	12.4(20)T 12.2(33)SXI 12.2(33)SRE	EEM 2.4 is supported in Cisco IOS Release 12.4(20)T and later releases, and introduced several new features. The following sections provide information about this feature: • Embedded Event Manager Policies, page 2 • Embedded Event Manager Built-In Environment Variables Used in EEM Applets, page 3 • Configuring and Tracking a Stub Object Using Embedded Event Manager, page 21 • Configuring and Tracking a Stub Object Using Embedded Event Manager: Example, page 65 The following commands were introduced by this feature: **attribute (EEM)**, **correlate**, **event manager detector rpc**, **event manager directory user repository**, **event manager update user policy**, **event manager scheduler clear**, **event manager update user policy**, **event owner**, **event rpc**, **event snmp-notification**, **show event manager detector**, **show event manager version**, **trigger (EEM)**.

Table 7 Feature Information for Writing EEM Policies Using the Cisco IOS CLI (continued)

Feature Name	Releases	Feature Information
Embedded Event Manger 3.0	12.4(22)T 12.2(33)SRE	EEM 3.0 is supported in Cisco IOS Release 12.4(22)T and later releases, and introduced several new features. The following sections provide information about this feature: • Embedded Event Manager Policies, page 2 • Embedded Event Manager Built-In Environment Variables Used in EEM Applets, page 3 • Configuring and Tracking a Stub Object Using Embedded Event Manager, page 21 • Configuring and Tracking a Stub Object Using Embedded Event Manager: Example, page 65 The following commands were introduced or modified by this feature: **action add**, **action append**, **action break**, **action comment**, **action context retrieve**, **action context save**, **action continue**, **action decrement**, **action divide**, **action else**, **action elseif**, **action end**, **action exit**, **action foreach**, **action gets**, **action if**, **action if goto**, **action increment**, **action info type interface-names**, **action info type snmp getid**, **action info type snmp inform**, **action info type snmp oid**, **action info type snmp trap**, **action info type snmp var**, **action multiply**, **action puts**, **action regexp**, **action set (EEM)**, **action string compare**, **action string equal**, **action string first**, **action string index**, **action string last**, **action string length**, **action string match**, **action string range**, **action string replace**, **action string tolower**, **action string toupper**, **action string trim**, **action string trimleft**, **action string trimright**, **action subtract**, **action while**, **event cli**, **event ipsla**, **event manager detector routing**, **event manager scheduler**, **event manager scheduler clear**, **event manager scheduler hold**, **event manager scheduler modify**, **event manager scheduler release**, **event nf**, **event routing**, **show event manager policy active**, **show event manager policy pending**, and **show event manager scheduler**.

Table 7 Feature Information for Writing EEM Policies Using the Cisco IOS CLI (continued)

Feature Name	Releases	Feature Information
Embedded Event Manager 3.1	15.0(1)M	EEM 3.1 is supported in Cisco IOS Release 15.0(1)M and later releases, and introduced several new features. The following sections provide information about this feature: • Embedded Event Manager Policies, page 2 • Embedded Event Manager Built-In Environment Variables Used in EEM Applets, page 3 • Configuring and Tracking a Stub Object Using Embedded Event Manager, page 21 • Configuring and Tracking a Stub Object Using Embedded Event Manager: Example, page 65 The following commands were introduced or modified by this feature: **action syslog**, **description**, **event manager applet**, **event manager policy**, **event snmp-notification**, **event snmp-object**, **show event manager policy registered**, and **show event manager policy available**.

1. This is a minor enhancement. Minor enhancements are not typically listed in Feature Navigator.

Writing Embedded Event Manager Policies Using Tcl

First Published: October 31, 2005
Last Updated: December 9, 2009

This module describes how software developers can write and customize Embedded Event Manager (EEM) policies using Tool command language (Tcl) scripts to handle Cisco IOS software faults and events. EEM is a policy-driven process by means of which faults in the Cisco IOS software system are reported through a defined application programing interface (API). The EEM policy engine receives notifications when faults and other events occur. EEM policies implement recovery on the basis of the current state of the system and the actions specified in the policy for a given event. Recovery actions are triggered when the policy is run.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the "Feature Information for Writing Embedded Event Manager Policies Using Tcl" section on page 238.

Use Cisco Feature Navigator to find information about platform support and Cisco IOS, Catalyst OS, and Cisco IOS XE software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Contents

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

Prerequisites for Writing Embedded Event Manager Policies Using Tcl

- Before writing EEM policies, you should be familiar with the "Embedded Event Manager Overview" module.
- If you want to write EEM policies using the command-line interface (CLI) commands, you should be familiar with the "Writing Embedded Event Manager Policies Using the Cisco IOS CLI" module.

Information About Writing Embedded Event Manager Policies Using Tcl

To write EEM policies using Tcl, you should understand the following concepts:

EEM Policies

EEM offers the ability to monitor events and take informational or corrective action when the monitored events occur or reach a threshold. An EEM policy is an entity that defines an event and the actions to be taken when that event occurs. There are two types of EEM policies: an applet or a script. An applet is a simple form of policy that is defined within the command-line interface (CLI) configuration. A script is a form of policy that is written in Tool Command Language (Tcl).

EEM Applet

An EEM applet is a concise method for defining event screening criteria and the actions to be taken when that event occurs. In EEM applet configuration mode, three types of configuration statements are supported. The event commands are used to specify the event criteria to trigger the applet to run, the action commands are used to specify an action to perform when the EEM applet is triggered, and the **set** command is used to set the value of an EEM applet variable. Currently only the _exit_status variable is supported for the **set** command.

Only one event configuration command is allowed within an applet configuration. When applet configuration submode is exited and no event command is present, a warning is displayed stating that no event is associated with the applet. If no event is specified, the applet is not considered registered. When no action is associated with the applet, events are still triggered but no actions are performed. Multiple action configuration commands are allowed within an applet configuration. Use the **show event manager policy registered** command to display a list of registered applets.

Before modifying an EEM applet, be aware that the existing applet is not replaced until you exit applet configuration mode. While you are in applet configuration mode modifying the applet, the existing applet may be executing. It is safe to modify the applet without unregistering it, because changes are written to a temporary file. When you exit applet configuration mode, the old applet is unregistered and the new version is registered.

Action configuration commands within an applet are uniquely identified using the *label* argument, which can be any string value. Actions are sorted within an applet in ascending alphanumeric key sequence using the *label* argument as the sort key, and they are run using this sequence. The same *label* argument can be used in different applets; the labels must be unique only within one applet.

The Embedded Event Manager schedules and runs policies on the basis of an event specification that is contained within the policy itself. When applet configuration mode is exited, EEM examines the event and action commands that are entered and registers the applet to be run when a specified event occurs.

For more details about writing EEM policies using the Cisco IOS CLI, see the "Writing Embedded Event Manager Policies Using the Cisco IOS CLI" module.

EEM Script

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

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

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

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

EEM Policy Tcl Command Extension Categories

There are different categories of EEM policy Tcl command extensions.

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

Table 1 EEM Policy Tcl Command Extension Categories

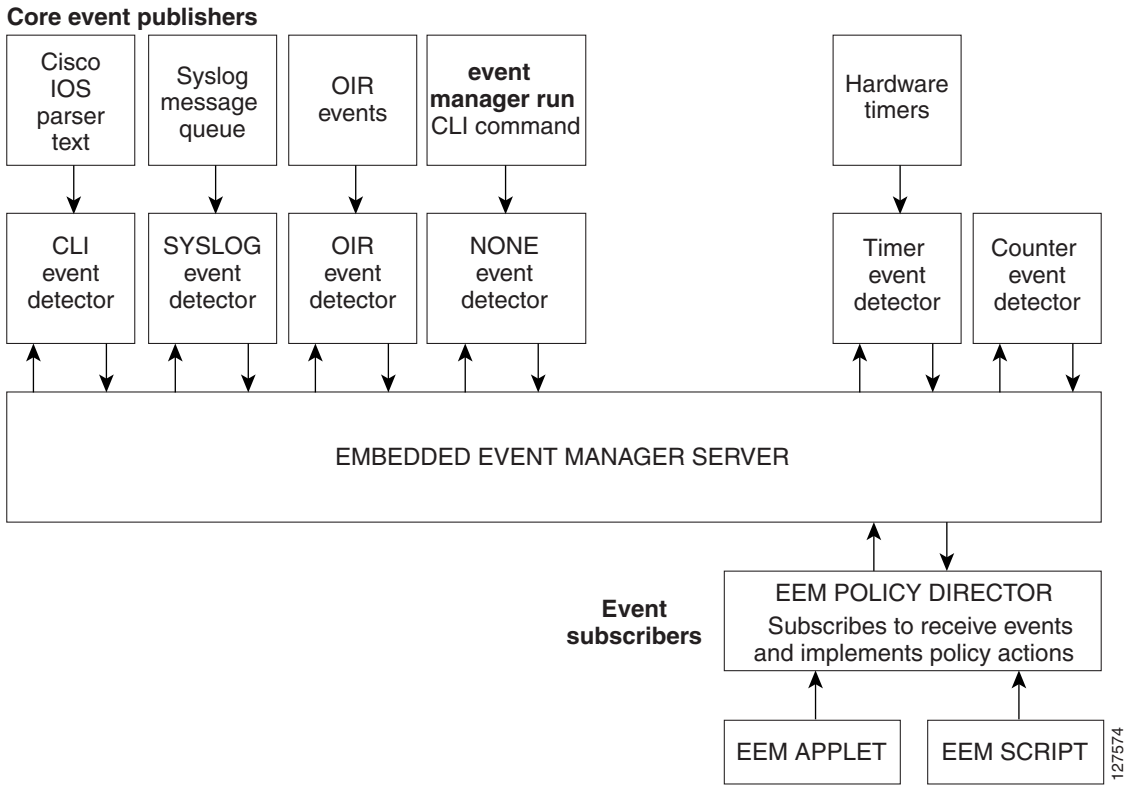
Category	Definition
EEM event Tcl command extensions (three types: event information, event registration, and event publish)	This category is represented by the **event_register**_*xxx* family of event-specific commands. There is a separate event information Tcl command extension in this category as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

General Flow of EEM Event Detection and Recovery

EEM is a flexible, policy-driven framework that supports in-box monitoring of different components of the system with the help of software agents known as event detectors. Figure 1 shows the relationship between the EEM server, the core event publishers (event detectors), and the event subscribers (policies). Basically, event publishers screen events and publish them when there is a match on an event specification that is provided by the event subscriber. Event detectors notify the EEM server when an event of interest occurs.

When an event or fault is detected, Embedded Event Manager determines from the event publishers—an example would be the OIR events publisher in Figure 1—if a registration for the encountered fault or event has occurred. EEM matches the event registration information with the event data itself. A policy registers for the detected event with the Tcl command extension **event_register**_*xxx*. The event information Tcl command extension **event_reqinfo** is used in the policy to query the Embedded Event Manager for information about the detected event.

Figure 1 Embedded Event Manager Core Event Detectors



Safe-Tcl

Safe-Tcl is a safety mechanism that allows untrusted Tcl scripts to run in an interpreter that was created in the safe mode. The safe interpreter has a restricted set of commands that prevent accessing some system resources and harming the host and other applications. For example, it does not allow commands to access critical Cisco IOS file system directories.

Cisco-defined scripts run in full Tcl mode, but user-defined scripts run in Safe-Tcl mode. Safe-Tcl allows Cisco to disable or customize individual Tcl commands. For more details about Tcl commands, go to http://www.tcl.tk/man/.

The following list of Tcl commands are restricted with a few exceptions. Restrictions are noted against each command or command keyword:

- **cd**—Change directory is not allowed to one of the restricted Cisco directory names.
- **encoding**—The commands **encoding names**, **encoding convertfrom**, and **encoding convertto** are permitted. The **encoding system** command with no arguments is permitted, but the **encoding system** command with the **?encoding?** keyword is not permitted.
- **exec**—Not permitted.
- **fconfigure**—Permitted.
- **file**—The following are permitted:
  - **file dirname**
  - **file exists**

  - **file extension**
  - **file isdirectory**
  - **file join**
  - **file pathtype**
  - **file rootname**
  - **file split**
  - **file stat**
  - **file tail**
- **file**—The following are not permitted:
  - **file atime**
  - **file attributes**
  - **file channels**
  - **file copy**
  - **file delete**
  - **file executable**
  - **file isfile**
  - **file link**
  - **file lstat**
  - **file mkdir**
  - **file mtime**
  - **file nativename**
  - **file normalize**
  - **file owned**
  - **file readable**
  - **file readlink**
  - **file rename**
  - **file rootname**
  - **file separator**
  - **file size**
  - **file system**
  - **file type**
  - **file volumes**
  - **file writable**
- **glob**—The **glob** command is not permitted when searching in one of the restricted Cisco directories. Otherwise, it is permitted.
- **load**—Only files that are in the user policy directory or the user library directory are permitted to be loaded. Static packages (for example, libraries that consist of C code) are not permitted to be loaded with the **load** command.

- **open**—The **open** command is not allowed for a file that is located in one of the restricted Cisco directories.
- **pwd**—The **pwd** command is not permitted.
- **socket**—The **socket** command is permitted.
- **source**—The **source** command is permitted for files that are in the user policy directory or the user library directory.

Bytecode Support for EEM 2.4

In Cisco IOS Release 12.4(20)T, EEM 2.4 introduces bytecode language (BCL) support by accepting files with the standard bytecode script extension .tbc. Tcl version 8.3.4 defines a BCL and includes a compiler that translates Tcl scripts into BCL. Valid EEM policy file extensions in EEM 2.4 for user and system policies are .tcl (Tcl Text files) and .tbc (Tcl bytecode files).

Storing Tcl scripts in bytecode improves the execution speed of the policy because the code is precompiled, creates a smaller policy size, and obscures the policy code. Obfuscation makes it a little more difficult to modify scripts and hides logic to preserve intellectual property rights.

Support for bytecode is being added to provide another option for release of supported and trusted code. We recommend that you only run well understood, or trusted and supported software on network devices. To generate Tcl bytecode for IOS EEM support, use TclPro versions 1.4 or 1.5.

To translate a Tcl script to bytecode you can use procomp, part of Free TclPro Compiler, or Active State Tcl Development Kit. When a Tcl script is compiled using procomp, the code is scrambled and a .tbc file is generated. The bytecode files are platform-independent and can be generated on any operating system on which TclPro is available, including Windows, Linux, and UNIX. Procomp is part of TclPro and available from http://www.tcl.tk/software/tclpro.

Registration Substitution

In addition to regular Tcl substitution, EEM 2.3 (in Cisco IOS Releases 12.2(33)SXH and 12.2(33)SB, and later releases) permits the substitution of an individual parameter in an EEM event registration statement line with an environment variable.

EEM 2.4 in Cisco IOS Release 12.4(20)T introduces the ability to replace multiple parameters in event registration statement lines with a single environment variable.

Note Only the first environment variable supports multiple parameter substitution. Individual parameters can still be specified with additional environment variables after the initial variable.

To illustrate the substitution, a single environment variable, $_eem_syslog_statement is configured as:

```
::cisco::eem::event_register_syslog pattern COUNT
```

Using the registration substitution, the $_eem_syslog_statement environment variable is used in the following EEM user policy:

```
$_eem_syslog_statement occurs $_eem_occurs_val
action_syslog "this is test 3"
```

Environment variables must be defined before a policy using them is registered. To define the $_eem_syslog_statement environment variable:

```
Router(config)# event manager environment eem_syslog_statement
::cisco::eem::event_register_syslog pattern COUNT
Router(config)# event manager environment eem_occurs_val 2
```

Cisco File Naming Convention for EEM

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

- An optional prefix—Mandatory.—indicating, if present, that this is a system policy that should be registered automatically at boot time if it is not already registered. For example: Mandatory.sl_text.tcl.
- A filename body part containing a two-character abbreviation (see Table 2) for the first event specified; an underscore part; and a descriptive field part that further identifies the policy.
- A filename suffix part defined as .tcl.

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

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

Table 2 Two-Character Abbreviation Specification

ap	event_register_appl
cl	event_register_cli
ct	event_register_counter
go	event_register_gold
if	event_register_interface
io	event_register_ioswdsysmon
la	event_register_ipsla
nf	event_register_nf
no	event_register_none
oi	event_register_oir
pr	event_register_process
rf	event_register_rf
rs	event_register_resource
rt	event_register_routing
rp	event_register_rpc
sl	event_register_syslog
sn	event_register_snmp
st	event_register_snmp_notification
so	event_register_snmp_object
tm	event_register_timer

Table 2 Two-Character Abbreviation Specification

tr	event_register_track
ts	event_register_timer_subscriber
wd	event_register_wdsysmon

How to Write Embedded Event Manager Policies Using Tcl

This section contains the following tasks:

Registering and Defining an EEM Tcl Script

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

Prerequisites

You must have a policy available that is written in the Tcl scripting language. Sample policies are provided—see the details in the "Sample EEM Policies" section on page 20 to see which policies are available for the Cisco IOS release image that you are using—and these sample policies are stored in the system policy directory.

SUMMARY STEPS

1. **enable**
2. **show event manager environment** [**all** | *variable-name*]
3. **configure terminal**
4. **event manager environment** *variable-name string*
5. Repeat Step 4 to configure all the environment variables required by the policy to be registered in Step 6.

6. **event manager policy** *policy-filename* [**type** {**system** | **user**}] [**trap**]
7. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show event manager environment** [**all** | *variable-name*] **Example:** `Router# show event manager environment all`	(Optional) Displays the name and value of EEM environment variables. • The optional **all** keyword displays all the EEM environment variables. • The optional *variable-name* argument displays information about the specified environment variable.
Step 3	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 4	**event manager environment** *variable-name string* **Example:** `Router(config)# event manager environment _cron_entry 0-59/2 0-23/1 * * 0-6`	Configures the value of the specified EEM environment variable. • In this example, the software assigns a CRON timer environment variable to be set to the second minute of every hour of every day.
Step 5	Repeat Step 4 to configure all the environment variables required by the policy to be registered in Step 6.	—
Step 6	**event manager policy** *policy-filename* [**type** {**system** | **user**}] [**trap**] **Example:** `Router(config)# event manager policy tm_cli_cmd.tcl type system`	Registers the EEM policy to be run when the specified event defined within the policy occurs. • Use the **system** keyword to register a Cisco-defined system policy. • Use the **user** keyword to register a user-defined system policy. • Use the **trap** keyword to generate an SNMP trap when the policy is triggered. • In this example, the sample EEM policy named tm_cli_cmd.tcl is registered as a system policy.
Step 7	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode and returns to privileged EXEC mode.

Examples

In the following example, the **show event manager environment** privileged EXEC command is used to display the name and value of all EEM environment variables.

```
Router# show event manager environment all

No.  Name                           Value
1    _cron_entry                    0-59/2 0-23/1 * * 0-6
2    _show_cmd                      show ver
3    _syslog_pattern                .*UPDOWN.*Ethernet1/0.*
4    _config_cmd1                   interface Ethernet1/0
5    _config_cmd2                   no shut
```

Displaying EEM Registered Policies

Perform this optional task to display EEM registered policies.

SUMMARY STEPS

1. **enable**
2. **show event manager policy registered** [**event-type** *event-name*] [**time-ordered** | **name-ordered**] [**detailed** *policy-filename*]

DETAILED STEPS

Step 1 **enable**

Enables privileged EXEC mode. Enter your password if prompted.

```
Router> enable
```

Step 2 **show event manager policy registered** [**event-type** *event-name*] [**time-ordered** | **name-ordered**] [**detailed** *policy-filename*]

Use this command with the **time-ordered** keyword to display information about currently registered policies sorted by time, for example:

```
Router# show event manager policy registered time-ordered

No.  Type    Event Type          Trap  Time Registered           Name
1    system  timer cron          Off   Wed May11  01:43:18 2005  tm_cli_cmd.tcl
 name {crontimer2} cron entry {0-59/1 0-23/1 * * 0-7}
 nice 0 priority normal maxrun 240

2    system  syslog              Off   Wed May11  01:43:28 2005  sl_intf_down.tcl
 occurs 1 pattern {.*UPDOWN.*Ethernet1/0.*}
 nice 0 priority normal maxrun 90

3    system  proc abort          Off   Wed May11  01:43:38 2005  pr_cdp_abort.tcl
 instance 1 path {cdp2.iosproc}
 nice 0 priority normal maxrun 20
```

Use this command with the **name-ordered** keyword to display information about currently registered policies sorted by name, for example:

```
Router# show event manager policy registered name-ordered

No.  Type    Event Type          Trap  Time Registered           Name
```

```
1    system  proc abort          Off   Wed May11  01:43:38 2005  pr_cdp_abort.tcl
 instance 1 path {cdp2.iosproc}
 nice 0 priority normal maxrun 20

2    system  syslog             Off   Wed May11  01:43:28 2005  sl_intf_down.tcl
 occurs 1 pattern {.*UPDOWN.*Ethernet1/0.*}
 nice 0 priority normal maxrun 90

3    system  timer cron         Off   Wed May11  01:43:18 2005  tm_cli_cmd.tcl
 name {crontimer2} cron entry {0-59/1 0-23/1 * * 0-7}
 nice 0 priority normal maxrun 240
```

Use this command with the **event-type** keyword to display information about currently registered policies for the event type specified in the *event-name* argument, for example:

```
Router# show event manager policy registered event-type syslog

No.  Type    Event Type          Time Registered           Name
1    system  syslog              Wed May11  01:43:28 2005  sl_intf_down.tcl
 occurs 1 pattern {.*UPDOWN.*Ethernet1/0.*}
 nice 0 priority normal maxrun 90
```

Unregistering EEM Policies

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

SUMMARY STEPS

1. **enable**
2. **show event manager policy registered** [**event-type** *event-name*] [**system** | **user**] [**time-ordered** | **name-ordered**] [**detailed** *policy-filename*]
3. **configure terminal**
4. **no event manager policy** *policy-filename*
5. **exit**
6. Repeat Step 2 to ensure that the policy has been removed.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show event manager policy registered** [**event-type** *event-name*] [**system** | **user**] [**time-ordered** | **name-ordered**] [**detailed** *policy-filename*] **Example:** `Router# show event manager policy registered`	(Optional) Displays the EEM policies that are currently registered. • The optional **system** or **user** keyword displays the registered system or user policies. • If no keywords are specified, EEM registered policies for all event types are displayed in time order.
Step 3	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 4	**no event manager policy** *policy-filename* **Example:** `Router(config)# no event manager policy pr_cdp_abort.tcl`	Removes the EEM policy from the configuration, causing the policy to be unregistered. • In this example, the **no** form of the command is used to unregister a specified policy.
Step 5	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode and returns to privileged EXEC mode.
Step 6	Repeat Step 2 to ensure that the policy has been removed. **Example:** `Router# show event manager policy registered`	—

Examples

In the following example, the **show event manager policy registered** privileged EXEC command is used to display the three EEM policies that are currently registered:

```
Router# show event manager policy registered

No.  Type    Event Type          Trap  Time Registered           Name
1    system  timer cron          Off   Tue Oct11  01:43:18 2005 tm_cli_cmd.tcl
 name {crontimer2} cron entry {0-59/1 0-23/1 * * 0-7}
 nice 0 priority normal maxrun 240.000

2    system  syslog              Off   Tue Oct11  01:43:28 2005 sl_intf_down.tcl
 occurs 1 pattern {.*UPDOWN.*Ethernet1/0.*}
 nice 0 priority normal maxrun 90.000
```

```
3    system  proc abort          Off   Tue Oct11  01:43:38 2005 pr_cdp_abort.tcl
 instance 1 path {cdp2.iosproc}
 nice 0 priority normal maxrun 20.000
```

After the current policies are displayed, it is decided to delete the pr_cdp_abort.tcl policy using the **no** form of the **event manager policy** command:

```
Router# configure terminal
Router(config)# no event manager policy pr_cdp_abort.tcl
Router(config)# exit
```

The **show event manager policy registered** privileged EXEC command is entered again to display the EEM policies that are currently registered. The policy pr_cdp_abort.tcl is no longer registered.

```
Router# show event manager policy registered

No.  Type    Event Type          Trap  Time Registered           Name
1    system  timer cron          Off   Tue Oct11  01:45:17 2005 tm_cli_cmd.tcl
 name {crontimer2} cron entry {0-59/1 0-23/1 * * 0-7}
 nice 0 priority normal maxrun 240.000

2    system  syslog              Off   Tue Oct11  01:45:27 2005 sl_intf_down.tcl
 occurs 1 pattern {.*UPDOWN.*Ethernet1/0.*}
 nice 0 priority normal maxrun 90.000
```

Suspending EEM Policy Execution

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

SUMMARY STEPS

1. **enable**
2. **show event manager policy registered** [**event-type** *event-name*] [**system | user**] [**time-ordered | name-ordered**] [**detailed** *policy-filename*]
3. **configure terminal**
4. **event manager scheduler suspend**
5. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show event manager policy registered** [**event-type** *event-name*] [**system** | **user**] [**time-ordered** | **name-ordered**] [**detailed** *policy-filename*] **Example:** `Router# show event manager policy registered`	(Optional) Displays the EEM policies that are currently registered. • The optional **system** or **user** keyword displays the registered system or user policies. • If no keywords are specified, EEM registered policies for all event types are displayed in time order.
Step 3	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 4	**event manager scheduler suspend** **Example:** `Router(config)# event manager scheduler suspend`	Immediately suspends the execution of all EEM policies.
Step 5	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode and returns to privileged EXEC mode.

Examples

In the following example, the **show event manager policy registered** privileged EXEC command is used to display all the EEM registered policies:

```
Router# show event manager policy registered

No.  Type    Event Type          Trap  Time Registered           Name
1    system  timer cron          Off   Sat Oct11  01:43:18 2003  tm_cli_cmd.tcl
 name {crontimer2} cron entry {0-59/1 0-23/1 * * 0-7}
 nice 0 priority normal maxrun 240.000

2    system  syslog              Off   Sat Oct11  01:43:28 2003  sl_intf_down.tcl
 occurs 1 pattern {.*UPDOWN.*Ethernet1/0.*}
 nice 0 priority normal maxrun 90.000

3    system  proc abort          Off   Sat Oct11  01:43:38 2003  pr_cdp_abort.tcl
 instance 1 path {cdp2.iosproc}
 nice 0 priority normal maxrun 20.000
```

The **event manager scheduler suspend** command is entered to immediately suspend the execution of all EEM policies:

```
Router# configure terminal
Router(config)# event manager scheduler suspend

*Nov  2 15:34:39.000: %HA_EM-6-FMS_POLICY_EXEC: fh_io_msg: Policy execution has been
suspended
```

Managing EEM Policies

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

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

SUMMARY STEPS

1. **enable**
2. **show event manager directory user** [**library** | **policy**]
3. **configure terminal**
4. **event manager directory user** {**library** *path* | **policy** *path*}
5. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show event manager directory user** [**library** | **policy**] **Example:** `Router# show event manager directory user library`	(Optional) Displays the directory to use for storing EEM user library or policy files. • The optional **library** keyword displays the directory to use for user library files. • The optional **policy** keyword displays the directory to use for user-defined EEM policies.
Step 3	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 4	**event manager directory user** {**library** *path* | **policy** *path*} **Example:** `Router(config)# event manager directory user library disk0:/usr/lib/tcl`	Specifies a directory to use for storing user library files or user-defined EEM policies. • Use the *path* argument to specify the absolute pathname to the user directory.
Step 5	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode and returns to privileged EXEC mode.

Examples

In the following example, the **show event manager directory user** privileged EXEC command is used to display the directory, if it exists, to use for storing EEM user library files:

```
Router# show event manager directory user library

disk0:/usr/lib/tcl
```

Modifying History Table Size and Displaying EEM History Data

Perform this optional task to change the size of the history tables and to display EEM history data.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **event manager history size** {**events** | **traps**} [*size*]
4. **exit**
5. **show event manager history events** [**detailed**] [**maximum** *number*]
6. **show event manager history traps** {**server** | **policy**}

DETAILED STEPS

Step 1 **enable**

Enables privileged EXEC mode. Enter your password if prompted.

```
Router> enable
```

Step 2 **configure terminal**

Enters global configuration mode.

```
Router# configure terminal
```

Step 3 **event manager history size** {**events** | **traps**} [*size*]

Use this command to change the size of the EEM event history table or the size of the EEM SNMP trap history table. In the following example, the size of the EEM event history table is changed to 30 entries:

```
Router(config)# event manager history size events 30
```

Step 4 **exit**

Exits global configuration mode and returns to privileged EXEC mode.

```
Router(config)# exit
```

Step 5 **show event manager history events** [**detailed**] [**maximum** *number*]

Use this command to display information about each EEM event that has been triggered.

```
Router# show event manager history events

No.  Time of Event             Event Type          Name
1    Fri Sep  9 13:48:40 2005  syslog              applet: one
2    Fri Sep  9 13:48:40 2005  syslog              applet: two
3    Fri Sep  9 13:48:40 2005  syslog              applet: three
4    Fri Sep  9 13:50:00 2005  timer cron          script: tm_cli_cmd.tcl
5    Fri Sep  9 13:51:00 2005  timer cron          script: tm_cli_cmd.tcl
```

Step 6 **show event manager history traps** [**server** | **policy**]

Use this command to display the EEM SNMP traps that have been sent either from the EEM server or from an EEM policy.

```
Router# show event manager history traps

No.  Time                      Trap Type           Name
1    Fri Sep  9 13:48:40 2005  server              applet: four
2    Fri Sep  9 13:57:03 2005  policy              script: no_snmp_test.tcl
```

Displaying Software Modularity Process Reliability Metrics Using EEM

Perform this optional task to display reliability metrics for Cisco IOS Software Modularity processes. The **show event manager metric processes** command was introduced in Cisco IOS Release 12.2(18)SXF4 and later releases and is supported only in Software Modularity images.

SUMMARY STEPS

1. **enable**
2. **show event manager metric processes** {**all** | *process-name*}

DETAILED STEPS

Step 1 **enable**

Enables privileged EXEC mode. Enter your password if prompted.

```
Router> enable
```

Step 2 **show event manager metric process** {**all** | *process-name*}

Use this command to display the reliability metric data for processes. The system keeps a record of when processes start and end, and this data is used as the basis for reliability analysis. In this partial example, the first and last entries showing the metric data for the processes on all the cards inserted in the system are displayed.

```
Router# show event manager metric process all

====================================
process name: devc-pty, instance: 1
sub_system id: 0, version: 00.00.0000
--------------------------------
last event type: process start
recent start time: Fri Oct10  20:34:40 2005
recent normal end time: n/a
recent abnormal end time: n/a
number of times started: 1
number of times ended normally: 0
number of times ended abnormally: 0
most recent 10 process start times:
-------------------------
Fri Oct10  20:34:40 2005
-------------------------

most recent 10 process end times and types:

cumulative process available time: 6 hours 30 minutes 7 seconds 378 milliseconds
cumulative process unavailable time: 0 hours 0 minutes 0 seconds 0 milliseconds
process availability:  0.100000000
number of abnormal ends within the past 60 minutes (since reload): 0
number of abnormal ends within the past 24 hours (since reload): 0
number of abnormal ends within the past 30 days (since reload): 0
.
.
.
====================================
process name: cdp2.iosproc, instance: 1
sub_system id: 0, version: 00.00.0000
--------------------------------
last event type: process start
recent start time: Fri Oct10  20:35:02 2005
recent normal end time: n/a
recent abnormal end time: n/a
number of times started: 1
number of times ended normally: 0
number of times ended abnormally: 0
most recent 10 process start times:
-------------------------
Fri Oct10  20:35:02 2005
-------------------------

most recent 10 process end times and types:

cumulative process available time: 6 hours 29 minutes 45 seconds 506 milliseconds
cumulative process unavailable time: 0 hours 0 minutes 0 seconds 0 milliseconds
process availability:  0.100000000
number of abnormal ends within the past 60 minutes (since reload): 0
number of abnormal ends within the past 24 hours (since reload): 0
number of abnormal ends within the past 30 days (since reload): 0
```

Troubleshooting Tips

Use the **debug event manager** command in privileged EXEC mode to troubleshoot EEM command operations. Use any debugging command with caution because the volume of output generated can slow or stop the router operations. We recommend that this command be used only under the supervision of a Cisco engineer.

Modifying the Sample EEM Policies

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

Sample EEM Policies

Cisco includes a set of sample policies shown in Table 3. You can copy the sample policies to a user directory and then modify the policies, or you can write your own policies. Tcl is currently the only Cisco-supported scripting language for policy creation. Tcl policies can be modified using a text editor such as Emacs. Policies must execute within a defined number of seconds of elapsed time, and the time variable can be configured within a policy. The default is currently 20 seconds.

Table 3 describes the sample EEM policies.

Table 3 Sample EEM Policy Descriptions

Name of Policy	Description
pr_cdp_abort.tcl	Introduced in Cisco IOS Release 12.2(18)SXF4 Software Modularity images. This policy monitors for cdp2.iosproc process abort events. It will log a message to SYSLOG and send an e-mail with the details of the abort.
pr_crash_reporter.tcl	Introduced in Cisco IOS Release 12.2(18)SXF4 Software Modularity images. This policy monitors for all process abort events. When an event occurs, the policy will send crash information, including the crashdump file, to the specified URL where a CGI script processes the data.
pr_iprouting_abort.tcl	Introduced in Cisco IOS Release 12.2(18)SXF4 Software Modularity images. This policy monitors for iprouting.iosproc process abort events. It will log a message to SYSLOG and send an e-mail with the details of the abort.
sl_intf_down.tcl	This policy runs when a configurable syslog message is logged. It will execute a configurable CLI command and e-mail the results.
tm_cli_cmd.tcl	This policy runs using a configurable CRON entry. It will execute a configurable CLI command and e-mail the results.

Table 3 Sample EEM Policy Descriptions (continued)

Name of Policy	Description
tm_crash_history.tcl	Introduced in Cisco IOS Release 12.2(18)SXF4 Software Modularity images. This policy runs at midnight every day and e-mails a process crash history report to a specified e-mail address.
tm_crash_reporter.tcl	Introduced in Cisco IOS Release 12.4(2)T. This policy runs 5 seconds after it is registered. If the policy is saved in the configuration, it will also run each time that the router is reloaded. The policy will prompt for the reload reason. If the reload was due to a crash, the policy will search for the latest crashinfo file and send this information to a specified URL location.
tm_fsys_usage.tcl	Introduced in Cisco IOS Release 12.2(18)SXF4 Software Modularity images. This policy runs using a configurable CRON entry and monitors disk space usage. A syslog message will be displayed if disk space usage crosses configurable thresholds.
wd_mem_reporter.tcl	Introduced in Cisco IOS Release 12.2(18)SXF4 Software Modularity images. This policy reports on low system memory conditions when the amount of memory available falls below 20 percent of the initial available system memory. A syslog message will be displayed and, optionally, an e-mail will be sent.

For more details about the sample policies available and how to run them, see the "EEM Event Detector Demo: Examples" section on page 37.

SUMMARY STEPS

1. **enable**
2. **show event manager policy available detailed** *policy-filename*
3. Cut and paste the contents of the sample policy displayed on the screen to a text editor.
4. Edit the policy and save it with a new filename.
5. Copy the new file back to the router flash memory.
6. **configure terminal**
7. **event manager directory user** {**library** *path* | **policy** *path*}
8. **event manager policy** *policy-filename* [**type** {**system** | **user**}] [**trap**]

DETAILED STEPS

Step 1 **enable**

Enables privileged EXEC mode. Enter your password if prompted.

```
Router> enable
```

Step 2 **show event manager policy available detailed** *policy-filename*

Displays the actual specified sample policy including details about the environment variables used by the policy and instructions for running the policy. In Cisco IOS 12.2(18)SXF4, the **detailed** keyword was introduced for the **show event manager policy available** and the **show event manager policy registered** commands. In Cisco IOS releases prior to 12.2(18)SXF4, you must copy one of the two Tcl scripts from the configuration examples section in this document (see the "Programming Policies with Tcl: Sample Scripts Example" section on page 43). In the following example, details about the sample policy tm_cli_cmd.tcl are displayed on the screen.

```
Router# show event manager policy available detailed tm_cli_cmd.tcl
```

Step 3 Cut and paste the contents of the sample policy displayed on the screen to a text editor.

Use the edit and copy functions to move the contents from the router to a text editor on another device.

Step 4 Edit the policy and save it with a new filename.

Use the text editor to modify the policy as a Tcl script. For file naming conventions, see the "Cisco File Naming Convention for EEM" section on page 8.

Step 5 Copy the new file back to the router flash memory.

Copy the file to the flash file system on the router—typically disk0:. For more details about copying files, see the "Using the Cisco IOS File System" chapter in the *Cisco IOS Configuration Fundamentals Configuration Guide*.

Step 6 **configure terminal**

Enters global configuration mode.

```
Router# configure terminal
```

Step 7 **event manager directory user** {**library** *path* | **policy** *path*}

Specifies a directory to use for storing user library files or user-defined EEM policies. In the following example, the user_library directory on disk0 is specified as the directory for storing user library files.

```
Router(config)# event manager directory user library disk0:/user_library
```

Step 8 **event manager policy** *policy-filename* [**type** {**system** | **user**}] [**trap**]

Registers the EEM policy to be run when the specified event defined within the policy occurs. In the following example, the new EEM policy named test.tcl is registered as a user-defined policy.

```
Router(config)# event manager policy test.tcl type user
```

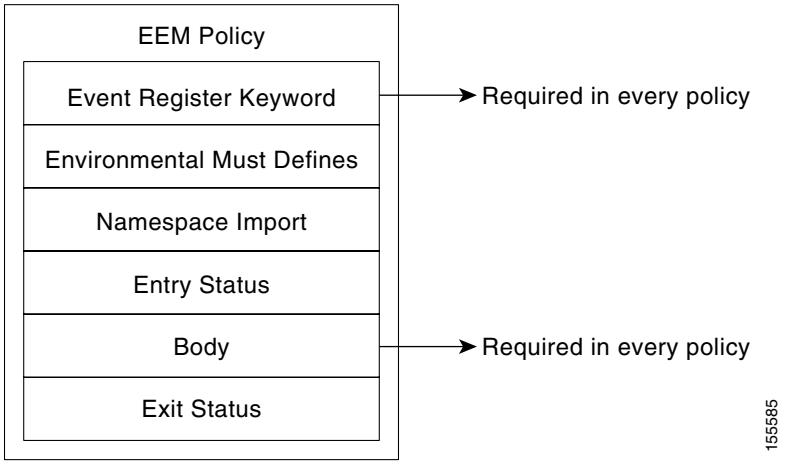
Programming EEM Policies with Tcl

Perform this task to help you program a policy using Tcl command extensions. We recommend that you copy an existing policy and modify it. There are two required parts that must exist in an EEM Tcl policy: the **event_register** Tcl command extension and the body. All other sections shown in the "Tcl Policy Structure and Requirements" concept are optional.

Tcl Policy Structure and Requirements

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

Figure 2 Tcl Policy Structure and Requirements



The start of every policy must describe and register the event to detect using an **event_register** Tcl command extension. This part of the policy schedules the running of the policy. For a list of the available EEM **event_register** Tcl command extensions, see the "EEM Event Registration Tcl Command Extensions" section on page 56. The following example Tcl code shows how to register the **event_register_timer** Tcl command extension:

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

The environment must defines section is optional and includes the definition of environment variables. The following example Tcl code shows how to check for, and define, some environment variables.

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

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

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

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

- The **event_reqinfo** event information Tcl command extension that is used to query the EEM for information about the detected event. For a list of the available EEM event information Tcl command extensions, see the "EEM Event Information Tcl Command Extension" section on page 117.
- The action Tcl command extensions, such as **action_syslog**, that are used to specify EEM specific actions. For a list of the available EEM action Tcl command extensions, see the "EEM Action Tcl Command Extensions" section on page 156.
- The system information Tcl command extensions, such as **sys_reqinfo_routername**, that are used to obtain general system information. For a list of the available EEM system information Tcl command extensions, see the "EEM System Information Tcl Command Extensions" section on page 183.
- Use of the SMTP library (to send e-mail notifications) or the CLI library (to run CLI commands) from a policy. For a list of the available SMTP library Tcl command extensions, see the "SMTP Library Command Extensions" section on page 201. For a list of the available CLI library Tcl command extensions, see the "CLI Library Command Extensions" section on page 204.
- The **context_save** and **context_retrieve** Tcl command extensions that are used to save Tcl variables for use by other policies.

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

```
# Query the event info and log a message.
array set arr_einfo [event_reqinfo]

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

global timer_type timer_time_sec
set timer_type $arr_einfo(timer_type)
set timer_time_sec $arr_einfo(timer_time_sec)

# Log a message.
set msg [format "timer event: timer type %s, time expired %s" \
        $timer_type [clock format $timer_time_sec]]

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

EEM Entry Status

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

Entry status designations may use one of three possible values: 0 (previous policy was successful), Not=0 (previous policy failed), and Undefined (no previous policy was executed).

EEM Exit Status

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

EEM Policies and Cisco Error Number

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

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

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

_cerrno: 32-Bit Error Return Values

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

```
XYSSSSSSSSSSSSSEEEEEEEEPPPPPPPPP
```

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

```
862439AE
```

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

```
10000110001001000011100110101110
```

This 32-bit integer is divided up into the five variables shown in Table 4.

Table 4 _cerrno: 32-Bit Error Return Value Variables

Variable	Description
XY	The error class (indicates the severity of the error). This variable corresponds to the first two bits in the 32-bit error return value; 10 in the case above, which indicates CERR_CLASS_WARNING: See Table 5 for the four possible error class encodings specific to this variable.
SSSSSSSSSSSSS	The subsystem number that generated the most recent error (13 bits = 8192 values). This is the next 13 bits of the 32-bit sequence, and its integer value is contained in $_cerr_sub_num.

Table 4 _cerrno: 32-Bit Error Return Value Variables (continued)

Variable	Description
EEEEEEEE	The subsystem specific error number (8 bits = 256 values). This segment is the next 8 bits of the 32-bit sequence, and the string corresponding to this error number is contained in $_cerr_sub_err.
PPPPPPPP	The pass-through POSIX error code (9 bits = 512 values). This represents the last of the 32-bit sequence, and the string corresponding to this error code is contained in $_cerr_posix_err.

Error Class Encodings for XY

The first variable, XY, references the possible error class encodings shown in Table 5.

:

Table 5 Error Class Encodings

00	CERR_CLASS_SUCCESS
01	CERR_CLASS_INFO
10	CERR_CLASS_WARNING
11	CERR_CLASS_FATAL

An error return value of zero means SUCCESS.

SUMMARY STEPS

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

DETAILED STEPS

Step 1 **enable**

Enables privileged EXEC mode. Enter your password if prompted.

```
Router> enable
```

Step 2 **show event manager policy available detailed** *policy-filename*

Displays the actual specified sample policy including details about the environment variables used by the policy and instructions for running the policy. In Cisco IOS 12.2(18)SXF4, the **detailed** keyword was introduced for the **show event manager policy available** and the **show event manager policy registered** commands. In Cisco IOS releases prior to 12.2(18)SXF4, you must copy one of the two Tcl scripts from the configuration examples section in this document (see the "Programming Policies with Tcl: Sample Scripts Example" section on page 43). In the following example, details about the sample policy tm_cli_cmd.tcl are displayed on the screen.

```
Router# show event manager policy available detailed tm_cli_cmd.tcl
```

Step 3 Cut and paste the contents of the sample policy displayed on the screen to a text editor.

Use the edit and copy functions to move the contents from the router to a text editor on another device. Use the text editor to edit the policy as a Tcl script.

Step 4 Define the required **event_register** Tcl command extension.

Choose the appropriate **event_register** Tcl command extension from Table 6 for the event that you want to detect, and add it to the policy.

Table 6 EEM Event Registration Tcl Command Extensions

Event Registration Tcl Command Extensions
event_register_appl
event_register_cli
event_register_counter
event_register_gold
event_register_interface
event_register_ioswdsysmon
event_register_ipsla
event_register_nf
event_register_none
event_register_oir
event_register_process
event_register_resource
event_register_rf
event_register_routing
event_register_rpc
event_register_snmp
event_register_snmp_notification

Table 6 EEM Event Registration Tcl Command Extensions (continued)

Event Registration Tcl Command Extensions
event_register_snmp_object
event_register_syslog
event_register_timer
event_register_timer_subscriber
event_register_track
event_register_wdsysmon

Step 5 Add the appropriate namespace under the ::cisco hierarchy.

Policy developers can use the new namespace ::cisco in Tcl policies in order to group all the extensions used by Cisco IOS EEM. There are two namespaces under the ::cisco hierarchy, and Table 7 shows which category of EEM Tcl command extension belongs under each namespace.

Table 7 Cisco IOS EEM Namespace Groupings

Namespace	Category of Tcl Command Extension
::cisco::eem	EEM event registration
	EEM event information
	EEM event publish
	EEM action
	EEM utility
	EEM context library
	EEM system information
	CLI library
::cisco::lib	SMTP library

Note Make sure that you import the appropriate namespaces or use the qualified command names when using the above commands.

Step 6 Program the must defines section to check for each environment variable that is used in this policy.

This is an optional step. Must defines are a section of the policy that tests whether any EEM environment variables that are required by the policy are defined before the recovery actions are taken. The must defines section is not required if the policy does not use any EEM environment variables. EEM environment variables for EEM scripts are Tcl global variables that are defined external to the policy before the policy is run. To define an EEM environment variable, use the Embedded Event Manager configuration command **event manager environment** CLI command. By convention all Cisco EEM environment variables begin with "_" (an underscore). In order to avoid future conflict, customers are urged not to define new variables that start with "_".

Note You can display the Embedded Event Manager environment variables set on your system by using the **show event manager environment** privileged EXEC command.

For example, Embedded Event Manager environment variables defined by the sample policies include e-mail variables. The sample policies that send e-mail must have the variables shown in Table 8 set in order to function properly.

Table 8 describes the e-mail-specific environment variables used in the sample EEM policies.

Table 8 E-mail-Specific Environmental Variables Used by the Sample Policies

Environment Variable	Description	Example
_email_server	A Simple Mail Transfer Protocol (SMTP) mail server used to send e-mail.	The e-mail server name can be in any one of the following template formats: • username:password@host • username@host • host
_email_to	The address to which e-mail is sent.	engineering@example.com
_email_from	The address from which e-mail is sent.	devtest@example.com
_email_cc	The address to which the e-mail must be copied.	manager@example.com

The following example of a must define section shows how to program a check for e-mail-specific environment variables.

Example 1 Example of Must Defines

```
if {![info exists _email_server]} {
    set result \
        "Policy cannot be run: variable _email_server has not been set"
    error $result $errorInfo
}
if {![info exists _email_from]} {
    set result \
        "Policy cannot be run: variable _email_from has not been set"
    error $result $errorInfo
}
if {![info exists _email_to]} {
    set result \
        "Policy cannot be run: variable _email_to has not been set"
    error $result $errorInfo
}
if {![info exists _email_cc]} {
    set result \
        "Policy cannot be run: variable _email_cc has not been set"
    error $result $errorInfo
}
```

Step 7 Program the body of the script.

In this section of the script, you can define any of the following:

- The **event_reqinfo** event information Tcl command extension that is used to query the EEM for information about the detected event.
- The action Tcl command extensions, such as **action_syslog**, that are used to specify EEM specific actions.

- The system information Tcl command extensions, such as **sys_reqinfo_routername**, that are used to obtain general system information.
- The **context_save** and **context_retrieve** Tcl command extensions that are used to save Tcl variables for use by other policies.
- Use of the SMTP library (to send e-mail notifications) or the CLI library (to run CLI commands) from a policy.

Step 8 Check the entry status to determine if a policy has previously run for this event.

If the prior policy is successful, the current policy may or may not require execution. Entry status designations may use one of three possible values: 0 (previous policy was successful), Not=0 (previous policy failed), and Undefined (no previous policy was executed).

Step 9 Check the exit status to determine whether or not to apply the default action for this event, if a default action exists.

A value of zero means do not perform the default action. A value of nonzero means perform the default action. The exit status will be passed to subsequent policies that are run for the same event.

Step 10 Set Cisco Error Number (_cerrno) Tcl global variables.

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

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

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

Step 11 Save the Tcl script with a new filename, and copy the Tcl script to the router.

Embedded Event Manager policy filenames adhere to the following specification:

- An optional prefix—Mandatory.—indicating, if present, that this is a system policy that should be registered automatically at boot time if it is not already registered. For example: Mandatory.sl_text.tcl.
- A filename body part containing a two-character abbreviation (see Table 2 on page 8) for the first event specified; an underscore character part; and a descriptive field part further identifying the policy.
- A filename suffix part defined as .tcl.

For more details, see the "Cisco File Naming Convention for EEM" section on page 8.

Copy the file to the flash file system on the router—typically disk0:. For more details about copying files, see the "Using the Cisco IOS File System" chapter in the *Cisco IOS Configuration Fundamentals Configuration Guide*.

Step 12 **configure terminal**

Enters global configuration mode.

```
Router# configure terminal
```

Step 13 **event manager directory user** {**library** *path* | **policy** *path*}

Specifies a directory to use for storing user library files or user-defined EEM policies. In the following example, the user_library directory on disk0 is specified as the directory for storing user library files.

```
Router(config)# event manager directory user library disk0:/user_library
```

Step 14 **event manager policy** *policy-filename* [**type** {**system** | **user**}] [**trap**]

Registers the EEM policy to be run when the specified event defined within the policy occurs. In the following example, the new EEM policy named cl_mytest.tcl is registered as a user-defined policy.

```
Router(config)# event manager policy cl_mytest.tcl type user
```

Step 15 Cause the policy to execute, and observe the policy.

To test that the policy runs, generate the conditions that will cause the policy to execute and observe that the policy runs as expected.

Step 16 Use debugging techniques if the policy does not execute correctly.

Use the Cisco IOS **debug event manager** CLI command with its various keywords to debug issues. Refer to the "Troubleshooting Tips" section on page 31 for details about using Tcl-specific keywords.

Troubleshooting Tips

- Use the **debug event manager tcl commands** CLI command to debug issues with Tcl extension commands. When enabled, this command displays all data that is passed in and read back from the TTY session that handles the CLI interactions. This data helps ensure users that the commands they are passing to the CLI are valid.
- The CLI library allows users to run CLI commands and obtain the output of commands in Tcl. Use the **debug event manager tcl cli-library** CLI command to debug issues with the CLI library.
- The SMTP library allows users to send e-mail messages to an SMTP e-mail server. Use the **debug event manager tcl smtp_library** CLI command to debug issues with the SMTP library. When enabled, this command displays all data that is passed in and read back from the SMTP library routines. This data helps ensure users that the commands they are passing to the SMTP library are valid.
- Tcl is a flexible language that allows you to override commands. For example, you can modify the **set** command and create a version of the **set** command that displays a message when a scalar variable is set. When the **set** command is entered in a policy, a message is displayed anytime a scalar variable is set, and this provides a way to debug scalar variables. To view an example of this debugging technique, see the "Tracing Tcl set Command Operations: Example" section on page 50.

To view examples of the some of these debugging techniques, see the "Debugging Embedded Event Manager Policies: Examples" section on page 48.

Creating an EEM User Tcl Library Index

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

SUMMARY STEPS

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

DETAILED STEPS

Step 1 On your workstation (UNIX, Linux, PC, or Mac) create a library directory and copy the Tcl library files into the directory.

The following example files can be used to create a tclIndex on a workstation running the Tcl shell:

lib1.tcl

```
proc test1 {} {
    puts "In procedure test1"
}

proc test2 {} {
    puts "In procedure test2"
}
```

lib2.tcl

```
proc test3 {} {
    puts "In procedure test3"
}
```

Step 2 **tclsh**

Use this command to enter the Tcl shell.

```
workstation% tclsh
```

Step 3 **auto_mkindex** *directory_name* ***.tcl**

Use the **auto_mkindex** command to create the tclIndex file. The tclIndex file that contains a directory of all the procedures contained in the Tcl library files. We recommend that you run auto_mkindex inside a directory because there can only be a single tclIndex file in any directory and you may have other Tcl files to be grouped together. Running auto_mkindex in a directory determines which tcl source file or files are indexed using a specific tclIndex.

```
workstation% auto_mkindex eem_library *.tcl
```

The following example TclIndex is created when the lib1.tcl and lib2.tcl files are in a library file directory and the **auto_mkindex** command is run.

tclIndex

```
# Tcl autoload index file, version 2.0
# This file is generated by the "auto_mkindex" command
# and sourced to set up indexing information for one or
# more commands.  Typically each line is a command that
# sets an element in the auto_index array, where the
# element name is the name of a command and the value is
# a script that loads the command.

set auto_index(test1) [list source [file join $dir lib1.tcl]]
set auto_index(test2) [list source [file join $dir lib1.tcl]]
set auto_index(test3) [list source [file join $dir lib2.tcl]]
```

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

Step 5 Copy a user-defined EEM policy file written in Tcl to the directory used for storing user-defined EEM policies on the target router. The directory can be the same directory used in Step 4.

The directory for storing user-defined EEM policies can be the same directory used in Step 4. The following example user-defined EEM policy can be used to test the Tcl library support in EEM.

libtest.tcl

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

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

global auto_index auto_path

puts [array names auto_index]

if { [catch {test1} result]} {
    puts "calling test1 failed result = $result $auto_path"
}

if { [catch {test2} result]} {
    puts "calling test2 failed result = $result $auto_path"
}

if { [catch {test3} result]} {
    puts "calling test3 failed result = $result $auto_path"
}
```

Step 6 **enable**

Enables privileged EXEC mode. Enter your password if prompted.

```
Router> enable
```

Step 7 **configure terminal**

Enables global configuration mode.

```
Router# configure terminal
```

Step 8 **event manager directory user library** *path*

Use this command to specify the EEM user library directory; this is the directory to which the files in Step 4 were copied.

```
router(config)# event manager directory user library disk2:/eem_library
```

Step 9 **event manager directory user policy** *path*

Use this command to specify the EEM user policy directory; this is the directory to which the file in Step 5 was copied.

```
router(config)# event manager directory user policy disk2:/eem_policies
```

Step 10 **event manager policy** *policy-name* [**type** {**system** | **user**} [**trap**]

Use this command to register a user-defined EEM policy. In this example, the policy named libtest.tcl is registered.

```
router(config)# event manager policy libtest.tcl
```

Step 11 **event manager run** *policy-name*

Use this command to manually run an EEM policy. In this example, the policy named libtest.tcl is run to test the Tcl support in EEM. The example output shows that the test for Tcl support in EEM was successful.

```
router(config)# event manager run libtest.tcl

The following output is displayed:

01:24:37: %HA_EM-6-LOG: libtest.tcl: In procedure test1
01:24:37: %HA_EM-6-LOG: libtest.tcl: In procedure test2
01:24:37: %HA_EM-6-LOG: libtest.tcl: In procedure test3
```

Creating an EEM User Tcl Package Index

Perform this task to create a Tcl package index file that contains a directory of all the Tcl packages and version information contained in a library of Tcl package files. Tcl packages are supported using the Tcl **package** keyword, and this support was introduced in Cisco IOS Release 12.4(11)T.

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

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

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

SUMMARY STEPS

1. On your workstation (UNIX, Linux, PC, or Mac) create a library directory and copy the Tcl package files into the directory.
2. **tclsh**
3. **pkg_mkIndex** *directory_name* ***.tcl**
4. Copy the Tcl package files from Step 1 and the pkgIndex file from Step 3 to the directory used for storing user library files on the target router.
5. Copy a user-defined EEM policy file written in Tcl to the directory used for storing user-defined EEM policies on the target router.
6. **enable**
7. **configure terminal**
8. **event manager directory user library** *path*
9. **event manager directory user policy** *path*
10. **event manager policy** *policy-filename* [**type {system | user}**] [**trap**]
11. **event manager run** *policy-filename*

DETAILED STEPS

Step 1 On your workstation (UNIX, Linux, PC, or Mac) create a library directory and copy the Tcl package files into the directory.

Step 2 **tclsh**

Use this command to enter the Tcl shell.

```
workstation% tclsh
```

Step 3 **pkg_mkindex** *directory_name* ***.tcl**

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

```
workstation% pkg_mkindex eem_library *.tcl
```

The following example pkgIndex is created when some Tcl package files are in a library file directory and the **pkg_mkindex** command is run.

pkgIndex

```
# Tcl package index file, version 1.1
# This file is generated by the "pkg_mkIndex" command
# and sourced either when an application starts up or
# by a "package unknown" script. It invokes the
# "package ifneeded" command to set up package-related
# information so that packages will be loaded automatically
# in response to "package require" commands. When this
# script is sourced, the variable $dir must contain the
# full path name of this file's directory.

package ifneeded xmlrpc 0.3 [list source [file join $dir xmlrpc.tcl]]
```

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

Step 5 Copy a user-defined EEM policy file written in Tcl to the directory used for storing user-defined EEM policies on the target router. The directory can be the same directory used in Step 4.

The directory for storing user-defined EEM policies can be the same directory used in Step 4. The following example user-defined EEM policy can be used to test the Tcl package support in EEM.

packagetest.tcl

```
::cisco::eem::event_register_none maxrun 1000000.000
#
# test if xmlrpc available
#
#
# Namespace imports
#
namespace import ::cisco::eem::*
namespace import ::cisco::lib::*
#
package require xmlrpc
puts "Did you get an error?"
```

Step 6 **enable**

Enables privileged EXEC mode. Enter your password if prompted.

```
Router> enable
```

Step 7 **configure terminal**

Enables global configuration mode.

```
Router# configure terminal
```

Step 8 **event manager directory user library** *path*

Use this command to specify the EEM user library directory; this is the directory to which the files in Step 4 were copied.

```
router(config)# event manager directory user library disk2:/eem_library
```

Step 9 **event manager directory user policy** *path*

Use this command to specify the EEM user policy directory; this is the directory to which the file in Step 5 was copied.

```
router(config)# event manager directory user policy disk2:/eem_policies
```

Step 10 **event manager policy** *policy-name* **[type {system | user} [trap]**

Use this command to register a user-defined EEM policy. In this example, the policy named packagetest.tcl is registered.

```
router(config)# event manager policy packagetest.tcl
```

Step 11 **event manager run** *policy-name*

Use this command to manually run an EEM policy. In this example, the policy named packagetest.tcl is run to test the Tcl package support in EEM.

```
router(config)# event manager run packagetest.tcl
```

Configuration Examples for Writing Embedded Event Manager Policies Using Tcl

This section contains the following configuration examples:

Assigning a Username for a Tcl Session: Examples

The following example shows how to set a username to be associated with a Tcl session. If you are using authentication, authorization, and accounting (AAA) security and implement authorization on a command basis, you should use the **event manager session cli username** command to set a username to be associated with a Tcl session. The username is used when a Tcl policy executes a CLI command. TACACS+ verifies each CLI command using the username associated with the Tcl session that is running the policy. Commands from Tcl policies are not usually verified because the router must be in privileged EXEC mode to register the policy. In the example, the username is yourname, and this is the username that is used whenever a CLI command session is initiated from within an EEM policy.

```
configure terminal
 event manager session cli username yourname
 end
```

EEM Event Detector Demo: Examples

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

EEM Sample Policy Descriptions

This configuration example features four of the sample EEM policies:

- sl_intf_down.tcl—Is run when a configurable syslog message is logged. It executes up to two configurable CLI commands and e-mails the results.
- tm_cli_cmd.tcl—Is run using a configurable CRON entry. It executes a configurable CLI command and e-mails the results.

- tm_crash_reporter.tcl—Introduced in Cisco IOS Release 12.4(2)T, 12.2(31)SB3, and 12.2(33)SRB. Is run 5 seconds after it is registered and 5 seconds after the router boots up. When triggered, the script attempts to find the reload reason. If the reload reason was due to a crash, the policy searches for the related crashinfo file and sends this information to a URL location specified by the user in the environment variable _crash_reporter_url.
- tm_fsys_usage.tcl—Introduced in Cisco IOS Release 12.2(18)SXF4 Cisco IOS Software Modularity images. This policy runs using a configurable CRON entry and monitors disk space usage. A syslog message is displayed if disk space usage crosses configurable thresholds.

Event Manager Environment Variables for the Sample Policies

Event manager environment variables are Tcl global variables that are defined external to the EEM policy before the policy is registered and run. The sample policies require three of the e-mail environment variables to be set (see Table 8 on page 29 for a list of the e-mail variables); only _email_cc is optional. Other required and optional variable settings are outlined in the following tables.

Table 9 describes the EEM environment variables that must be set before the sl_intf_down.tcl sample policy is run.

Table 9 Environment Variables Used in the sl_intf_down.tcl Policy

Environment Variable	Description	Example
_config_cmd1	The first configuration command that is executed.	**interface Ethernet1/0**
_config_cmd2	The second configuration command that is executed. This variable is optional and need not be specified.	**no shutdown**
_syslog_pattern	A regular expression pattern match string that is used to compare syslog messages to determine when the policy runs.	.*UPDOWN.*FastEthernet0/0.*

Table 10 describes the EEM environment variables that must be set before the tm_cli_cmd.tcl sample policy is run.

Table 10 Environment Variables Used in the tm_cli_cmd.tcl Policy

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

Table 11 describes the EEM environment variables that must be set before the tm_crash_reporter.tcl sample policy is run.

Table 11 Environment Variables Used in the tm_crash_reporter.tcl Policy

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

Table 12 describes the EEM environment variables that must be set before the tm_fsys_usage.tcl sample policy is run.

Table 12 Environment Variables Used in the tm_fsys_usage.tcl Policy

Environment Variable	Description	Example
_tm_fsys_usage_cron	A CRON specification that is used in the **event_register** Tcl command extension. If unspecified, the tm_fsys_usage.tcl policy is triggered once per minute. This variable is optional and need not be specified.	0-59/1 0-23/1 * * 0-7
_tm_fsys_usage_debug	When this variable is set to a value of 1, disk usage information is displayed for all entries in the system. This variable is optional and need not be specified.	1
_tm_fsys_usage_ freebytes	Free byte threshold for systems or specific prefixes. If free space falls below a given value, a warning is displayed. This variable is optional and need not be specified.	disk2:98000000
_tm_fsys_usage_percent	Disk usage percentage thresholds for systems or specific prefixes. If the disk usage percentage exceeds a given percentage, a warning is displayed. If unspecified, the default disk usage percentage is 80 percent for all systems. This variable is optional and need not be specified.	nvram:25 disk2:5

Registration of Some EEM Policies

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

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

Basic Configuration Details for All Sample Policies

To allow e-mail to be sent from the Embedded Event Manager, the **hostname** and **ip domain-name** commands must be configured. The EEM environment variables must also be set. After a Cisco IOS image has been booted, use the following initial configuration, substituting appropriate values for your network. The environment variables for the tm_fsys_usage sample policy (see Table 12 on page 39) are all optional and are not listed here:

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

Using the Sample Policies

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

- Running the sl_intf_down.tcl Sample Policy, page 40
- Running the tm_cli_cmd.tcl Sample Policy, page 41
- Running the tm_crash_reporter.tcl Sample Policy, page 41
- Running the tm_fsys_usage.tcl Sample Policy, page 42

Running the sl_intf_down.tcl Sample Policy

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

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

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

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

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

Running the tm_cli_cmd.tcl Sample Policy

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

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

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

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

Running the tm_crash_reporter.tcl Sample Policy

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

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

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

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

Running the tm_fsys_usage.tcl Sample Policy

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

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

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

Programming Policies with Tcl: Sample Scripts Example

This section contains two of the sample policies that are included as EEM system policies. These two policies were introduced in Cisco IOS Release 12.3(14)T and 12.2(18)SXF5 images. For more details about these policies, see the "EEM Event Detector Demo: Examples" section on page 37.

tm_cli_cmd.tcl Sample Policy

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

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

#------------------------------------------------------------------
# EEM policy that will periodically execute a cli command and email the
# results to a user.
#
# July 2005, Cisco EEM team
#
# Copyright (c) 2005 by cisco Systems, Inc.
# All rights reserved.
#------------------------------------------------------------------

### The following EEM environment variables are used:
###
### _cron_entry (mandatory)            - A CRON specification that determines
###                                      when the policy will run. See the
###                                      IOS Embedded Event Manager
###                                      documentation for more information
###                                      on how to specify a cron entry.
### Example: _cron_entry                 0-59/1 0-23/1 * * 0-7
###
### _log_file (mandatory without _email_....)
###                                    - A filename to append the output to.
###                                      If this variable is defined, the
###                                      output is appended to the specified
###                                      file with a timestamp added.
### Example: _log_file                   disk0:/my_file.log
###
### _email_server (mandatory without _log_file)
###                                    - A Simple Mail Transfer Protocol (SMTP)
###                                      mail server used to send e-mail.
### Example: _email_server               mailserver.example.com
###
### _email_from (mandatory without _log_file)
###                                    - The address from which e-mail is sent.
### Example: _email_from                 devtest@example.com
###
### _email_to (mandatory without _log_file)
###                                    - The address to which e-mail is sent.
### Example: _email_to                   engineering@example.com
###
### _email_cc (optional)               - The address to which the e-mail must
###                                      be copied.
### Example: _email_cc                   manager@example.com
###
### _show_cmd (mandatory)              - The CLI command to be executed when
###                                      the policy is run.
```

```
### Example: _show_cmd                    show version
###

# check if all required environment variables exist
# If any required environment variable does not exist, print out an error msg and quit
if {![info exists _log_file]} {
    if {![info exists _email_server]} {
    set result \
        "Policy cannot be run: variable _log_file or _email_server has not been set"
    error $result $errorInfo
    }
    if {![info exists _email_from]} {
    set result \
        "Policy cannot be run: variable _log_file or _email_from has not been set"
    error $result $errorInfo
    }
    if {![info exists _email_to]} {
    set result \
        "Policy cannot be run: variable _log_file ore _email_to has not been set"
    error $result $errorInfo
    }
    if {![info exists _email_cc]} {
    #_email_cc is an option, must set to empty string if not set.
    set _email_cc ""
    }
}

if {![info exists _show_cmd]} {
    set result \
        "Policy cannot be run: variable _show_cmd has not been set"
    error $result $errorInfo
}


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

# query the event info and log a message
array set arr_einfo [event_reqinfo]

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

global timer_type timer_time_sec
set timer_type $arr_einfo(timer_type)
set timer_time_sec $arr_einfo(timer_time_sec)

# log a message
set msg [format "timer event: timer type %s, time expired %s" \
        $timer_type [clock format $timer_time_sec]]

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

# 1. execute the command
if [catch {cli_open} result] {
    error $result $errorInfo
```

```
} else {
    array set cli1 $result
}
if [catch {cli_exec $cli1(fd) "en"} result] {
    error $result $errorInfo
}
# save exact execution time for command
set time_now [clock seconds]
# execute command
if [catch {cli_exec $cli1(fd) $_show_cmd} result] {
    error $result $errorInfo
} else {
    set cmd_output $result
    # format output: remove trailing router prompt
    regexp {\n*(.*\n)([^\n]*)$} $result dummy cmd_output
}
if [catch {cli_close $cli1(fd) $cli1(tty_id)} result] {
    error $result $errorInfo
}

# 2. log the success of the CLI command
set msg [format "Command \"%s\" executed successfully" $_show_cmd]
action_syslog priority info msg $msg
if {$_cerrno != 0} {
    set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
        $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}

# 3. if _log_file is defined, then attach it to the file
if {[info exists _log_file]} {
    # attach output to file
    if [catch {open $_log_file a+} result] {
        error $result
    }
    set fileD $result
    # save timestamp of command execution
    #      (Format = 00:53:44 PDT Mon May 02 2005)
    set time_now [clock format $time_now -format "%T %Z %a %b %d %Y"]
    puts $fileD "%%% Timestamp = $time_now"
    puts $fileD $cmd_output
    close $fileD
}

# 4. if _email_server is defined send the email out
if {[info exists _email_server]} {
    set routername [info hostname]
    if {[string match "" $routername]} {
    error "Host name is not configured"
    }

    if [catch {smtp_subst [file join $tcl_library email_template_cmd.tm]} \
        result] {
    error $result $errorInfo
    }

    if [catch {smtp_send_email $result} result] {
    error $result $errorInfo
    }
}
```

sl_intf_down.tcl Sample Policy

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

```
::cisco::eem::event_register_syslog occurs 1 pattern $_syslog_pattern maxrun 90

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

### The following EEM environment variables are used:
###
### _syslog_pattern (mandatory)        - A regular expression pattern match string
###                                      that is used to compare syslog messages
###                                      to determine when policy runs
### Example: _syslog_pattern             .*UPDOWN.*FastEthernet0/0.*
###
### _email_server (mandatory)          - A Simple Mail Transfer Protocol (SMTP)
###                                      mail server used to send e-mail.
### Example: _email_server               mailserver.example.com
###
### _email_from (mandatory)            - The address from which e-mail is sent.
### Example: _email_from                 devtest@example.com
###
### _email_to (mandatory)              - The address to which e-mail is sent.
### Example: _email_to                   engineering@example.com
###
### _email_cc (optional)               - The address to which the e-mail must
###                                      be copied.
### Example: _email_cc                   manager@example.com
###
### _config_cmd1 (optional)            - The first configuration command that
###                                      is executed.
### Example: _config_cmd1                interface Ethernet1/0
###
### _config_cmd2 (optional)            - The second configuration command that
###                                      is executed.
### Example: _config_cmd2                no shutdown
###

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

```
}
if {![info exists _email_cc]} {
     #_email_cc is an option, must set to empty string if not set.
     set _email_cc ""
}

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

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

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

set msg $arr_einfo(msg)
set config_cmds ""

# 2. execute the user-defined config commands
if [catch {cli_open} result] {
    error $result $errorInfo
} else {
    array set cli1 $result
}
if [catch {cli_exec $cli1(fd) "en"} result] {
    error $result $errorInfo
}
if [catch {cli_exec $cli1(fd) "config t"} result] {
    error $result $errorInfo
}

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

    append config_cmds $_config_cmd1
}

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

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

after 60000
# 3. send the notification email
set routername [info hostname]
if {[string match "" $routername]} {
    error "Host name is not configured"
}
```

```
if [catch {smtp_subst [file join $tcl_library email_template_cfg.tm]} result] {
    error $result $errorInfo
}
if [catch {smtp_send_email $result} result] {
    error $result $errorInfo
}
```

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

```
email_template_cfg.tm

Mailservername: $_email_server
From: $_email_from
To: $_email_to
Cc: $_email_cc
Subject: From router $routername: Periodic $_show_cmd Output

$cmd_output
```

Debugging Embedded Event Manager Policies: Examples

The following examples show how to debug the CLI library and the SMTP library.

Debugging the CLI Library

The CLI library allows users to run CLI commands and obtain the output of commands in Tcl. An Embedded Event Manager **debug** command has been provided for users of this library. The command to enable CLI library debugging is **debug event manager tcl cli_library**. When enabled, this command displays all data that is passed in and read back from the TTY session that handles the CLI interactions. This data helps ensure users that the commands that they are passing to the CLI are valid.

Example of the debug event manager tcl cli_library Command

This example uses the sample policy sl_intf_down.tcl. When triggered, sl_intf_down.tcl passes a configuration command to the CLI through the CLI library. The command passed in below is **show event manager environment**. This command is not a valid command in configuration mode. Without the **debug** command enabled, the output is shown below:

```
00:00:57:sl_intf_down.tcl[0]:config_cmds are show eve man env
00:00:57:%SYS-5-CONFIG_I:Configured from console by vty0
```

Notice that with the output above the user would not know whether or not the command succeeded in the CLI. With the **debug event manager tcl cli_library** command enabled, the user sees the following:

```
01:17:07: sl_intf_down.tcl[0]: DEBUG(cli_lib) : CTL : cli_open called.
01:17:07: sl_intf_down.tcl[0]: DEBUG(cli_lib) : OUT : nelson>
01:17:07: sl_intf_down.tcl[0]: DEBUG(cli_lib) : IN  : nelson>enable
01:17:07: sl_intf_down.tcl[0]: DEBUG(cli_lib) : OUT : nelson#
01:17:07: sl_intf_down.tcl[0]: DEBUG(cli_lib) : IN  : nelson#configure terminal
01:17:07: sl_intf_down.tcl[0]: DEBUG(cli_lib) : OUT : Enter configuration commands, one
per line.  End with CNTL/Z.
01:17:07: sl_intf_down.tcl[0]: DEBUG(cli_lib) : OUT : nelson(config)#
01:17:07: sl_intf_down.tcl[0]: DEBUG(cli_lib) : IN  : nelson(config)#show event manager
environment
01:17:07: sl_intf_down.tcl[0]: DEBUG(cli_lib) : OUT :                  ^
01:17:07: sl_intf_down.tcl[0]: DEBUG(cli_lib) : OUT : % Invalid input detected at '^'
marker.
01:17:07: sl_intf_down.tcl[0]: DEBUG(cli_lib) : OUT : nelson(config)#
01:17:07: sl_intf_down.tcl[0]: DEBUG(cli_lib) : IN  : nelson(config)#end
01:17:07: sl_intf_down.tcl[0]: DEBUG(cli_lib) : OUT : nelson#
01:17:07: sl_intf_down.tcl[0]: DEBUG(cli_lib) : CTL : cli_close called.
```

```
01:17:07: sl_intf_down.tcl[0]: DEBUG(cli_lib) : IN  : nelson#exit
01:17:07: sl_intf_down.tcl[0]: config_cmds are show event manager environment
01:17:07: %SYS-5-CONFIG_I: Configured from console by vty0
```

The output above shows that **show event manager environment** is an invalid command in configuration mode. The IN keyword signifies all data passed in to the TTY through the CLI library. The OUT keyword signifies all data read back from the TTY through the CLI library. The CTL keyword signifies helper functions used in the CLI library. These helper functions are used to set up and remove connections to the CLI.

Debugging the SMTP Library

The SMTP library allows users to send e-mail messages to an SMTP e-mail server. An Embedded Event Manager **debug** command has been provided for users of this library. The command to enable SMTP library debugging is **debug event manager tcl smtp_library**. When enabled, this command displays all data that is passed in and read back from the SMTP library routines. This data helps ensure users that the commands that they are passing to the SMTP library are valid.

Example of the debug event manager tcl smtp_library Command

This example uses the sample policy tm_cli_cmd.tcl. When triggered, tm_cli_cmd.tcl runs the command **show event manager policy available system** through the CLI library. The result is then mailed to a user through the SMTP library. The output will help debug any issues related to using the SMTP library.

With the **debug event manager tcl smtp_library** command enabled, the users see the following on the console:

```
00:39:46: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_read  : 220 XXXX.example.com ESMTP
XXXX 1.1.0; Tue, 25 Jun 2002 14:20:39 -0700 (PDT)
00:39:46: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : HELO XXXX.example.com
00:39:46: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_read  : 250 XXXX.example.com Hello
XXXX.example.com [XXXX], pleased to meet you
00:39:46: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : MAIL FROM:<XX@example.com>
00:39:46: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_read  : 250 <XX@example.com>... Sender
ok
00:39:46: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : RCPT TO:<XX@example.com>
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_read  : 250 <XX@example.com>...
Recipient ok
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : RCPT TO:<XX@example.com>
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_read  : 250 <XX@example.com>...
Recipient ok
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : DATA
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_read  : 354 Enter mail, end with "."
on a line by itself
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : Date: 25 Jun 2002 14:35:00 UTC
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : Message-ID:
<20020625143500.2387058729877@XXXX.example.com>
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : From: XX@example.com
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : To: XX@example.com
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : Cc: XX@example.com
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : Subject: From router nelson:
Periodic show eve man po ava system Output
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : No.  Type   Time Created
Name
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : 1    system  Fri May3
20:42:34 2002      pr_cdp_abort.tcl
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : 2    system  Fri May3
20:42:54 2002      pr_iprouting_abort.tcl
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : 3    system  Wed Apr3
02:16:33 2002      sl_intf_down.tcl
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : 4    system  Mon Jun24
23:34:16 2002      tm_cli_cmd.tcl
```

```
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : 5    system  Wed Mar27
05:53:15 2002      tm_crash_hist.tcl
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : nelson#
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write :
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : .
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_read  : 250 ADE90179 Message accepted
for delivery
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : QUIT
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_read  : 221 XXXX.example.com closing
connection
```

Tracing Tcl set Command Operations: Example

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

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

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

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

RPC Event Detector: Example

```
TCL script (rpccli.tcl):

::cisco::eem::event_register_rpc

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

proc run_cli { clist } {
    set rbuf ""

    if {[llength $clist] < 1} {
    return -code ok $rbuf
    }

    if {[catch {cli_open} result]} {
        return -code error $result
    } else {
    array set cliarr $result
    }

    if {[catch {cli_exec $cliarr(fd) "enable"} result]} {
        return -code error $result
    }

    if {[catch {cli_exec $cliarr(fd) "term length 0"} result]} {
        return -code error $result
    }

    foreach cmd $clist {
```

```
    if {[catch {cli_exec $cliarr(fd) $cmd} result]} {
            return -code error $result
    }

    append rbuf $result
    }

    if {[catch {cli_close $cliarr(fd) $cliarr(tty_id)} result]} {
        puts "WARNING: $result"
    }

    return -code ok $rbuf
}

proc run_cli_interactive { clist } {
    set rbuf ""

    if {[llength $clist] < 1} {
    return -code ok $rbuf
    }

    if {[catch {cli_open} result]} {
        return -code error $result
    } else {
    array set cliarr $result
    }

    if {[catch {cli_exec $cliarr(fd) "enable"} result]} {
        return -code error $result
    }

    if {[catch {cli_exec $cliarr(fd) "term length 0"} result]} {
        return -code error $result
    }

    foreach cmd $clist {
        array set sendexp $cmd

    if {[catch {cli_write $cliarr(fd) $sendexp(send)} result]} {
            return -code error $result
    }

    foreach response $sendexp(responses) {
        array set resp $response

        if {[catch {cli_read_pattern $cliarr(fd) $resp(expect)} result]} {
                return -code error $result
        }

        if {[catch {cli_write $cliarr(fd) $resp(reply)} result]} {
                return -code error $result
        }
    }

    if {[catch {cli_read $cliarr(fd)} result]} {
            return -code error $result
    }

    append rbuf $result
    }

    if {[catch {cli_close $cliarr(fd) $cliarr(tty_id)} result]} {
        puts "WARNING: $result"
    }
```

```
    return -code ok $rbuf
}

array set arr_einfo [event_reqinfo]

set args $arr_einfo(argc)

set cmds [list]

for { set i 0 } { $i < $args } { incr i } {
    set arg "arg${i}"
    # Split each argument on the '^' character.  The first element is
    # the command, and each subsequent element is a prompt followed by
    # a response to that prompt.
    set cmdlist [split $arr_einfo($arg) "^"]
    set cmdarr(send) [lindex $cmdlist 0]
    set cmdarr(responses) [list]
    if { [expr ([llength $cmdlist] - 1) % 2] != 0 } {
    return -code 88
    }
    set cmdarr(responses) [list]
    for { set j 1 } { $j < [llength $cmdlist] } { incr j 2 } {
    set resps(expect) [lindex $cmdlist $j]
    set resps(reply) [lindex $cmdlist [expr $j + 1]]
    lappend cmdarr(responses) [array get resps]
    }
    lappend cmds [array get cmdarr]
}

set rc [catch {run_cli_interactive $cmds} output]

if { $rc != 0 } {
    error $output $errorInfo
    return -code 88
}

puts $output
```

Where to Go Next

- For information about EEM overview, go to "Embedded Event Manager Overview" module.
- For information about writing EEM policies using the Cisco IOS CLI, go to the "Writing Embedded Event Manager Policies Using the Cisco IOS CLI" module.

Additional References

The following sections provide references related to writing Embedded Event Manager policies using Tcl.

Related Documents

Related Topic	Document Title
Cisco IOS commands	*Cisco IOS Master Commands List, All Releases*
Network Management commands (including EEM commands): complete command syntax, defaults, command mode, command history, usage guidelines, and examples	*Cisco IOS Network Management Command Reference*
Embedded Event Manager overview	Embedded Event Manager Overview module.
Embedded Event Manager policy writing using the CLI	Writing Embedded Event Manager Policies Using the Cisco IOS CLI module
Embedded Resource Manager	Embedded Resource Manager module

MIBs

MIB	MIBs Link
CISCO-EMBEDDED-EVENT-MGR-MIB	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

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

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

EEM Policy Tcl Command Extension Reference

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

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

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

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

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

  ```
  [type ?]
  ```

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

  ```
  priority low|normal|high
  ```

EEM Event Registration Tcl Command Extensions

event_register_appl

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

In order to register for an application event, a subsystem must be specified. Either a Tcl policy or the internal Embedded Event Manager (EEM) API can publish an application event. If the event is being published by a policy, the sub_system argument that is reserved for a policy is 798.

Syntax

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

Arguments

tag	(Optional) String identifying a tag that can be used with the trigger Tcl command extension to support multiple event statements within a Tcl script.
sub_system	(Mandatory) Number assigned to the EEM policy that published the application event. The number is set to 798 because all other numbers are reserved for Cisco use. If this argument is not specified, all components are matched.
type	(Mandatory) Event subtype within the specified event. The sub_system and type arguments uniquely identify an application event. If this argument is not specified, all types are matched. If you specify this argument, you must choose an integer between 1 and 4294967295, inclusive. There must be a match of component and type between the **event_publish** command extension and the **event_register_appl** command extension in order for the publishing and registration to work.

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

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

Result String

None

Set _cerrno

No

event_register_cli

Registers for a CLI event. Use this Tcl command extension to run a policy when a CLI command of a specific pattern is entered based on pattern matching performed against an expanded CLI command.

Note The user can enter an abbreviated CLI command, such as **sh mem summary**, and the parser will expand the command to **show memory summary** to perform the matching.

Note The functionality provided in the CLI event detector only allows a regular expression pattern match on a valid IOS CLI command itself. This does not include text after a pipe character when redirection is used.

Syntax

```
event_register_cli [tag ?] sync yes|no skip yes|no
[occurs ?] [period ?] pattern ? [default ?] [enter] [questionmark] [tab] [mode]
[queue_priority low|normal|high|last] [maxrun ?] [nice 0|1]
```

Arguments

tag	(Optional) String identifying a tag that can be used with the trigger Tcl command extension to support multiple event statements within a Tcl script.
sync	(Mandatory) A “yes” means that the policy (the event publish) will run synchronously with the CLI command; a “no” means that the event publish will be performed asynchronously with the CLI command. The event detector will be notified when the policy completes running. The exit status of the policy indicates whether or not the CLI command should be executed: if the exit status is zero, which means that the policy is executed successfully, the CLI command will not be executed; otherwise, the CLI command will be executed.
skip	Mandatory if the sync argument is “no” and should not exist if the sync argument is “yes.” If the skip argument is “yes,” it means that the CLI command should not be executed. If the skip argument is “no,” it means that the CLI command should be executed. **Caution** When the skip argument is “yes,” unintended results may be produced if the pattern match is made for configuration commands because the CLI command that matches the regular expression will not be executed.
occurs	(Optional) The number of occurrences before the event is raised. If this argument is not specified, the event is raised on the first occurrence. If this argument is specified, it must be an integer between 1 and 4294967295, inclusive.

period	(Optional) Specifies a backward looking time window in which all CLI events must occur (the occurs clause must be satisfied) in order for an event to be published (specified in SSSSSSSSSS[.MMM] format, where SSSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the most recent event is used.
pattern	(Mandatory) Specifies the regular expression used to perform the CLI command pattern match.
default	(Optional) The time period during which the CLI event detector waits for the policy to exit (specified in SSSSSSSSSS[.MMM] format, where SSSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If the default time period expires before the policy exits, the default action will be executed. The default action is to run the command. If this argument is not specified, the default time period is set to 30 seconds.
queue_priority	(Optional) Priority level at which the script will be queued: • queue_priority low—Specifies that the script is to be queued at the lowest of the three priority levels. • queue_priority normal—Specifies that the script is to be queued at a priority level greater than low priority but less than high priority. • queue_priority high—Specifies that the script is to be queued at the highest of the three priority levels. • queue_priority last—Specifies that the script is to be queued at the lowest priority level. If more than one script is registered with the "queue_priority_last" argument set, these scripts will execute in the order in which the events are published. **Note** The queue_priority argument specifies the queuing priority, but not the execution priority, of the script being registered. If this argument is not specified, the default queuing priority is normal.
enter	(Optional) Specifies to perform the event match when the user presses the Enter key. When this parameter is used, the input string will not be expanded before matching.
questionmark	(Optional) Specifies to perform the event match when the user presses the ? key. When this parameter is used, the input string will not be expanded before matching.
tab	(Optional) Specifies to perform the event match when the user presses the Tab key. When this parameter is used, the input string will not be expanded before matching.

mode	(Optional) Events will only be generated when the parser is in the specified parser mode. The available modes can be listed using the **show parser dump** CLI command. The mode parameter is checked when any one of the optional parameters—enter, questionmark, or tab— is specified.
maxrun	(Optional) Maximum run time of the script (specified in SSSSSSSSSS[.MMM] format, where SSSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the default 20-second run-time limit is used.
nice	(Optional) Policy run-time priority setting. When the nice argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.

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

Result String

None

Set _cerrno

No

Note This policy runs before the CLI command is executed. For example, suppose policy_CLI is registered to run when the **copy** command is entered. When the **copy** command is entered, the CLI event detector finds a pattern match and triggers this policy to run. When the policy execution ends, the CLI event detector determines if the **copy** command needs to be executed according to “sync”, “skip” (set in the policy), and the exit status of the policy execution if needed.

event_register_counter

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

Syntax

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

Arguments

tag	(Optional) String identifying a tag that can be used with the trigger Tcl command extension to support multiple event statements within a Tcl script.
name	(Mandatory) Name of the counter.
entry_op	(Mandatory) Entry comparison operator used to compare the current counter value with the entry value; if true, an event will be raised and event monitoring will be disabled until exit criteria are met.
entry_val	(Mandatory) Value with which the current counter value should be compared to decide if the counter event should be raised.
exit_op	(Mandatory) Exit comparison operator used to compare the current counter value with the exit value; if true, event monitoring for this event will be reenabled.
exit_val	(Mandatory) Value with which the current counter value should be compared to decide if the exit criteria are met.

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

Result String

None

Set _cerrno

No

event_register_gold

Registers for a Generic Online Diagnostic (GOLD) failure event. Use this Tcl command extension to run a policy on the basis of a Generic Online Diagnostic (GOLD) failure event for the specified card and subcard.

Syntax

```
event_register_gold card all|card_number
[subcard all|subcard_number]
[new_failure TRUE|FALSE]
[severity_major TRUE]
[severity_minor TRUE]
[severity_normal TRUE]
[action_notify TRUE|FALSE]
[testing_type [bootup|ondemand|schedule|monitoring]]
[test_name [testname]]
[test_id [testnumber]]
[consecutive_failure consecutive_failure_number]
[platform_action [action_flag]]
[maxrun ?]
[queue_priority low|normal|high|last]
[nice 0|1]
```

Arguments

card	(Mandatory) Specifies whether all cards or one card is to be monitored: • card all—Specifies that all cards are to be monitored. This is the default. • card-number—Specifies that the card identified by the number card-number is to be monitored. This argument must be specified to complete the **event_register_gold** Tcl command extension.
subcard	(Optional) Specifies that one or more subcards are to be monitored: • subcard all—Specifies that all subcards are to be monitored. • subcard-number—Specifies that the subcard identified by the number subcard-number is to be monitored. If this argument is not specified, all subcards are monitored by default.
new_failure	(Optional) Specifies event criteria based on the new test failure information from GOLD: • new_failure TRUE—Specifies that the event criterion for the new test failure is true from GOLD. • new_failure FALSE—Specifies that the event criterion for the new test failure is false from GOLD. If this argument is not specified, the new test failure information from GOLD is not considered in the event criteria.
severity_major	(Optional) Specifies that the event criteria for diagnostic result matches with the diagnostic major error from GOLD.

severity_minor	(Optional) Specifies that the event criteria for diagnostic result matches with diagnostic minor error from GOLD.
severity_normal	(Optional) Specifies that the event criteria for diagnostic result matches with diagnostic normal from GOLD. This is the default.
action_notify	(Optional) Specifies the event criteria based on the action notify information from GOLD: • action_notify TRUE—Specifies that the event criterion for the action notify is true from GOLD. • action_notify FALSE—Specifies that the event criterion for the action notify is false from GOLD. If this argument is not specified, the action notify information from GOLD is not considered in the event criteria.
testing_type	(Optional) Specifies the event criteria based on the testing types of the diagnostic from GOLD: • testing_type bootup—Specifies the diagnostic tests that are running on system bootup. • testing_type ondemand—Specifies the diagnostic tests that are running from CLI after the card is online. • testing_type schedule—Specifies the scheduled diagnostic tests. • testing_type monitoring—Specifies the diagnostic tests that are running periodically in the background to monitor the health of the system. If this argument is not specified, the testing type information from GOLD is not considered in the event criteria and the policy applies to all the diagnostic testing types.
test_name	(Optional) Specifies the event criteria based on the test name: • test_name test-name—Specifies the event criteria based on the test with the name test-name. If this argument is not specified, the test name information from GOLD is not considered in the event criteria.
test_id	(Optional) Specifies the event criteria based on test ID: • test_id test-id—Specifies the event criteria based on the test with the ID number test-id. The maximum value of test-id is 65535. **Note** Because the test ID can be different for the same test on different line cards, usually the test_name keyword should be used instead. If the test ID is specified and conflicts with the specified test name, the test name overwrites the test ID. If this argument is not specified, test ID information from GOLD is not considered in the event criteria.

consecutive_failure	(Optional) Specifies the event criteria based on consecutive test failure information from GOLD: • consecutive_failure consecutive-failure-number—Specifies that the event criterion is based on the occurrence of consecutive-failure-number consecutive test failures. If this argument is not specified, consecutive test failure information from GOLD is not considered in the event criteria.
platform_action	(Optional) Specifies whether callback to the platform is needed when all the event criteria are matched. When callback is needed, the platform needs to register a callback function through the provided registry. • platform_action action-flag-number—Specifies that, when callback to the platform is needed, specific information is specified by the platform-specific action-flag-number value. The maximum value of action-flag-number is 65535. **Note** It is up to the platform to determine what action needs to be taken based on the flag. If this argument is not specified, there is no callback.
maxrun	(Optional) Specifies the maximum runt time of the script. • maxrun max-run-time-number—Specifies that the maximum run time of the script is max-run-time-number seconds. The maximum value of max-run-time-number is 4294967295 seconds. If this argument is not specified, the default run time is 20 seconds.

queue_priority	(Optional) Priority level at which the script will be queued: • queue_priority low—Specifies that the script is to be queued at the lowest of the three priority levels. • queue_priority normal—Specifies that the script is to be queued at a priority level greater than low priority but less than high priority. • queue_priority high—Specifies that the script is to be queued at the highest of the three priority levels. • queue_priority last—Specifies that the script is to be queued at the lowest priority level. If more than one script is registered with the "queue_priority_last" argument set, these scripts will execute in the order in which the events are published. **Note** The queue_priority argument specifies the queuing priority, but not the execution priority, of the script being registered. If this argument is not specified, the default queuing priority is normal.
nice	(Optional) Policy run-time priority setting: • nice 0—Specifies that the policy is run at the default run-time priority level. • nice 1—Specifies that the policy is run at a run-time priority that is less than the default priority. If this argument is not specified, the default run-time priority is used.

Result String

None

Set _cerrno

No

event_register_interface

Registers for an interface counter event. Use this Tcl command extension to generate an event when specified interface counters exceed specified thresholds.

Syntax

```
event_register_interface [tag ?] name ?
parameter ? entry_op gt|ge|eq|ne|lt|le
entry_val ? entry_val_is_increment TRUE|FALSE
entry_type value|increment|rate
[exit_comb or|and]
[exit_op gt|ge|eq|ne|lt|le]
[exit_val ?] [exit_val_is_increment TRUE|FALSE]
[exit_type value|increment|rate]
[exit_time ?] [poll_interval ?]
[average_factor ?] [queue_priority low|normal|high|last]
[maxrun ?] [nice 0|1]
```

Arguments

tag	(Optional) String identifying a tag that can be used with the trigger Tcl command extension to support multiple event statements within a Tcl script.
name	(Mandatory) The name of the interface being monitored, for example, Ethernet 0/0. Abbreviations and spaces are not allowed.
parameter	(Mandatory) The name of the counter being compared as follows: • input_errors—Includes runts, giants, no buffer, CRC, frame, overrun, and ignored counts. Other input-related errors can also cause the input errors count to be increased, and some datagrams may have more than one error; therefore, this sum may not balance with the sum of enumerated input error counts. • input_errors_crc—Cyclic redundancy checksum generated by the originating LAN station or far-end device does not match the checksum calculated from the data received. • input_errors_frame—Number of packets received incorrectly having a CRC error and a noninteger number of octets. • input_errors_overrun—Number of times the receiver hardware was unable to hand received data to a hardware buffer because the input rate exceeded the receiver's ability to handle the data. • input_packets_dropped—Number of packets dropped because of a full input queue. • interface_resets—Number of times that an interface has been completely reset. • output_buffer_failures—Number of failed buffers and number of buffers swapped out. • output_buffer_swappedout—Number of packets swapped to DRAM.

parameter (continued)	• output_errors—Sum of all errors that prevented the final transmission of datagrams out of the interface being examined. Note that this may not balance with the sum of the enumerated output errors, because some datagrams may have more than one error, and others may have errors that do not fall into any of the specifically tabulated categories. • output_errors_underrun—Number of times that the transmitter has been running faster than the router can handle. • output_packets_dropped—Number of packets dropped because of a full output queue. • receive_broadcasts—Number of broadcast or multicast packets received by the interface. • receive_giants—Number of packets that are discarded because they exceed the maximum packet size of the medium. • receive_rate_bps—Interface receive rate in bytes per second. • receive_rate_pps—Interface receive rate in packets per second. • receive_runts—Number of packets that are discarded because they are smaller than the minimum packet size of the medium. • receive_throttle—Number of times that the receiver on the port was disabled, possibly because of buffer or processor overload. • reliability—Reliability of the interface as a fraction of 255 (255/255 is 100 percent reliability), calculated as an exponential average over 5 minutes. • rxload—Receive rate of the interface as a fraction of 255 (255/255 is 100 percent). • transmit_rate_bps—Interface transmit rate in bytes per second. • transmit_rate_pps—Interface transmit rate in packets per second. • txload—Transmit rate of the interface as a fraction of 255 (255/255 is 100 percent).
entry_op	(Mandatory) The comparison operator used to compare the current interface value with the entry value; if true, an event will be raised and event monitoring will be disabled until exit criteria are met.
entry_val	(Mandatory) The value at which the event will be triggered.
entry_val_is_increment	(Mandatory) If TRUE, the entry_val field is treated as an incremental difference and is compared with the difference between the current counter value and the value when the event was last true (the first polled sample if this is a new event). A negative value checks the incremental difference for a counter that is decreasing. If FALSE, the entry_val field is compared against the current counter value. **Note** In Cisco IOS Release 12.4(20)T, this keyword is deprecated, and if specified, the syntax is converted into equivalent entry-type keyword syntax.

entry-type	Specifies a type of operation to be applied to the object ID specified by the entry-val argument. Value is defined as the actual value of the entry-val argument. Increment uses the entry-val field as an incremental difference and the entry-val is compared with the difference between the current counter value and the value when the event was last triggered (or the first polled sample if this is a new event). A negative value checks the incremental difference for a counter that is decreasing. Rate is defined as the average rate of change over a period of time. The time period is the average-factor value multiplied by the poll-interval value. At each poll interval the difference between the current sample and the previous sample is taken and recorded as an absolute value. An average of the previous average-factor value samples is taken to be the rate of change.
exit_comb	(Optional) Used to indicate the combination of exit condition tests required to rearm the event trigger; if the and operator is specified, both exit value and exit time tests must be true to cause rearm; if the or operator is specified, either exit value or exit time tests can be true to cause event monitoring to be rearmed.
exit_op	(Optional) The comparison operator used to compare the current interface value with the exit value; if true, event monitoring for this event will be reenabled.
exit_val	(Optional) The value at which the event is rearmed to be monitored again.
exit_val_is_increment	(Optional) If TRUE, the exit_val field is treated as an incremental difference and is compared with the difference between the current counter value and the value when the event was last true. A negative value checks the incremental difference for a counter that is decreasing. If FALSE, the exit_val field is compared against the current counter value. **Note** In Cisco IOS Release 12.4(20)T, this keyword is deprecated, and if specified, the syntax is converted into equivalent exit-type keyword syntax.
exit-type	(Optional) Specifies a type of operation to be applied to the object ID specified by the exit-val argument. If not specified, the value is assumed. Value is defined as the actual value of the exit-val argument. Increment uses the exit-val field as an incremental difference and the exit-val is compared with the difference between the current counter value and the value when the event was last triggered (or the first polled sample if this is a new event). A negative value checks the incremental difference for a counter that is decreasing. Rate is defined as the average rate of change over a period of time. The time period is the average-factor value multiplied by the poll-interval value. At each poll interval the difference between the current sample and the previous sample is taken and recorded as an absolute value. An average of the previous average-factor value samples is taken to be the rate of change.

exit_time	(Optional) The time period at which the event is rearmed to be monitored again (specified in SSSSSSSSSS[.MMM] format, where SSSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999).
poll_interval	(Optional) The frequency used to collect the samples (specified in SSSSSSSSSS[.MMM] format, where SSSSSSSSSS must be an integer representing seconds between 60 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). The poll interval value must not be less than 1 second. The default is 1 second.
average-factor	(Optional) Number in the range from 1 to 64 used to calculate the period used for rate-based calculations. The average-factor value is multiplied by the poll-interval value to derive the period in milliseconds. The minimum average factor value is 1.
queue_priority	(Optional) Priority level at which the script will be queued: • queue_priority low—Specifies that the script is to be queued at the lowest of the three priority levels. • queue_priority normal—Specifies that the script is to be queued at a priority level greater than low priority but less than high priority. • queue_priority high—Specifies that the script is to be queued at the highest of the three priority levels. • queue_priority last—Specifies that the script is to be queued at the lowest priority level. If more than one script is registered with the "queue_priority_last" argument set, these scripts will execute in the order in which the events are published. **Note** The queue_priority argument specifies the queuing priority, but not the execution priority, of the script being registered. If this argument is not specified, the default queuing priority is normal.
maxrun	(Optional) Maximum run time of the script (specified in SSSSSSSSSS[.MMM] format, where SSSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the default 20-second run-time limit is used.
nice	(Optional) Policy run-time priority setting. When the nice argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.

Result String

None

Set _cerrno

No

event_register_ioswdsysmon

Registers for an IOSWDSysMon event. Use this Tcl command extension to generate an event when a Cisco IOS task exceeds specific CPU utilization or memory thresholds. A Cisco IOS task is called a Cisco IOS process in native Cisco IOS.

Syntax

```
event_register_ioswdsysmon [tag ?] [timewin ?] [sub12op and|or] [sub1 ?] [sub2 ?]
[queue_priority low|normal|high|last] [maxrun ?] [nice 0|1]
```

Arguments

tag	(Optional) String identifying a tag that can be used with the trigger Tcl command extension to support multiple event statements within a Tcl script.
timewin	(Optional) Defines the time window within which all of the subevents must occur in order for an event to be generated (specified in SSSSSSSSSS[.MMM] format, where SSSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999).
sub12_op	(Optional) The combination operator for comparison between subevent 1 and subevent 2.
sub1	(Optional) The subevent 1 specification.
sub2	(Optional) The subevent 2 specification.
queue_priority	(Optional) Priority level at which the script will be queued: • queue_priority low—Specifies that the script is to be queued at the lowest of the three priority levels. • queue_priority normal—Specifies that the script is to be queued at a priority level greater than low priority but less than high priority. • queue_priority high—Specifies that the script is to be queued at the highest of the three priority levels. • queue_priority last—Specifies that the script is to be queued at the lowest priority level. If more than one script is registered with the "queue_priority_last" argument set, these scripts will execute in the order in which the events are published. **Note** The queue_priority argument specifies the queuing priority, but not the execution priority, of the script being registered. If this argument is not specified, the default queuing priority is normal.

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

Subevent Syntax

```
cpu_proc path ? taskname ? op gt|ge|eq|ne|lt|le val ? [period ?]

mem_proc path ? taskname ? op gt|ge|eq|ne|lt|le val ? [is_percent TRUE|FALSE] [period ?]
```

Subevent Arguments

cpu_proc	(Mandatory) Specifies the use of a sample collection of CPU statistics.
path	(Mandatory) Software Modularity images only. The pathname of the POSIX process that contains the Cisco IOS scheduler to be monitored. For example, /sbin/cdp2.iosproc.
taskname	(Mandatory) The name of the Cisco IOS task to be monitored.
op	(Mandatory) The comparison operator used to compare the collected usage sample with the specified value; if true, an event will be raised.
val	(Mandatory) The value to be compared.
period	(Optional) The elapsed time period for the collection samples to be averaged (specified in SSSSSSSSSS[.MMM] format, where SSSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the most recent sample is used.
mem_proc	(Mandatory) Specifies the use of a sample collection of memory statistics.
is_percent	(Optional) Whether the specified value is a percentage.

Result String

None

Set _cerrno

No

event_register_ipsla

Registers for an event that is triggered by the **event ipsla** command. Use this Tcl command to publish an event when an IPSLA reaction is triggered. The group ID or the operation ID is required to register the event.

Syntax

```
event_register_ipsla [tag ?] group_name ? operation_id ? [reaction_type ?]
[dest_ip_addr ?][queue_priority low|normal|high|last] [maxrun ?] [nice 0|1]
```

Arguments

tag	(Optional) String identifying a tag that can be used with the trigger Tcl command extension to support multiple event statements within a Tcl script.
group_name	(Mandatory) Specifies the IP SLAs group name.
operation_id	(Mandatory) Specifies the IP SLA operation ID. Number must be in the range from 1 to 2147483647.

reaction_type	(Optional) Specifies the reaction to be taken for the specified IP SLAs operation. Type of IP SLAs reaction—One of the following keywords can be specified: **connectionLoss**, **icpif**, **jitterAvg**, **jitterDSAvg**, **jitterSDAvg**, **maxOfNegativeDS**, **maxOfNegativeSD**, **maxOfPositiveDS**, **maxOfPositiveSD**, **mos**, **packetLateArrival**, **packetLossDS**, **packetLossSD**, **packetMIA**, **packetOutOfSequence**, **rtt**, **timeout** or **verifyError** can be specified. Type of IP SLAs reaction. One of the following keywords can be specified: • connectionLoss • icpif • jitterAvg • jitterDSAvg • jitterSDAvg • maxOfNegativeDS • maxOfNegativeSD • maxOfPositiveDS • maxOfPositiveSD • mos • packetLateArrival • packetLossDS • packetLossSD • packetMIA • packetOutOfSequence • rtt • timeout • verifyError
dest_ip_address	(Optional) Specifies the destination IP address of the destination port for which the IP SLAs events are monitored.

queue_priority	(Optional) Priority level at which the script will be queued: • queue_priority low—Specifies that the script is to be queued at the lowest of the three priority levels. • queue_priority normal—Specifies that the script is to be queued at a priority level greater than low priority but less than high priority. • queue_priority high—Specifies that the script is to be queued at the highest of the three priority levels. • queue_priority last—Specifies that the script is to be queued at the lowest priority level. If more than one script is registered with the "queue_priority_last" argument set, these scripts will execute in the order in which the events are published. **Note** The queue_priority argument specifies the queuing priority, but not the execution priority, of the script being registered. If this argument is not specified, the default queuing priority is normal.
maxrun	(Optional) Maximum run time of the script (specified in SSSSSSSSSS[.MMM] format, where SSSSSSSSSS must be an integer representing seconds between 0 and 31536000, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the default 20-second run-time limit is used.
nice	(Optional) Policy run-time priority setting. When the nice argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.

Result String

None

Set _cerrno

No

event_register_nf

Registers for an event when a NetFlow event is triggered by the **event nf** command. Use this Tcl command to publish an event when an NetFlow reaction is triggered..

Syntax

```
event_register_nf [tag ?] monitor_name ? event_type create|update|delete
exit_event_type create|update|delete event1-event4 ? [maxrun ?] [nice 0|1]
```

Arguments

tag	(Optional) String identifying a tag that can be used with the trigger Tcl command extension to support multiple event statements within a Tcl script.
monitor_name	(Mandatory) The name of the NetFlow monitor.
event_type	(Mandatory) The type of event to monitor for the create, update, and delete flow.
exit_event_type	(Mandatory) The event-type (create, delete, update) at which the event is rearmed to be monitored again.
event1- event4	(Mandatory) Specifies the event and its attributes to monitor. Valid values are **event1**, **event2**, **event3**, and **event4**. The subevent keywords can be used alone, together, or in any combination with each other, but each keyword can be used only once.
maxrun	(Optional) Maximum run time of the script (specified in SSSSSSSSSS[.MMM] format, where SSSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the default 20-second run-time limit is used.
nice	(Optional) Policy run-time priority setting. When the nice argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.

Subevent Syntax

```
field ? rate_interval ? event1 only entry_value ? entry_op eq|ge|gt|le|lt|wc
[exit_value ?] [exit_op eq|ge|gt|le|lt|wc] [exit_rate_interval ? event1 only]
```

Subevent Arguments

field	(Mandatory) Specifies the cache or field attribute to be monitored. One of the following attributes can be specified: • **counter** {**bytes** | **packets**}—Specifies the counter fields. • **datalink** {**dot1q** | **mac**}—Specifies the datalink (layer2) fields. • **flow** {**direction** | **sampler**}—Specifies the flow identifying fields. • **interface** {**input** | **output**}—Specifies the interface fields. • **ipv4** *field-type*—Specifies the IPv4 fields. • **ipv6** *field-type*—IPv6 fields • **routing** *routing-attrribute***—**Specifies the routing attributes. • **timestamp sysuptime** {**first** | **last**}**—**Specifies the timestamp fields. • **transport** *field-type*—Specifies the Transport layer fields.
rate_interval	(Mandatory) Specifies the rate interval value in seconds used to calculate the rate. This field is only valid for event1.
entry_value	(Mandatory) Specifies the field or rate value.
entry_op	(Mandatory) Specifies the field operator. The comparison operator valid values are: • **eq** - Equal to • **ge** - Greater than or equal to • **gt** - Greater than • **le** - Less than or equal to • **lt** - Less than • **wc** - Wildcard
exit_value	(Optional) The value at which the event is rearmed to be monitored again.

exit_op	(Optional) The comparison operator used to compare the current event field or rate value with the exit value; if true, event monitoring for this event is reenabled. The comparison operator valid values are: • **eq** - Equal to • **ge** - Greater than or equal to • **gt** - Greater than • **le** - Less than or equal to • **lt** - Less than • **wc** - Wildcard
exit_rate_interval	(Optional) Specifies the exit rate interval value in seconds used to calculate the exit rate value. This field is only valid for event1.

Result String

None

Set _cerrno

No

event_register_none

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

Syntax

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

Arguments

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

Result String

None

Set _cerrno

No

event_register_oir

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

Syntax

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

Arguments

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

Result String

None

Set _cerrno

No

event_register_process

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

Syntax

```
event_register_process [tag ?] abort|term|start|user_restart|user_shutdown
[sub_system ?] [version ?] [instance ?] [path ?] [node ?]
[queue_priority low|normal|high|last] [maxrun ?] [nice 0|1]
```

Arguments

tag	(Optional) String identifying a tag that can be used with the trigger Tcl command extension to support multiple event statements within a Tcl script.
abort	(Mandatory) Abnormal process termination. Process may abort because of exiting with a nonzero exit status, receiving a kernel-generated signal, or receiving a SIGTERM or SIGKILL signal that is not sent because of user request.
term	(Mandatory) Normal process termination.
start	(Mandatory) Process start.
user_restart	(Mandatory) Process termination due to the process restart request from the CLI command.
user_shutdown	(Mandatory) Process termination due to the process kill request from the CLI command.
sub_system	(Optional) Number assigned to the EEM policy that published the process event. Number is set to 798 because all other numbers are reserved for Cisco use.
version	(Optional) Version number of the process assigned by the version manager. Must be of the form major_number.minor_number.level. If specified, each component of the version number must be an integer between 1 and 4294967295, inclusive.
instance	(Optional) Process instance ID. If specified, this argument must be an integer between 1 and 4294967295, inclusive.
path	(Optional) Process pathname (a regular expression string). If the value of the process-name argument contains embedded blanks, enclose it in double quotation marks. Use path ".*" to match all processes.

node	(Optional) The node name is a string that consists of the word “node” followed by two fields separated by a slash character using the following format: node<slot-number>/<cpu-number> The slot-number is the hardware slot number. The cpu-number is the hardware CPU number. For example, the SP CPU in a Supervisor card on a Cisco Catalyst 6500 series switch located in slot 0 would be specified as node0/0. The RP CPU in a Supervisor card on a Cisco Catalyst 6500 series switch located in slot 0 would be addressed as node0/1. If the node argument is not specified, the default node specification is always the regular expression pattern match of * representing all applicable nodes.
queue_priority	(Optional) Priority level at which the script will be queued: • queue_priority low—Specifies that the script is to be queued at the lowest of the three priority levels. • queue_priority normal—Specifies that the script is to be queued at a priority level greater than low priority but less than high priority. • queue_priority high—Specifies that the script is to be queued at the highest of the three priority levels. • queue_priority last—Specifies that the script is to be queued at the lowest priority level. If more than one script is registered with the “queue_priority_last” argument set, these scripts will execute in the order in which the events are published. **Note** The queue_priority argument specifies the queuing priority, but not the execution priority, of the script being registered. If this argument is not specified, the default queuing priority is normal.
maxrun	(Optional) Maximum run time of the script (specified in SSSSSSSSSS[.MMM] format, where SSSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the default 20-second run-time limit is used.
nice	(Optional) Policy run-time priority setting. When the nice argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.

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

Result String

None

Set _cerrno

No

event_register_resource

Registers for an Embedded Resource Manager (ERM) event. Use this Tcl command extension to run a policy on the basis of an ERM event report for a specified policy. ERM events are screened by the EEM Resource event detector, allowing an EEM policy to be run when a match occurs for the specified ERM policy.

Syntax

```
event_register_resource policy policy-name [queue_priority low|normal|high|last]
[maxrun ?] [nice 0|1]
```

Arguments

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

Result String

None

Set _cerrno

No

event_register_rf

Registers for a Redundancy Facility (RF) event. Use this Tcl command extension to run a policy when an RF progression or status event notification occurs.

Syntax

```
event_register_rf [tag ?] event ?
[queue_priority low|normal|high|last]
[maxrun ?] [nice 0|1]
```

Arguments

tag	(Optional) String identifying a tag that can be used with the trigger Tcl command extension to support multiple event statements within a Tcl script.
event	(Mandatory) Name of the RF progression or status event. Valid values are: • RF_PROG_ACTIVE • RF_PROG_ACTIVE_DRAIN • RF_PROG_ACTIVE_FAST = 200 • RF_PROG_ACTIVE_PRECONFIG • RF_PROG_ACTIVE_POSTCONFIG • RF_PROG_EXTRALOAD • RF_PROG_HANDBACK • RF_PROG_INITIALIZATION • RF_PROG_PLATFORM_SYNC • RF_PROG_STANDBY_BULK • RF_PROG_STANDBY_COLD • RF_PROG_STANDBY_CONFIG • RF_PROG_STANDBY_FILESYS • RF_PROG_STANDBY_HOT • RF_PROG_STANDBY_OIR_SYNC_DONE • RF_REGISTRATION_STATUS • RF_STATUS_MAINTENANCE_ENABLE • RF_STATUS_MANUAL_SWACT • RF_STATUS_OPER_REDUNDANCY_MODE_CHANGE • RF_STATUS_PEER_COMM • RF_STATUS_PEER_PRESENCE • RF_STATUS_REDUNDANCY_MODE_CHANGE • RF_STATUS_SWACT_INHIBIT

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

Result String

None

Set _cerrno

No

event_register_routing

Registers for an event that is triggered by the **event routing** command. These events are handled by the routing event detector to publish an event when route entries change in Routing Information Base (RIB) infrastructure. Use this Tcl command extension to run a routing policy for this script. The network IP address for the route to be monitored must be specified.

Syntax

```
event_register_routing [tag ?] network ? length [ge|le|ne] [type add|remove|modify|all]
[protocol ?] [queue_priority normal|low|high|last] [maxrun ?] [nice {0 | 1}]
```

Arguments

tag	(Optional) String identifying a tag that can be used with the trigger Tcl command extension to support multiple event statements within a Tcl script.
network	Specifies the network IP address. The network number can be any valid IP address or prefix.
length	Specifies the length of the network mask in bits. The bit mask can be a number from 0 to 32. • **ge**—(Optional) Specifies the minimum prefix length to be matched. The **ge** keyword represents greater than or equal to operator. • **le**—(Optional) Specifies the maximum prefix length to be matched. The **le** keyword represents the less than or equal to operator. • **ne**—(Optional) Specifies the prefix length not to be matched. The **ne** keyword represents not equal to operator. When **ge**, **le** and **ne** keywords are not configured, an exact match of network length is processed.
type	(Optional) Specifies the desired policy trigger. The type options are **add**, **remove**, **modify**, and **all**. The default is **all**.
protocol	(Optional) Specifies the protocol value for the network being monitored. One of the following protocols can be used: **all**, **bgp**, **connected**, **eigrp**, **isis**, **iso-igrp**, **mobile**, **odr**, **ospf**, **rip**, and **static**. The default is **all**.

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

Result String

None

Set _cerrno

No

event_register_rpc

Registers for an event that is triggered by the EEM SSH Remote Procedure Call (RPC) command. These events are handled by the RPC event detector that screens for this event. Use this Tcl command extension to run a RPC policy for this script.

Syntax

```
event_register_rpc [queue_priority {normal | low | high | last}] [maxrun <sec.msec>] [nice
{0 | 1}] [default <sec.msec>]
```

Arguments

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

nice	(Optional) Policy run-time priority setting. When the nice argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.
default	(Optional) The time period during which the CLI event detector waits for the policy to exit (specified in SSSSSSSSSS[.MMM] format, where SSSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If the default time period expires before the policy exits, the default action will be executed. The default action is to run the command. If this argument is not specified, the default time period is set to 30 seconds.

Result String

None

Set _cerrno

No

event_register_snmp

Registers for a Simple Network Management Protocol (SNMP) statistics event. Use this Tcl command extension to run a policy when a given counter specified by an SNMP object ID (oid) crosses a defined threshold.

Syntax

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

Arguments

tag	(Optional) String identifying a tag that can be used with the trigger Tcl command extension to support multiple event statements within a Tcl script.
oid	(Mandatory) OID number of data element in SNMP dot notation (for example, 1.3.6.1.2.1.2.1.0). The types of OIDs allowed are: • COUNTER_TYPE • COUNTER_64_TYPE • GAUGE_TYPE • INTEGER_TYPE • OCTET_PRIM_TYPE • OPAQUE_PRIM_TYPE • TIME_TICKS_TYPE
entry_op	(Mandatory) Entry comparison operator used to compare the current OID data value with the entry value; if true, an event will be raised and event monitoring will be disabled until exit criteria are met.
get_type	(Mandatory) Type of SNMP get operation that needs to be applied to the OID specified. If the get_type argument is “exact,” the value of the specified OID is retrieved; if the get_type argument is “next,” the value of the lexicographical successor to the specified OID is retrieved.
entry_val	(Mandatory) Value with which the current oid data value should be compared to decide if the SNMP event should be raised.

entry-type	Specifies a type of operation to be applied to the object ID specified by the entry-val argument. Value is defined as the actual value of the entry-val argument. Increment uses the entry-val field as an incremental difference and the entry-val is compared with the difference between the current counter value and the value when the event was last triggered (or the first polled sample if this is a new event). A negative value checks the incremental difference for a counter that is decreasing. Rate is defined as the average rate of change over a period of time. The time period is the average-factor value multiplied by the poll-interval value. At each poll interval the difference between the current sample and the previous sample is taken and recorded as an absolute value. An average of the previous average-factor value samples is taken to be the rate of change.
exit_comb	(Optional) Exit combination operator used to indicate the combination of exit condition tests required to decide if the exit criteria are met so that the event monitoring can be reenabled. If it is "and," both exit value and exit time tests must be passed to meet the exit criteria. If it is "or," either exit value or exit time tests can be passed to meet the exit criteria. When exit_comb is "and," exit_op, and exit_val (exit_time) must exist. When exit_comb is "or," (exit_op and exit_val) or (exit_time) must exist.
exit_op	(Optional) Exit comparison operator used to compare the current oid data value with the exit value; if true, event monitoring for this event will be reenabled.
exit_val	(Optional) Value with which the current oid data value should be compared to decide if the exit criteria are met.
exit-type	(Optional) Specifies a type of operation to be applied to the object ID specified by the exit-val argument. If not specified, the value is assumed. Value is defined as the actual value of the exit-val argument. Increment uses the exit-val field as an incremental difference and the exit-val is compared with the difference between the current counter value and the value when the event was last triggered (or the first polled sample if this is a new event). A negative value checks the incremental difference for a counter that is decreasing. Rate is defined as the average rate of change over a period of time. The time period is the average-factor value multiplied by the poll-interval value. At each poll interval the difference between the current sample and the previous sample is taken and recorded as an absolute value. An average of the previous average-factor value samples is taken to be the rate of change.
exit_time	(Optional) Number of POSIX timer units after an event is raised when event monitoring will be enabled again. Specified in SSSSSSSSSS[.MMM] format where SSSSSSSSSS must be an integer number representing seconds between 0 and 4294967295, inclusive. MMM represents milliseconds and must be an integer number between 0 and 999.

poll_interval	(Mandatory) Interval between consecutive polls in POSIX timer units. Currently the interval is forced to be at least 1 second (specified in SSSSSSSSSS[.MMM] format, where SSSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999).
average-factor	(Optional) Number in the range from 1 to 64 used to calculate the period used for rate-based calculations. The average-factor value is multiplied by the poll-interval value to derive the period in milliseconds. The minimum average factor value is 1.
queue_priority	(Optional) Priority level at which the script will be queued: • queue_priority low—Specifies that the script is to be queued at the lowest of the three priority levels. • queue_priority normal—Specifies that the script is to be queued at a priority level greater than low priority but less than high priority. • queue_priority high—Specifies that the script is to be queued at the highest of the three priority levels. • queue_priority last—Specifies that the script is to be queued at the lowest priority level. If more than one script is registered with the "queue_priority_last" argument set, these scripts will execute in the order in which the events are published. **Note** The queue_priority argument specifies the queuing priority, but not the execution priority, of the script being registered. If this argument is not specified, the default queuing priority is normal.
maxrun	(Optional) Maximum run time of the script (specified in SSSSSSSSSS[.MMM] format, where SSSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the default 20-second run-time limit is used.
nice	(Optional) Policy run-time priority setting. When the nice argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.

Result String

None

Set _cerrno

No

event_register_snmp_notification

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

Syntax

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

Arguments

tag	(Optional) String identifying a tag that can be used with the trigger Tcl command extension to support multiple event statements within a Tcl script.
oid	(Mandatory) OID number of the data element in SNMP dot notation (for example, 1.3.6.1.2.1.2.1.0). If the specified OID ends with a dot (.), then all OIDs that start with the OID number before the dot are matched. The types of OIDs allowed are: • COUNTER_TYPE • COUNTER_64_TYPE • GAUGE_TYPE • INTEGER_TYPE • OCTET_PRIM_TYPE • OPAQUE_PRIM_TYPE • TIME_TICKS_TYPE
oid_val	(Mandatory) OID value with which the current OID data value should be compared to decide if the SNMP event should be raised.
op	(Mandatory) Comparison operator used to compare the current OID data value with the SNMP Protocol Data Unit (PDU) OID data value; if this is true, an event is raised.
maxrun	(Optional) Maximum run time of the script (specified in ssssssss[.mmm] format, where ssssssss must be an integer representing seconds between 0 and 31536000, inclusive, and where mmm must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the default 20-second run-time limit is used.
src_ip_address	(Optional) Source IP address where the SNMP notification trap originates. The default is all; it is set to receive SNMP notification traps from all IP addresses.

dest_ip_address	(Optional) Destination IP address where the SNMP notification trap is sent. The default is all; it is set to receive SNMP traps from all destination IP addresses.
queue_priority	(Optional) Priority level at which the script will be queued: • queue_priority low—Specifies that the script is to be queued at the lowest of the three priority levels. • queue_priority normal—Specifies that the script is to be queued at a priority level greater than low priority but less than high priority. • queue_priority high—Specifies that the script is to be queued at the highest of the three priority levels. • queue_priority last—Specifies that the script is to be queued at the lowest priority level. If more than one script is registered with the queue_priority_last argument set, these scripts will execute in the order in which the events are published. **Note** The queue_priority argument specifies the queuing priority, but not the execution priority, of the script being registered. If this argument is not specified, the default queuing priority is normal.
default	(Optional) Specifies the time period in seconds during which the snmp notification event detector waits for the policy to exit. The time period is specified in sssssssss[.mmm] format, where sssssssss must be an integer representing seconds between 0 and 4294967295 and mmm must be an integer representing milliseconds between 0 and 999.
nice	(Optional) Policy run-time priority setting. When the nice argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.
direction	(Optional) The direction of the incoming or outgoing SNMP trap or inform PDU to filter. The default value is incoming.
msg_op	(Optional) The action to be taken on the SNMP PDU (drop it or send it) once the event is triggered. The default value is send.

Result String

None

Set _cerrno

No

event_register_snmp_object

Registers for a Simple Network Management Protocol (SNMP) object event. Use this Tcl command extension to replace the value when an SNMP with the specified SNMP-object ID (OID) is encountered on a specific interface or address.

Syntax

```
event_register_snmp_object oid ?
type {int|uint|counter|counter64|gauge|ipv4|octect|oid|string}
sync {yes|no}
skip {yes|no}
[istable {yes|no}]
[default ?]
[queue_priority {normal|low|high|last}]
[maxrun ?]
[nice {0|1}]
```

Arguments

oid	(Mandatory) OID number of the data element in SNMP dot notation (for example, 1.3.6.1.2.1.2.1.0). If the specified OID ends with a dot (.), then all OIDs that start with the OID number before the dot are matched. The types of OIDs allowed are: • COUNTER_TYPE • COUNTER_64_TYPE • GAUGE_TYPE • INTEGER_TYPE • OCTET_PRIM_TYPE • OPAQUE_PRIM_TYPE • TIME_TICKS_TYPE
type	(Mandatory) OID value type.
sync	(Mandatory) A “yes” means that the EEM policy will be notified. If the applet set_exit_status or Tcl return value is 0, then SNMP will handle the request. If the return value is 1, SNMP will use the value provided by the policy for the get request and will not process the set request. A “no” means that EEM will not be notified and SNMP will handle the request. Only one OID can be associated with a synchronous policy. However, multiple synchronous policies can be registered for the same OID.
skip	Mandatory if the sync argument is “no” and should not exist if the sync argument is “yes.” If the skip argument is “yes,” it means that SNMP will handle the request. If the skip argument is “no,” it means that SNMP will act as if the object does not exist.
istable	(Optional) A value of “no” means the OID is scalar object, and “yes” means the OID is table object.

default	(Optional) The time period during which the SNMP Object event detector waits for the policy to exit (specified in sssssssssss[.mmm] format, where sssssssssss must be an integer representing seconds between 0 and 4294967295, inclusive, and where mmm must be an integer representing milliseconds between 0 and 999). If the default time period expires before the policy exits, the default action will be executed. The default action is to process the set or get request normally by SNMP subsystem. If this argument is not specified, the default time period is set to 30 seconds.
maxrun	(Optional) Maximum run time of the script (specified in ssssssss[.mmm] format, where ssssssss must be an integer representing seconds between 0 and 31536000, inclusive, and where mmm must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the default 20-second run-time limit is used.
queue_priority	(Optional) Priority level at which the script will be queued: • queue_priority low—Specifies that the script is to be queued at the lowest of the three priority levels. • queue_priority normal—Specifies that the script is to be queued at a priority level greater than low priority but less than high priority. • queue_priority high—Specifies that the script is to be queued at the highest of the three priority levels. • queue_priority last—Specifies that the script is to be queued at the lowest priority level. If more than one script is registered with the queue_priority_last argument set, these scripts will execute in the order in which the events are published. **Note** The queue_priority argument specifies the queuing priority, but not the execution priority, of the script being registered. If this argument is not specified, the default queuing priority is normal.
nice	(Optional) Policy run-time priority setting. When the nice argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.

Result String

None

Set _cerrno

No

event_register_syslog

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

Syntax

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

Arguments

tag	(Optional) String identifying a tag that can be used with the trigger Tcl command extension to support multiple event statements within a Tcl script.
occurs	(Optional) Number of occurrences before the event is raised; if not specified, the event is raised on the first occurrence. If specified, the value must be greater than 0.
period	(Optional) Time interval, in seconds and milliseconds, during which the one or more occurrences must take place in order to raise an event (specified in SSSSSSSSSS[.MMM] format where SSSSSSSSSS must be an integer number representing seconds between 0 and 4294967295, inclusive, and where MMM represents milliseconds and must be an integer number between 0 and 999). If this argument is not specified, no period check is applied.
pattern	(Mandatory) A regular expression used to perform syslog message pattern match. This argument is what the policy uses to identify the logged syslog message.
priority	(Optional) The message priority to be screened. If this argument is specified, only messages that are at the specified logging priority level, or lower, are screened. If this argument is not specified, the default priority is 0.

queue_priority	(Optional) Priority level at which the script will be queued: • queue_priority low—Specifies that the script is to be queued at the lowest of the three priority levels. • queue_priority normal—Specifies that the script is to be queued at a priority level greater than low priority but less than high priority. • queue_priority high—Specifies that the script is to be queued at the highest of the three priority levels. • queue_priority last—Specifies that the script is to be queued at the lowest priority level. If more than one script is registered with the "queue_priority_last" argument set, these scripts will execute in the order in which the events are published. **Note** The queue_priority argument specifies the queuing priority, but not the execution priority, of the script being registered. If this argument is not specified, the default queuing priority is normal.
maxrun	(Optional) Maximum run time of the script (specified in SSSSSSSSSS[.MMM] format, where SSSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the default 20-second run-time limit is used.
nice	(Optional) Policy run-time priority setting. When the nice argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.
severity_xxx	(Optional) The event severity to be screened. If this argument is specified, only messages that are at the specified severity level are screened. See Table 13 for the severity level mapping for syslog events.

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

Table 13 Severity Level Mapping For Syslog Events

Severity Keyword	Syslog Priority	Description
severity_fatal	LOG_EMERG (0)	System is unusable.
severity_critical	LOG_ALERT (1)	Critical conditions, immediate attention required.
severity_major	LOG_CRIT (2)	Major conditions.
severity_minor	LOG_ERR (3)	Minor conditions.
severity_warning	LOG_WARNING (4)	Warning conditions.
severity_notification	LOG_NOTICE (5)	Basic notification, informational messages.

Table 13 Severity Level Mapping For Syslog Events

severity_normal	LOG_INFO (6)	Normal event, indicates returning to a normal state.
severity_debugging	LOG_DEBUG (7)	Debugging messages.

Result String

None

Set _cerrno

No

event_register_timer

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

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

Syntax

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

Arguments

tag	(Optional) String identifying a tag that can be used with the trigger Tcl command extension to support multiple event statements within a Tcl script.
watchdog	(Mandatory) Watchdog timer.
countdown	(Mandatory) Countdown timer.
absolute	(Mandatory) Absolute timer.
cron	(Mandatory) CRON timer.
name	(Optional) Name of the timer.

cron_entry	(Optional) Must be specified if the CRON timer type is specified. Must not be specified if any other timer type is specified. A cron_entry is a partial UNIX crontab entry (the first five fields) as used with the UNIX CRON daemon. A cron_entry specification consists of a text string with five fields. The fields are separated by spaces. The fields represent the time and date when CRON timer events will be triggered. The fields are described in Table 14. Ranges of numbers are allowed. Ranges are two numbers separated with a hyphen. The specified range is inclusive. For example, 8-11 for an hour entry specifies execution at hours 8, 9, 10, and 11. A field may be an asterisk (*), which always stands for “first-last.” Lists are allowed. A list is a set of numbers (or ranges) separated by commas. Examples: “1,2,5,9” and “0-4,8-12”. Step values can be used in conjunction with ranges. Following a range with “/<number>” specifies skips of the number’s value through the range. For example, “0-23/2” can be used in the hour field to specify an event that is triggered every other hour. Steps are also permitted after an asterisk, so if you want to say “every two hours”, use “*/2”. Names can also be used for the month and the day of week fields. Use the first three letters of the particular day or month (case does not matter). Ranges or lists of names are not allowed. The day on which a timer event is triggered can be specified by two fields: day of month and day of week. If both fields are restricted (that is, are not *), an event will be triggered when either field matches the current time. For example, “30 4 1,15 * 5” would cause an event to be triggered at 4:30 a.m. on the 1st and 15th of each month, plus every Friday. Instead of the first five fields, one of seven special strings may appear. These seven special strings are described in Table 15. Example 1: “0 0 1,15 * 1” would trigger an event at midnight on the 1st and 15th of each month, as well as on every Monday. To specify days by only one field, the other field should be set to *; “0 0 * * 1” would trigger an event at midnight only on Mondays. Example 2: “15 16 1 * *” would trigger an event at 4:15 p.m. on the first day of each month. Example 3: “0 12 * * 1-5” would trigger an event at noon on Monday through Friday of each week. Example 4: “@weekly” would trigger an event at midnight once a week on Sunday.
time	(Optional) Must be specified if a timer type other than CRON is specified. Must not be specified if the CRON timer type is specified. For watchdog and countdown timers, the number of seconds and milliseconds until the timer expires; for the absolute timer, the calendar time of the expiration time. Time is specified in SSSSSSSSSS[.MMM] format, where SSSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999. An absolute expiration date is the number of seconds and milliseconds since January 1, 1970. If the date specified has already passed, the timer expires immediately.

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

Table 14 Time and Date When CRON Events Will Be Triggered

Field	Allowed Values
minute	0-59
hour	0-23
day of month	1-31
month	1-12 (or names, see below)
day of week	0-7 (0 or 7 is Sun, or names; see Table 15)

Table 15 Special Strings for cron_entry

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

Table 15 Special Strings for cron_entry

@midnight	Same as @daily.
@hourly	Trigger once an hour, “0 * * * *”.

Result String

None

Set _cerrno

No

See Also

event_register_timer_subscriber

event_register_timer_subscriber

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

Syntax

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

Arguments

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

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

Result String

None

Set _cerrno

No

See Also

event_register_timer

event_register_track

Registers for a report event from the Cisco IOS Object Tracking subsystem. Use this Tcl command extension to trigger a policy on the basis of a Cisco IOS Object Tracking subsystem report for a specified object number.

Syntax

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

Arguments

? (represents a number)	(Mandatory) Tracked object number in the range from 1 to 500, inclusive.
tag	(Optional) String identifying a tag that can be used with the trigger Tcl command extension to support multiple event statements within a Tcl script.
state	(Optional) Specifies that the tracked object transition will cause an event to be raised. If **up** is specified, an event will be raised when the tracked object transitions from a down state to an up state. If **down** is specified, an event will be raised when the tracked object transitions from an up state to a down state. If **any** is specified, an event will be raised when the tracked object transitions to or from any state.
queue_priority	(Optional) Priority level at which the script will be queued: • queue_priority low—Specifies that the script is to be queued at the lowest of the three priority levels. • queue_priority normal—Specifies that the script is to be queued at a priority level greater than low priority but less than high priority. • queue_priority high—Specifies that the script is to be queued at the highest of the three priority levels. • queue_priority last—Specifies that the script is to be queued at the lowest priority level. If more than one script is registered with the "queue_priority_last" argument set, these scripts will execute in the order in which the events are published. **Note** The queue_priority argument specifies the queuing priority, but not the execution priority, of the script being registered. If this argument is not specified, the default queuing priority is normal.

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

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

Result String

None

Set _cerrno

No

event_register_wdsysmon

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

Syntax

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

Each argument is position independent.

Note Operator definitions: and (logical and operation), or (logical or operation), andnot (logical and not operation). For example, “sub12_op and” is defined as raise an event when subevent 1 and subevent 2 are true; “sub23_op or” is defined as raise an event when the condition specified in sub12_op is true or subevent 3 is true. The logic can be diagrammed using:

if (((sub1 sub12_op sub2) sub23_op sub3) sub34_op sub4) is TRUE, raise event

Arguments

tag	(Optional) String identifying a tag that can be used with the trigger Tcl command extension to support multiple event statements within a Tcl script.
timewin	(Optional) Time window within which all of the subevents have to occur in order for an event to be generated (specified in SSSSSSSSSS[.MMM] format, where SSSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999).
sub12_op	(Optional) Combination operator for comparison between subevent 1 and subevent 2.
sub23_op	(Optional) Combination operator for comparison between subevent 1 and 2 and subevent 3.
sub34_op	(Optional) Combination operator for comparison between subevent 1 and 2 and subevent 3 and subevent 4.
sub1	(Optional) Indicates that subevent 1 is specified.
subevent-description	(Optional) Syntax for the subevent.
sub2	(Optional) Indicates that subevent 2 is specified.
sub3	(Optional) Indicates that subevent 3 is specified.

sub4	(Optional) Indicates that subevent 4 is specified.
node	(Optional) The node name to be monitored for deadlock conditions is a string that consists of the word "node" followed by two fields separated by a slash character using the following format: node<slot-number>/<cpu-number> The slot-number is the hardware slot number. The cpu-number is the hardware CPU number. For example, the SP CPU in a Supervisor card on a Cisco Catalyst 6500 series switch located in slot 0 would be specified as node0/0. The RP CPU in a Supervisor card on a Cisco Catalyst 6500 series switch located in slot 0 would be addressed as node0/1. If the node argument is not specified, the default node specification is the local node on which the registration is done.
queue_priority	(Optional) Priority level at which the script will be queued: • queue_priority low—Specifies that the script is to be queued at the lowest of the three priority levels. • queue_priority normal—Specifies that the script is to be queued at a priority level greater than low priority but less than high priority. • queue_priority high—Specifies that the script is to be queued at the highest of the three priority levels. • queue_priority last—Specifies that the script is to be queued at the lowest priority level. If more than one script is registered with the "queue_priority_last" argument set, these scripts will execute in the order in which the events are published. **Note** The queue_priority argument specifies the queuing priority, but not the execution priority, of the script being registered. If this argument is not specified, the default queuing priority is normal.
maxrun	(Optional) Maximum run time of the script (specified in SSSSSSSSSS[.MMM] format, where SSSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the default 20-second run-time limit is used.
nice	(Optional) Policy run-time priority setting. When the nice argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.

Subevents

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

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

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

```
1. deadlock procname ?
```

Arguments

procname	(Mandatory) A regular expression that specifies the process name that you wish to monitor for deadlock conditions. This subevent will ignore the time window even if it is given.

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

Arguments

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

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

Arguments

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

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

Arguments

op	(Optional) Comparison operator used to compare the collected total system CPU usage sample percentage with the specified percentage value; if true, an event will be raised.

val	(Optional) The percentage value with which the average CPU usage during the sample period should be compared.
period	(Optional) The time period for averaging the collection of samples (specified in SSSSSSSSSS[.MMM] format, where SSSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the most recent sample is used.

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

Arguments

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

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

Arguments

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

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

Arguments

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

Result String

None

Set _cerrno

No

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

EEM Event Information Tcl Command Extension

event_reqinfo

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

Syntax

event_reqinfo

Arguments

None

Result String

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

For EEM_EVENT_APPLICATION

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

Event Type	Description
event_id	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
event_type	Type of event.
event_type_string	An ASCII string that represents the name of the event for this event type.
event_pub_sec **event_pub_msec**	The time, in seconds and milliseconds, when the event was published to the Embedded Event Manager (EEM).
sub_system	Number assigned to the EEM policy that published the application event. Number is set to 798 because all other numbers are reserved for Cisco use.
type	Event subtype within the specified component.
data1 **data2** **data3** **data4**	Argument data that is passed to the application-specific event when the event is published. The data is character text, an environment variable, or a combination of the two.

For EEM_EVENT_CLI

```
"event_id %u event_type %u event_type_string {%s} event_pub_sec %u event_pub_msec %u
event_severity %u msg {%s} msg_count %d line %u key %u tty %u error_code %u"
```

Event Type	Description
event_id	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
event_type	Type of event.
event_type_string	An ASCII string that represents the name of the event for this event type.
event_pub_sec **event_pub_msec**	The time, in seconds and milliseconds, at which the event was published to the EEM.
event_severity	The severity of the event.

msg	Text entered at the CLI prompt.
msg_count	Number of times the pattern matched before the event was triggered.
line	The text the parser was able to expand up to the point where the matched key was entered.
key	The enter, questionmark, or tab key.
tty	Corresponds to the line number the user is executing the command on.
error_code	The error code in CLI. 0 —No error from parser up to point where a key was entered. 1—Command is ambiguous up to point where a key was entered. 4—Unknown command up to point where a key was entered.

For EEM_EVENT_COUNTER

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

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

For EEM_EVENT_GOLD

```
"event_id %u event_type %u event_type_string {%s} %u card %u sub_card %u"
"event_severity {%s} event_pub_sec %u event_pub_msec %u overall_result %u"
"new_failure {%s} action_notify {%s} tt %u tc %u bl %u ci %u pc %u cn {%s}"
"sn {%s} tn# {%s} ta# %s ec# {%s} rc# %u lf# {%s} tf# %u cf# %u tr# {%s}"
"tr#p# {%s} tr#d# {%s}"
```

Event Type	Description
action_notify	Action notify information in GOLD event: true or false.
bl	The boot-up diagnostic level, which can be one of the following values: • 0: complete diagnostic • 1: minimal diagnostics • 2: bypass diagnostic
card	Card information for the GOLD event.
cf*testnum*	Consecutive failure, where *testnum* is the test number. For example, **cf3** is the EEM built-in environment variable for consecutive failure of test 3.
ci	Card index.
cn	Card name.

ec*testnum*	Test error code, where *testnum* is the test number. For example, **ec3** is the EEM built-in environment variable for the error code of test 3.
event_id	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
event_pub_msec **event_pub_sec**	The time, in milliseconds and seconds, when the event was published to the EEM.
event_severity	GOLD event severity, which can be one of the following values: • normal • minor • major.
event_type	Type of event.
event_type_string	An ASCII string that represents the name of the event for this event type.
lf*testnum*	Last fail time, where *testnum* is the test number. For example, **lf3** is the EEM built-in variable for the last fail time of test 3. The timestamp format is *mmm dd yyyy hh*:*mm*:*ss*. For example, Mar 11 1960 08:47:00.
new_failure	The new test failure information in a GOLD event flag: true or false.
overall_result	The overall diagnostic result, which can be one of the following values: • 0: OK • 3: minor error • 4: major error • 14: unknown result
pc	Port counts.
rc*testnum*	Test total run count, where *testnum* is the test number. For example, **rc3** is the EEM built-in variable for the total run count of test 3.
sn	Card serial number.
sub_card	The subcard on which a GOLD failure event was detected.
ta*testnum*	Test attribute, where *testnum* is the test number. For example, **ta3** is the EEM built-in variable for the test attribute of test 3.
tc	Test counts.
tf*testnum*	Total failure count, where *testnum* is the test number. For example, **tf3** is the EEM built-in variable for the total failure count of test 3.
tn*testnum*	Test name, where *testnum* is the test number. For example, **tn3** is the EEM built-in variable for the name of test 3.

tr*testnum*	Test result, where *testnum* is the test number. For example, **tr6** is the EEM built-in variable for test 6 where test 6 is not a per-port test and not a per-device test. The test result is one of the following values: • P: diagnostic result Pass • F: diagnostic result Fail • U: diagnostic result Unknown
tr*testnum***d***devnum*	Per-device test result, where *testnum* is the test number and *devnum* is the device number. For example, **tr3d20** is the EEM built-in variable for the test result for test 3, device 20. The test result is one of the following values: • P: diagnostic result Pass • F: diagnostic result Fail • U: diagnostic result Unknown
tr*testnum***p***portnum*	Per-port test result, where *testnum* is the test number and *portnum* is the device number. For example, **tr5p20** is the EEM built-in variable for the test result for test 3, port 20. The test result is one of the following values: • P: diagnostic result Pass • F: diagnostic result Fail • U: diagnostic result Unknown
tt	The testing type, which can be one of the following: • 1: A boot-up diagnostic • 2: An on-demand diagnostic • 3: A schedule diagnostic • 4: A monitoring diagnostic

For EEM_EVENT_INTERFACE

```
"event_id %u event_type %u event_type_string {%s} %u event_pub_sec %u event_pub_msec %u"
"event_severity {%s} name {%s} parameter {%s} value %d"
```

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

event_severity	Interface event severity, which can be one of the following values: • normal • minor • major
name	Name of the interface.
parameter	Name of the parameter.
value	The incremental/decremental difference compared to the last event triggered or the absolute value of the parameter being monitored, depending on the specified value of entry_val_is_increment.

For EEM_EVENT_IOSWDSYSMON

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

Event Type	Description
event_id	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
event_type	Type of event.
event_type_string	An ASCII string that represents the name of the event for this event type.
event_pub_sec **event_pub_msec**	The time, in seconds and milliseconds, when the event was published to the EEM.
num_subs	Number of subevents.

Where the subevent info string is for a CPU_UTIL subevent,

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

Subevent Type	Description
type	Type of subevent.
procname	POSIX process name for this subevent.
pid	POSIX process ID for this subevent.
taskname	Cisco IOS task name for this subevent.
taskid	Cisco IOS task ID for this subevent.
value	Actual average CPU utilization over the measured interval.
sec, **msec**	Elapsed time period for this measured interval.

Where the subevent info string is for a MEM_UTIL subevent,

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

Subevent Type	Description
type	Type of subevent.
procname	POSIX process name for this subevent.
pid	POSIX process ID for this subevent.
taskname	Cisco IOS task name for this subevent.
taskid	Cisco IOS task ID for this subevent.
is_percent	TRUE or FALSE depending on whether the value is a percentage value.
value	Total memory use in KB or the actual average memory utilization for this measured interval.

diff	The percentage difference between the oldest sample in the measured interval and the latest sample; a negative value represents a decrease.
sec, **msec**	Elapsed time period for this measured interval.

For EEM_Event_IPSLA

```
"event_ID %u event_type %u event_pub_sec %u event_pub_msec %u event_severity %u"
"group_name %u operation_id %u condition %u reaction_type %u dest_ip_addr %u"
"threshold_rising %u threshold_falling%u measured_threshold_value %u"
"threshold_count1 %u threshold count2 %u"
```

Event Type	Description
event_id	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
event_type	The type of event to monitor for the create, update, and delete flow.
event_type_string	An ASCII string that represents the name of the event for this event type.
event_pub_sec **event_pub_msec**	The time, in seconds and milliseconds, when the event was published to the EEM.
event_severity	The severity of the event.
group_name	The name of the IPSLA group.
operation_id	The IPSLA operation ID.
condition	The condition of IPSLA, which can be one of the following: • cleared • occurred
reaction_type	The IPSLA reaction type.
dest_ip_address	The IPSLA destination IP address.
threshold rising	The IPSLA configured rising threshold value.
threshold falling	The IPSLA configured falling threshold value.
measured_threshold_value	The measured threshold value of the IPSLA operation.
threshold_count1	Corresponds to the argument of the threshold type1.
threshold_count2	Corresponds to the argument of the threshold type2.

For EEM_EVENT_NF

```
"event_ID %u event_type %u event_type_string {%s} event_pub_sec %u event_pub_msec %u
event_severity %u monitor_name %u event1-event4_field %u event1-event4_value
```

Event Type	Description
event_id	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
event_type	The type of event to monitor for the create, update, and delete flow.

event_type_string	An ASCII string that represents the name of the event for this event type.
event_pub_sec **event_pub_msec**	The time, in seconds and milliseconds, when the event was published to the EEM.
event_severity	The severity of the NetFlow event.
montior_name	The name of the NetFlow monitor.
event1-event4_field	Specifies the event and its attributes to monitor. Valid values are **event1**, **event2**, **event3**, and **event4**.
event1-event4_value	Specifies the event value and its attributes to monitor. Valid values are **event1**, **event2**, **event3**, and **event4**.

For EEM_EVENT_NONE

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

Event Type	Description
event_id	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
event_type	Type of event.
event_type_string	An ASCII string that represents the name of the event for this event type.
event_pub_sec **event_pub_msec**	The time, in seconds and milliseconds, when the event was published to the EEM.
event_severity	The severity of the event.
argc **arg1** **arg2** **arg3** **arg4** **arg6** **arg7** **arg8** **arg9** **arg10** **arg11** **arg12** **arg13** **arg14** **arg15**	The parameters that are passed from the XML SOAP command to the script.

For EEM_EVENT_OIR

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

```
"slot %u event %s"
```

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

For EEM_EVENT_PROCESS (Software Modularity Only)

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

Event Type	Description
event_id	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
event_type	Type of event.
event_type_string	An ASCII string that represents the name of the event for this event type.
event_pub_sec **event_pub_msec**	The time, in seconds and milliseconds, when the event was published to the EEM.
sub_system	Number assigned to the EEM policy that published the application-specific event. Number is set to 798 because all other numbers are reserved for Cisco use.
instance	Process instance ID.
process_name	Process name.
path	Process absolute name including path.
exit_status	Process last exit status.
respawn_count	Number of times that the process was restarted.
last_respawn_sec **last_respawn_msec**	The calendar time when the last restart occurred.
fail_count	Number of restart attempts of the process that failed. This count will be reset to 0 when the process is successfully restarted.

dump_count	Number of core dumps taken of the process.
node_name	Name of the node that the process is on. The node name is a string that consists of the word "node" followed by two fields separated by a slash character using the following format: **node***slot-number*/*cpu-number* The slot-number is the hardware slot number. The cpu-number is the hardware CPU number.

For EEM_EVENT_RESOURCE

```
"event_id %u event_type %u event_type_string {%s} %u event_pub_sec %u event_pub_msec %u"
"owner_id %lld user_id %lld" time_sent %llu dampen_time %d notify_data_flags %u"
"level {%s} direction {%s} configured_threshold %u current_value %u"
"policy_violation_flag {%s} policy_id %d"
```

Event Type	Description
event_id	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
event_type	Type of event.
event_type_string	An ASCII string that represents the name of the event for this event type.
event_pub_sec **event_pub_msec**	The time, in seconds and milliseconds, when the event was published to the EEM.
owner_id	The Embedded Resource Manager (ERM) owner ID.
user_id	The ERM user ID.
time_sent	The ERM event time, in nanoseconds.
dampen_time	The ERM dampen time, in nanoseconds.
notify_data_flags	The ERM notify data flag.
level	The ERM event level. The four event levels are normal, minor, major, and critical.
direction	The ERM event direction. The event direction can be one of the following: up, down, or no change.
configured_threshold	The configured ERM threshold.
current_value	The current value reported by ERM.
policy_violation_flag	The ERM policy violation flag; either false or true.
policy_id	The ERM policy ID.

For EEM_EVENT_RF

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

Event Type	Description
event_id	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.

event_type	Type of event.
event_type_string	An ASCII string that represents the name of the event for this event type.
event_pub_sec **event_pub_msec**	The time, in seconds and milliseconds, when the event was published to the EEM.
event	RF progression or status event notification that caused this event to be published.

For EEM_EVENT_Routing

```
"event_id %u event_type %u event_type_string {%s} %u event_pub_sec %u event_pub_msec %u"
"event_severity {%s} %u network %u mask %u protocol %u lastgateway %u distance %u"
"time_sec %u time_msec %u metric %u lastinterface %u"
```

Event Type	Description
event_id	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
event_type	Type of event.
event_type_string	An ASCII string that represents the name of the event for this event type.
event_pub_sec **event_pub_msec**	The time, in seconds and milliseconds, when the event was published to the EEM.
event_severity	The severity of the event.
network	The network prefix in IP address format
mask	The network mask in IP address format
protocol	Type of network protocol.
type	Type of event to add, remove or modify.
lastgateway	The last known gateway.
distance	The administrative distance.
time_sec **time_msec**	Time of event in seconds and milliseconds, when the event was published to the EEM.
metric	Path metric.
lastinterface	The last known interface.

For EEM_EVENT_RPC

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

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

event_pub_sec **event_pub_msec**	The time, in seconds and milliseconds, when the event was published to the EEM.
argc **arg0** **arg1** **arg2** **arg3** **arg4** **arg6** **arg7** **arg8** **arg9** **arg10** **arg11** **arg12** **arg13** **arg14**	The parameters that are passed from the XML SOAP command to the script.

For EEM_EVENT_SNMP

```
"event_id %u event_type %u event_type_string {%s} %u event_pub_sec %u event_pub_msec %u"
"event_severity {%s} oid {%s} val {%s} delta_val {%s}"
```

Event Type	Description
event_id	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
event_type	Type of event.
event_type_string	An ASCII string that represents the name of the event for this event type.
event_pub_sec **event_pub_msec**	The time, in seconds and milliseconds, when the event was published to the EEM.
event_severity	SNMP event severity, which can be one of the following values: • normal • minor • major
oid	Object ID of data element, in SNMP dot notation.
val	Value of the data element.
delta_val	Delta value between the value of the policies.

For EEM_EVENT_SNMP_Notification

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

```
event_severity {%s}" "oid {%s} oid_val {%s} src_ip_addr {%s} dest_ip_addr {%s} x_x_x_x_x
(varbinds) {%s} trunc_vb_buf {%s} trap_oid {%s} enterprise_oid {%s} generic_trap %u
specific_trap %u"
```

Event Type	Description
event_id	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
event_type	Type of event.
event_type_string	An ASCII string that represents the name of the event for this event type.
event_pub_sec **event_pub_msec**	The time, in seconds and milliseconds, when the event was published to the EEM.
oid	An user specified object ID.
oid_val	An user specified object ID value.
src_ip_addr	The source IP address of the SNMP protocol data unit (PDU).
dest_ip_addr	The destination IP address of the SNMP PDU.
x_x_x_x_x (varbinds)	The SNMP PDU varbind information.
trap_oid	Indicates the trap OID value.
enterprise_oid	Indicates the enterprise OID value.
generic_trap	Indicates one of a number of generic trap types. There are seven generic trap numbers zero to six.
specific_trap	Indicates one of a number of specific trap codes.

Event_reqinfo for EEM_EVENT_SNMP_Object

```
"event_id %u event_type %u event_type_string {%s} %u event_pub_sec %u event_pub_msec %u
event_severity {%s}" "oid {%s} request {%s} request_type {%s} value %u"
```

Event Type	Description
event_id	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
event_type	Type of event.
event_type_string	An ASCII string that represents the name of the event for this event type.
event_pub_sec **event_pub_msec**	The time, in seconds and milliseconds, when the event was published to the EEM.
event_severity	The severity of the event.
oid	The ID of the SNMP object in the received get or set request.
request	The get or set request type.
request_type	The type of request (exact or next).
value	For set requests only. The value to set the object to.

For EEM_EVENT_SYSLOG_MSG

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

Event Type	Description
event_id	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
event_type	Type of event.
event_type_string	An ASCII string that represents the name of the event for this event type.
event_pub_sec **event_pub_msec**	The time, in seconds and milliseconds, when the event was published to the EEM.
msg	The last syslog message that matches the pattern.

For EEM_EVENT_TIMER_ABSOLUTE EEM_EVENT_TIMER_COUNTDOWN EEM_EVENT_TIMER_WATCHDOG

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

Event Type	Description
event_id	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
event_type	Type of event.
event_type_string	An ASCII string that represents the name of the event for this event type.
event_pub_sec **event_pub_msec**	The time, in seconds and milliseconds, when the event was published to the EEM.
timer_type	Type of the timer. Can be one of the following: • watchdog • countdown • absolute
timer_time_sec **timer_time_msec**	Time when the timer expired.
timer_remain_sec **timer_remain_msec**	The remaining time before the next expiration.

For EEM_EVENT_TIMER_CRON

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

Event Type	Description
event_id	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
event_type	Type of event.
event_type_string	An ASCII string that represents the name of the event for this event type.
event_pub_sec **event_pub_msec**	The time, in seconds and milliseconds, when the event was published to the EEM.
timer_type	Type of the timer.
timer_time_sec **timer_time_msec**	Time when the timer expired.

For EEM_EVENT_TRACK

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

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

For EEM_EVENT_WDSYSMON (Software Modularity Only)

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

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

Where the subevent info string is for a deadlock subevent:

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

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

Where each entry is:

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

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

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

For dispatch_mgr Subevent

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

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

For cpu_proc Subevent

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

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

For cpu_tot Subevent

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

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

For mem_proc Subevent

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

Subevent Type	Description
type	Type of wdsysmon subevent.
node	Name of the node that the POSIX process is on.
procname	POSIX process name for this subevent.
pid	POSIX process ID for this subevent. **Note** The three fields above describe the process whose memory usage is being monitored.
is_percent	Can be either TRUE or FALSE. TRUE means that the value is a percentage value; FALSE means that the value is an absolute value (may be an averaged value).
value	If the **sec** and **msec** variables are specified as 0 or are unspecified in the event registration Tcl command extension, the process used memory is in the latest sample. If a time window is specified and is greater than zero in the event registration Tcl command extension, the averaged process used memory utilization is in the given time window.

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

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

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

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

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

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

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

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

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

For mem_tot_avail Subevent

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

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

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

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

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

- **used** is 0.
- **avail** is the averaged total available memory sample value in the specified time window.

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

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

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

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

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

For mem_tot_used Subevent

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

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

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

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

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

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

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

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

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

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

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

Set _cerrno

Yes

EEM Event Tcl Command Extension

event_completion

Sends a notification to the EEM server that the policy is done servicing the event that triggered it. The event only takes a single argument which is the **return_code** of this event instance.

Syntax

```
event_completion status ?
```

Arguments

status	(Mandatory) Exit status (return_code) of this event instance. A value of zero indicates no error and any other integer value indicates an error.

Result String

None

Set _cerrno

No

event_completion_with_wait

The **event_completion_with_wait** command combines the two commands **event_completion** and **event_wait** into a single command for ease of use.

The **event_completion** command sends a notification to the EEM server that the policy is done servicing the event that triggered it. The event only takes a single argument which is the **return_code** of this event instance.

The **event_wait** places the Tcl policy into a sleep state. When the Tcl policy receives a new signal announcing a new event, the policy is placed into a wake state and again returns to a sleep state. This loop continues. If **event_wait** policy is invoked before **event_completed** policy, an error results and the policy exits.

Syntax

```
event_completion_with_wait status ? [refresh_vars]
```

Arguments

status	(Mandatory) exit_status (return_code) of this event instance. A value of zero indicates no error. Any other integer value indicates an error.
refresh_vars	(Optional) Indicates whether built-in and environment variables should be updated (refreshed) from the EEM Policy Director during this event instance.

Result String

None

Set _cerrno

Yes

Sample Usage

Here is a similar example as above using this single command:

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

set i 1

while {1 == 1} {  # Start high performance policy loop

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

   action_syslog msg "event $i serviced" priority info

   if {$i == 5} {
       action_syslog msg "Exiting after servicing 5 events" priority info
       exit 0
   }
```

```
    incr i

    array set _event_state_arr [event_completion_with_wait status 0 refresh_vars 1]

    if {$_event_state_arr(event_state) != 0} {
        action_syslog msg "Exiting: failed event_state " \
                          " $event_state_arr(event_state)" priority info
        exit 0
    }
}
```

Note The running configuration output is the same as the event_publish Tcl command.

event_publish

Publishes an application-specific event.

Syntax

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

Arguments

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

Result String

None

Set _cerrno

Yes

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

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

Sample Usage

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

Script1 publishes a type 9999 EEM event to cause Script2 to run for the first time. Script1 is registered as a none event and is run using the Cisco IOS CLI **event manager run** command. Script2 is registered as an EEM application event of type 9999, and this script checks to see if the application publish arg1 data (the iteration number) exceeds the EEM environment variable test_iterations value. If the test_iterations value is exceeded, the script writes a message and exits; otherwise the script executes the remaining statements and reschedules another run. To measure the CPU utilization for Script2, use a value of test_iterations that is a multiple of 10 to calculate the amount of average CPU time used by Script2.

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

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

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

Script1 (script1.tcl)

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

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

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

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

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

Script2 (script2.tcl)

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

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

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

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

# Use the arg1 info from the previous run to determine when to end.
if {$iter >= $test_iterations} {
```

```
    # Log a message.
    action_syslog priority info msg "EEM application_publish test end"
    if {$_cerrno != 0} {
      set result [format \
            "component=%s; subsys err=%s; posix err=%s;\n%s" \
            $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
      error $result
    }
    exit 0
}
set iter [expr $iter + 1]

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

# Do whatever processing that you want to measure here.

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

event_wait

Places the Tcl policy into a sleep state. When the Tcl policy receives a new signal announcing a new event, the policy is placed into a wake state and again returns to a sleep state. This loop continues. If **event_wait** policy is invoked before **event_completed** policy, an error results and the policy exits.

Syntax

```
event_wait [refresh_vars]
```

Arguments

refresh_vars	(Optional) Indicates whether built-in and environment variables should be updated (refreshed) from the EEM Policy Director during this event instance.

Result String

None

Set _cerrno

No

Sample Usage

The **event_wait** event detector returns an array type value with a single element named **event_state**. Event_state is a value sent back from the EEM Server indicating whether or not an error has occurred in processing the event. An example of an error here would be if the user configured **event_wait** before configuring **event_completion** when handling the event instance.

The following sample output shows the use of both **event_completion** and **event_wait** Tcl commands:

```
::cisco::eem::event_register_syslog tag e1 occurs 1 pattern CLEAR maxrun 0

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

set i 1

while {1 == 1} {  # Start high performance policy loop

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

   action_syslog msg "event $i serviced" priority info

   if {$i == 5} {
       action_syslog msg "Exiting after servicing 5 events" priority info
       exit 0
   }

   incr i

   event_completion status 0

   array set _event_state_arr [event_wait refresh_vars 0]
```

```
   if {$_event_state_arr(event_state) != 0} {
       action_syslog msg "Exiting: failed event_state " \
                         " $event_state_arr(event_state)" priority info
       exit 0
   }
}
```

Here is an example of the running configuration:

```
Router#
01:00:44: %SYS-5-CONFIG_I: Configured from console by consoleclear counters
Clear "show interface" counters on all interfaces [confirm]
Router#
01:00:49: %CLEAR-5-COUNTERS: Clear counter on all interfaces by console
01:00:49: %HA_EM-6-LOG: high_perf_example.tcl: event 1 serviced
Router#
Router#clear counters
Clear "show interface" counters on all interfaces [confirm]
Router#
Router#
01:00:53: %CLEAR-5-COUNTERS: Clear counter on all interfaces by console
01:00:53: %HA_EM-6-LOG: high_perf_example.tcl: event 2 serviced
Router#clear counters
Clear "show interface" counters on all interfaces [confirm]
Router#
Router#
01:00:56: %CLEAR-5-COUNTERS: Clear counter on all interfaces by console
01:00:56: %HA_EM-6-LOG: high_perf_example.tcl: event 3 serviced
Router#
Router#
Router#clear counters
Clear "show interface" counters on all interfaces [confirm]
Router#
01:00:59: %CLEAR-5-COUNTERS: Clear counter on all interfaces by console
Router#
01:00:59: %HA_EM-6-LOG: high_perf_example.tcl: event 4 serviced
01:00:59: %HA_EM-6-LOG: high_perf_example.tcl: Exiting after servicing 5 events
Router#
Router#
Router#copy tftp disk1:
Address or name of remote host [dirt]?
Source filename [user/eem_scripts/high_perf_example.tcl]?
Destination filename [high_perf_example.tcl]?
%Warning:There is a file already existing with this name
Do you want to over write? [confirm]
Accessing tftp://dirt/user/eem_scripts/high_perf_example.tcl...
Loading user/eem_scripts/high_perf_example.tcl from 192.0.2.19 (via FastEthernet0/0): !
[OK - 909 bytes]

909 bytes copied in 0.360 secs (2525 bytes/sec)
Router#
Router#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#no event manager policy high_perf_example.tcl
Router(config)#event manager po high_perf_example.tcl
Router(config)#end
Router#
Router#
Router#
Router#
01:02:19: %SYS-5-CONFIG_I: Configured from console by consoleclear counters
Clear "show interface" counters on all interfaces [confirm]
```

```
Router#
01:02:23: %CLEAR-5-COUNTERS: Clear counter on all interfaces by console
Router#
Router#
01:02:23: %HA_EM-6-LOG: high_perf_example.tcl: event 1 serviced
Router#
Router#clear counters
Clear "show interface" counters on all interfaces [confirm]
Router#
Router#
01:02:26: %CLEAR-5-COUNTERS: Clear counter on all interfaces by console
01:02:26: %HA_EM-6-LOG: high_perf_example.tcl: event 2 serviced
Router#
Router#clear counters
Clear "show interface" counters on all interfaces [confirm]
Router#
Router#
01:02:29: %CLEAR-5-COUNTERS: Clear counter on all interfaces by console
01:02:29: %HA_EM-6-LOG: high_perf_example.tcl: event 3 serviced
Router#
Router#clear counters
Clear "show interface" counters on all interfaces [confirm]
Router#
Router#
01:02:33: %CLEAR-5-COUNTERS: Clear counter on all interfaces by console
Router#
01:02:33: %HA_EM-6-LOG: high_perf_example.tcl: event 4 serviced
Router#
Router#clear counters
Clear "show interface" counters on all interfaces [confirm]
Router#
Router#
Router#
01:02:36: %CLEAR-5-COUNTERS: Clear counter on all interfaces by console
01:02:36: %HA_EM-6-LOG: high_perf_example.tcl: event 5 serviced
01:02:36: %HA_EM-6-LOG: high_perf_example.tcl: Exiting after servicing 5 events
Router#
```

Also while an event has been serviced and is waiting for the next event to come in **show event manager policy active** command will display the following output:

```
Router#show event manager policy active
Key: p - Priority        :L - Low, H - High, N - Normal, Z - Last
     s - Scheduling node :A - Active, S - Standby

default class - 1 script event
 no.  job id      p s status  time of event             event type          name
 1    11          N A wait    Mon Oct20 14:15:24 2008   syslog
high_perf_example.tcl
```

In the above example the status is wait. This indicates that the policy is waiting for the next event to come in.

EEM Multiple Event Support Tcl Command Extensions

trigger

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

Syntax

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

Arguments

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

Result String

None

Set _cerrno

No

correlate

Builds a single complex event and allows boolean logic to relate events and tracked objects.

Syntax

```
correlate event ? track ? [andnot | and | or] event ? track ?
```

Arguments

event	Specifies the event that can be used with the **trigger** command to support multiple event statements within an script. If the event associated with the *event-tag* argument occurs for the number of times specified by the **trigger** command, the result is true. If not, the result is false.
track	Specifies the event object number for tracking. The range is from 1 to 500. If the tracked object is set, the result of the evaluation is true. If the tracked object is not set or is undefined, the result of the evaluation is false. This result is regardless of the state of the object.
andnot	(Optional) Specifies that if event 1 occurs the action is executed, and if event 2 and event 3 occur together the action is not executed.
and	(Optional) Specifies that if event 1 occurs the action is executed, and if event 2 and event 3 occur together the action is executed.
or	(Optional) Specifies that if event 1 occurs the action is executed, or else if event 2 and event 3 occur together the action is executed.

Result String

None

Set _cerrno

No

attribute

Specifies a complex event.

Syntax

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

Arguments

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

Result String

None

Set _cerrno

No

EEM Action Tcl Command Extensions

action_policy

Allows a Tcl script to run an Embedded Event Manager (EEM) policy that has been registered with the None event detector. The action of running an EEM policy can also be performed using the **event manager run** command.

Syntax

```
action_policy ?
```

Arguments

? (represents a string)	(Mandatory) The name of the EEM policy to be scheduled for execution. The policy must have been previously registered with the None event detector.

None

Result String

None

Set _cerrno

Yes

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

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

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

This error means that the policy is unknown because it is not registered.

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

This error means that the action command requested was unknown.

action_process

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

Syntax

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

Arguments

start	(Mandatory) Specifies that a process is to be started.
restart	(Mandatory) Specifies that a process is to be restarted.
kill	(Mandatory) Specifies that a process is to be stopped (killed).
job_id	(Optional) System manager assigned job ID for the process. If you specify this argument, it must be an integer between 1 and 4294967295, inclusive.
process_name	(Optional) Process name. Either job_id must be specified or process_name and instance must be specified.
instance	(Optional) Process instance ID. If you specify this argument, it must be an integer between 1 and 4294967295, inclusive.

Result String

None

Set _cerrno

Yes

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

This error means that the action command requested was unknown.

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

This error means that the arguments passed in were invalid.

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

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

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

This error means that the process name was not known.

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

This error means that an object size exceeded its maximum.

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

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

action_program

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

Syntax

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

Arguments

path	(Mandatory) The pathname of a program to run.
argv	(Optional) The argument string of the program.
envp	(Optional) The environment string of the program.
stdin	(Optional) The pathname for stdin.
stdout	(Optional) The pathname for stdout.
stderr	(Optional) The pathname for stderr.

Result String

None

Set _cerrno

Yes

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

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

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

This error means that the action command requested was unknown.

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

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

action_reload

Reloads the router.

Syntax

```
action_reload
```

Arguments

None

Result String

None

Set _cerrno

Yes

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

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

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

This error means that the action command requested was unknown.

action_script

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

Syntax

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

Arguments

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

Result String

None

Set _cerrno

Yes

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

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

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

This error means that the action command requested was unknown.

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

This error means that a configuration error has occurred.

action_snmp_trap

Sends a Simple Network Management Protocol (SNMP) trap using the Embedded Event Manager Notification MIB.

Syntax

```
action_snmp_trap [intdata1 ?] [intdata2 ?] [strdata ?]
```

Arguments

intdata1	(Optional) Arbitrary integer sent in trap.
intdata2	(Optional) Arbitrary integer sent in trap.
strdata	(Optional) Arbitrary string data sent in trap.

Result String

None

Set _cerrno

Yes

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

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

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

This error means that the action command requested was unknown.

action_snmp_object_value

Sets a Simple Network Management Protocol (SNMP) object ID and value to be returned for the SNMP get request.

Syntax

```
action_snmp_object_value event_id ? {int|uint|counter|gauge|ipv4|octect|string} ?
[next_oid ?]
```

Arguments

event_id	The event ID.
int	A 32-bit number used to specify a numbered type within the context of a managed object.
uint	A 32-bit number used to represent decimal value.
counter	A 32-bit number with a minimum value of 0.
gauge	A 32-bit number with a minimum value of 0.
ipv4	IP version 4 address.
octect	An octet string in hex notation used to represent physical addresses.
string	An octet string in text notation used to represent text strings.
next_oid	The OID of the next object in the table; NULL if it is the last object in the table.

Result String

None

Set _cerrno

Yes

action_switch

Switches processing to a secondary processor in a fully redundant environment. Before using the **action_switch** Tcl command extension, you must install a backup processor in the device. If the hardware is not fully redundant, the switchover action will not be performed.

Syntax

```
action_switch
```

Arguments

None

Result String

None

Set _cerrno

Yes

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

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

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

This error means that the action command requested was unknown.

action_syslog

Generates a periodic syslog message using the specified facility when an EEM script is triggered.

Syntax

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

Arguments

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

Result String

None

Set _cerrno

Yes

action_track_read

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

Syntax

```
action_track_read ?
```

Arguments

? (represents a number)	(Mandatory) Tracked object number in the range from 1 to 500, inclusive.

Result String

number `{%u}`
state `{%s}`

Set _cerrno

Yes

`FH_ENOTRACK`

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

action_track_set

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

Syntax

```
action_track_set ? state up|down
```

Arguments

? (represents a number)	(Mandatory) Tracked object number in the range from 1 to 500, inclusive.
state	(Mandatory) Specifies that the state of the tracked object will be set. If up is specified, the state of the tracked object will be set to up. If down is specified, the state of the tracked object will be set to down.

Result String

None

Set _cerrno

Yes

```
FH_ENOTRACK
```

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

EEM Utility Tcl Command Extensions

appl_read

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

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

Syntax

```
appl_read name ? length ?
```

Arguments

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

Result String

```
data %s
```

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

Set _cerrno

Yes

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

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

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

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

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

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

appl_reqinfo

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

Syntax

```
appl_reqinfo key ?
```

Arguments

key	(Mandatory) The string key of the data.

Result String

```
data %s
```

Where data is the application string data to be retrieved.

Set _cerrno

Yes

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

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

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

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

appl_setinfo

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

Syntax

```
appl_setinfo key ? data ?
```

Arguments

key	(Mandatory) The string key of the data.
data	(Mandatory) The application string data to save.

Result String

None

Set _cerrno

Yes

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

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

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

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

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

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

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

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

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

This error means that the API message length was invalid.

counter_modify

Modifies a counter value.

Syntax

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

Arguments

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

Result String

```
val_remain %d
```

Where val_remain is the current value of the counter.

Set _cerrno

Yes

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

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

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

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

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

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

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

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

description

Provides a brief description of the registered policy.

Syntax

```
description ?
```

Arguments

line	(Optional) Brief description of the policy consisting of 1 to 240 characters.

Result String

None

Set _cerrno

Yes

Sample Usage

The description statement is entered by the author of the policy. It can appear before or after any event registration statement in Tcl. The policy can have only one description.

Note Registration of a policy with more than one description statement will fail.

The following example shows how a brief description is provided for the **event_register_syslog** policy:

```
::cisco::eem::description "This Tcl command looks for the word count in syslog messages."
::cisco::eem::event_register_syslog tag 1 ...
::cisco::eem::event_register_snmp_object tag 2 ...
::cisco::eem::trigger
       ::cisco::eem::correlate event 1 and event 2
       ::cisco::eem::attribute tag 1 occurs 1
       ::cisco::eem::attribute tag 2 occurs 1
```

fts_get_stamp

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

Syntax

```
fts_get_stamp
```

Arguments

None

Result String

```
nsec %d
```

Where nsec is the number of nanoseconds since boot.

Set _cerrno

No

register_counter

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

Syntax

```
register_counter name ?
```

Arguments

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

Result String

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

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

Set _cerrno

Yes

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

This error means that the subscriber index was invalid.

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

This error means that the event detector was unavailable.

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

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

register_timer

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

Syntax

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

Arguments

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

Result String

```
event_id %u
```

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

Set _cerrno

Yes

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

This error means that the subscriber index was invalid.

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

This error means that the event detector was unavailable.

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

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

timer_arm

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

Syntax

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

Arguments

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

Result String

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

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

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

Set _cerrno

Yes

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

This error means that the event was not registered.

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

This error means that the event detector was unavailable.

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

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

timer_cancel

Cancels a timer.

Syntax

```
timer_cancel event_id ?
```

Arguments

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

Result String

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

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

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

Set _cerrno

Yes

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

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

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

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

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

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

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

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

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

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

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

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

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

This error means that the event detector was unavailable.

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

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

unregister_counter

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

Syntax

```
unregister_counter event_id ? event_spec_id ?
```

Arguments

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

Result String

None

Set _cerrno

Yes

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

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

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

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

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

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

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

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

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

This error means that the subscriber index was invalid.

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

This error means that the event detector was unavailable.

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

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

EEM System Information Tcl Command Extensions

Note All EEM system information commands—**sys_reqinfo_*xxx***—have the Set _cerrno section set to yes.

sys_reqinfo_cli_freq

Queries the frequency information of all command-line interface (CLI) events.

Syntax

```
sys_reqinfo_cli_freq
```

Arguments

None

Result String

```
rec_list {{CLI frequency string 0},{CLI frequency str 1}, ...}
```

Where each CLI frequency string is:

```
time_sec %ld time_msec %ld match_count %u raise_count %u occurs %u period_sec %ld
period_msec %ld pattern {%s}
```

rec_list	Marks the start of the CLI event frequency list.
time_sec time_msec	Last time when this CLI event was raised.
match count	Number of times that a CLI command matches the pattern specified by this CLI event specification.
raise_count	Number of times that this CLI event was raised. The following fields are information about the CLI event specification: • sync—A “yes” means that event publish should be performed sychronously. The event detector will be notified when the Event Manager Server has completed publishing the event. The Event Manager Server will return a code that indicates whether or not the CLI command should be executed. • skip—A “yes” means that the CLI command should not be executed if the sync flag is not set.
occurs	Number of occurrences before an event is raised; if this argument is not specified, an event is raised on the first occurrence.
period_sec period_msec	Number of occurrences must occur within this number of POSIX timer units in order to raise event; if this argument is not specified, it does not apply.
pattern	Regular expression used to perform CLI command pattern matching.

Set _cerrno

Yes

sys_reqinfo_cli_history

Queries the history of command-line interface (CLI) commands.

Syntax

```
sys_reqinfo_cli_history
```

Arguments

None

Result String

```
rec_list {{CLI history string 0}, {CLI history str 1},...}
```

Where each CLI history string is:

```
time_sec %ld time_msec %ld cmd {%s}
```

rec_list	Marks the start of the CLI command history list.
time_sec time_msec	Time when the CLI command was run.
cmd	Text of the CLI command.

Set _cerrno

Yes

sys_reqinfo_cpu_all

Queries the CPU utilization of the top processes (both POSIX processes and IOS processes) during a specified time period and in a specified order. This Tcl command extension is supported only in Software Modularity images.

Syntax

```
sys_reqinfo_cpu_all order cpu_used [sec ?] [msec ?] [num ?]
```

Arguments

order	(Mandatory) Order used for sorting the CPU utilization of processes.
cpu_used	(Mandatory) Specifies that the average CPU utilization, for the specified time window, will be sorted in descending order.
sec msec	(Optional) The time period, in seconds and milliseconds, during which the average CPU utilization is calculated. Must be integers in the range from 0 to 4294967295. If not specified, or if both sec and msec are specified as 0, the most recent CPU sample is used.
num	(Optional) Number of entries from the top of the sorted list of processes to be displayed. Must be an integer in the range from 1 to 4294967295. Default value is 5.

Result String

```
rec_list {{process CPU info string 0},{process CPU info string 1}, ...}
```

Where each process CPU info string is:

```
pid %u name {%s} cpu_used %u
```

rec_list	Marks the start of the process CPU information list.
pid	Process ID.
name	Process name.
cpu_used	Specifies that if sec and msec are specified with a number greater than zero, the average percentage is calculated from the process CPU utilization during the specified time period. If sec and msec are both zero or not specified, the average percentage is calculated from the process CPU utilization in the latest sample.

Set _cerrno

Yes

sys_reqinfo_crash_history

Queries the crash information of all processes that have ever crashed. This Tcl command extension is supported only in Software Modularity images.

Syntax

```
sys_reqinfo_crash_history
```

Arguments

None

Result String

```
rec_list {{crash info string 0},{crash info string 1}, ...}
```

Where each crash info string is:

```
job_id %u name {%s} respawn_count %u fail_count %u dump_count %u
inst_id %d exit_status 0x%x exit_type %d proc_state {%s} component_id 0x%x
crash_time_sec %ld crash_time_msec %ld
```

job_id	System manager assigned job ID for the process. An integer between 1 and 4294967295, inclusive.
name	Process name.
respawn_count	Total number of restarts for the process.
fail_count	Number of restart attempts of the process. This count is reset to zero when the process is successfully restarted.
dump_count	Number of core dumps performed.
inst_id	Process instance ID.
exit_status	Last exit status of the process.
exit_type	Last exit type.
proc_state	Sysmgr process states. One of the following: error, forced_stop, hold, init, ready_to_run, run, run_rnode, stop, waitEOltimer, wait_rnode, wait_spawntimer, wait_tpl.
component_id	Version manager assigned component ID for the component to which the process belongs.
crash_time_sec crash_time_msec	Seconds and milliseconds since January 1, 1970, which represent the last time the process crashed.

Set _cerrno

Yes

sys_reqinfo_mem_all

Queries the memory usage of the top processes (both POSIX and IOS) during a specified time period and in a specified order. This Tcl command extension is supported only in Software Modularity images.

Syntax

```
sys_reqinfo_mem_all order allocates|increase|used [sec ?] [msec ?] [num ?]
```

Arguments

order	(Mandatory) Order used for sorting the memory usage of processes.
allocates	(Mandatory) Specifies that the memory usage is sorted by the number of process allocations during the specified time window, and in descending order.
increase	(Mandatory) Specifies that the memory usage is sorted by the percentage of process memory increase during the specified time window, and in descending order.
used	(Mandatory) Specifies that the memory usage is sorted by the current memory used by the process.
sec msec	(Optional) The time period, in seconds and milliseconds, during which the process memory usage is calculated. Must be integers in the range from 0 to 4294967295. If both sec and msec are specified and are nonzero, the number of allocations is the difference between the number of allocations in the oldest and latest samples collected in the time period. The percentage is calculated as the the percentage difference between the memory used in the oldest and latest samples collected in the time period. If not specified, or if both sec and msec are specified as 0, the first sample ever collected is used as the oldest sample; that is, the time period is set to be the time from startup until the current moment.
num	(Optional) Number of entries from the top of the sorted list of processes to be displayed. Must be an integer in the range from 1 to 4294967295. Default value is 5.

Result String

```
rec_list {{process mem info string 0},{process mem info string 1}, ...}
```

Where each process mem info string is:

```
pid %u name {%s} delta_allocs %d initial_alloc %u current_alloc %u percent_increase %d
```

rec_list	Marks the start of the process memory usage information list.
pid	Process ID.
name	Process name.
delta_allocs	Specifies the difference between the number of allocations in the oldest and latest samples collected in the time period.
initial_alloc	Specifies the amount of memory, in kilobytes, used by the process at the start of the time period.

current_alloc	Specifies the amount of memory, in kilobytes, currently used by the process.
percent_increase	Specifies the percentage difference between the memory used in the oldest and latest samples collected in the time period. The percentage difference can be expressed as current_alloc minus initial_alloc times 100 and divided by initial_alloc.

Set _cerrno

Yes

sys_reqinfo_proc

Queries the information about a single POSIX process. This Tcl command extension is supported only in Software Modularity images.

Syntax

```
sys_reqinfo_proc job_id ?
```

Arguments

job_id	(Mandatory) System manager assigned job ID for the process. Must be an integer between 1 and 4294967295, inclusive.

Result String

```
job_id %u component_id 0x%x name {%s} helper_name {%s} helper_path {%s} path {%s}
node_name {%s} is_respawn %u is_mandatory %u is_hold %u dump_option %d
max_dump_count %u respawn_count %u fail_count %u dump_count %u
last_respawn_sec %ld last_respawn_msec %ld inst_id %u proc_state %s
level %d exit_status 0x%x exit_type %d
```

job_id	System manager assigned job ID for the process. An integer between 1 and 4294967295, inclusive.
component_id	Version manager assigned component ID for the component to which the process belongs.
name	Process name.
helper_name	Helper process name.
helper_path	Executable path of the helper process.
path	Executable path of the process.
node_name	System manager assigned node name for the node to which the process belongs.
is_respawn	Flag that specifies that the process can be respawned.
is_mandatory	Flag that specifies that the process must be alive.
is_hold	Flag that specifies that the process is spawned until called by the API.
dump_option	Core dumping options.
max_dump_count	Maximum number of core dumping permitted.
respawn_count	Total number of restarts for the process.
fail_count	Number of restart attempts of the process. This count is reset to zero when the process is successfully restarted.
dump_count	Number of core dumps performed.
last_respawn_sec last_respawn_msec	Seconds and milliseconds in POSIX timer units since January 1, 1970, which represent the last time the process was started.
inst_id	Process instance ID.
proc_state	Sysmgr process states. One of the following: error, forced_stop, hold, init, ready_to_run, run, run_rnode, stop, waitEOltimer, wait_rnode, wait_spawntimer, wait_tpl.

level	Process run level.
exit_status	Last exit status of the process.
exit_type	Last exit type.

Set _cerrno

Yes

sys_reqinfo_proc_all

Queries the information of all POSIX processes. This Tcl command extension is supported only in Software Modularity images.

Syntax

```
sys_reqinfo_proc_all
```

Arguments

None

Result String

```
rec_list {{process info string 0}, {process info string 1},...}
```

Where each process info string is the same as the result string of the **sysreq_info_proc** Tcl command extension.

Set _cerrno

Yes

sys_reqinfo_routername

Queries the router name.

Syntax

```
sys_reqinfo_routername
```

Arguments

None

Result String

```
routername %s
```

Where routername is the name of the router.

Set _cerrno

Yes

sys_reqinfo_snmp

Queries the value of the entity specified by a Simple Network Management Protocol (SNMP) object ID.

Syntax

```
sys_reqinfo_snmp oid ? get_type exact|next
```

Arguments

oid	(Mandatory) SNMP OID in dot notation (for example, 1.3.6.1.2.1.2.1.0).
get_type	(Mandatory) Type of SNMP get operation that needs to be applied to the specified oid. If the get_type is “exact,” the value of the specified oid is retrieved; if the get_type is “next,” the value of the lexicographical successor to the specified oid is retrieved.

Result String

```
oid {%s} value {%s}
```

oid	SNMP OID.
value	Value string of the associated SNMP data element.

Set _cerrno

Yes

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

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

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

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

```
(_cerr_sub_err = 37)    FH_ENOSNMPDATA  (can't retrieve data from SNMP)
```

This error means that there was no data for the SNMP object type.

```
(_cerr_sub_err = 51)    FH_ESTATSTYP  (invalid statistics data type)
```

This error means that the SNMP statistics data type was invalid.

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

This error means that the event detector was unavailable.

sys_reqinfo_syslog_freq

Queries the frequency information of all syslog events.

Syntax

```
sys_reqinfo_syslog_freq
```

Arguments

None

Result String

```
rec_list {{event frequency string 0}, {log freq str 1}, ...}
```

Where each event frequency string is:

```
time_sec %ld time_msec %ld match_count %u raise_count %u occurs %u
period_sec %ld period_msec %ld pattern {%s}
```

time_sec time_msec	Seconds and milliseconds in POSIX timer units since January 1, 1970, which represent the time the last event was raised.
match_count	Number of times that a syslog message matches the pattern specified by this syslog event specification since event registration.
raise_count	Number of times that this syslog event was raised.
occurs	Number of occurrences needed in order to raise the event; if not specified, the event is raised on the first occurrence.
period_sec period_msec	Number of occurrences must occur within this number of POSIX timer units in order to raise the event; if not specified, the period check does not apply.
pattern	Regular expression used to perform syslog message pattern matching.

Set _cerrno

Yes

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

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

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

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

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

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

```
(_cerr_sub_err = 45)    FH_ESEQNUM  (sequence or workset number out of sync)
```

This error means that the event detector sequence or workset number was invalid.

```
(_cerr_sub_err = 46)    FH_EREGEMPTY  (registration list is empty)
```

This error means that the event detector registration list was empty.

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

This error means that the event detector was unavailable.

sys_reqinfo_syslog_history

Queries the history of the specified syslog message.

Syntax

```
sys_reqinfo_syslog_history
```

Arguments

None

Result String

```
rec_list {{log hist string 0}, {log hist str 1}, ...}
```

Where each log hist string is:

```
time_sec %ld time_msec %ld msg {%s}
```

time_sec time_msec	Seconds and milliseconds since January 1, 1970, which represent the time the message was logged.
msg	Syslog message.

Set _cerrno

Yes

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

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

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

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

```
(_cerr_sub_err = 44)    FH_EHISTEMPTY  (history list is empty)
```

This error means that the history list was empty.

```
(_cerr_sub_err = 45)    FH_ESEQNUM  (sequence or workset number out of sync)
```

This error means that the event detector sequence or workset number was invalid.

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

This error means that the event detector was unavailable.

EEM Library Debug Command Extensions

cli_debug

Prints a command-line interface (CLI) debug statement to syslog. This Tcl command extension is used to print a CLI debug statement to syslog if the **debug event manager tcl cli_library** Cisco IOS CLI command is in effect.

Syntax

```
cli_debug spec_string debug_string
```

Arguments

spec_string	(Mandatory) The spec_string argument is used to indicate the type of debug statement.
debug_string	(Mandatory) The debug_string argument is used to indicate the debugging text.

Result String

None

Set _cerrno

No

smtp_debug

Prints a a Simple Mail Transfer Protocol (SMTP) debug statement to syslog. This Tcl command extension prints a SMTP debug statement to syslog if the **debug event manager tcl smtp_library** Cisco IOS command-line interface (CLI) command is in effect.

Syntax

```
smtp_debug spec_string debug_string
```

Arguments

spec_string	(Mandatory) The spec_string argument is used to indicate the type of debug statement.
debug_string	(Mandatory) The debug_string argument is used to indicate the debugging text.

Result String

None

Set _cerrno

No

SMTP Library Command Extensions

All Simple Mail Transfer Protocol (SMTP) library command extensions belong to the ::cisco::lib namespace.

To use this library, the user needs to provide an e-mail template file. The template file can include Tcl global variables so that the e-mail service and the e-mail text can be configured through the **event manager environment Cisco IOS** command-line interface (CLI) configuration command. There are commands in this library to substitute the global variables in the e-mail template file and to send the desired e-mail context with the To address, CC address, From address, and Subject line properly configured using the configured e-mail server.

E-Mail Template

The e-mail template file has the following format:

Note Based on RFC 2554, the SMTP e-mail server name—Mailservername— can be in any one of the following template formats: username:password@host, username@host, or host.

```
Mailservername:<space><the list of candidate SMTP server addresses>
From:<space><the e-mail address of sender>
To:<space><the list of e-mail addresses of recipients>
Cc:<space><the list of e-mail addresses that the e-mail will be copied to>
Sourceaddr:<space><the IP addresses of the recipients>
Subject:<subject line>
<a blank line>
<body>
```

Note Note that the template normally includes Tcl global variables to be configured.

Below is a sample e-mail template file:

```
Mailservername: $_email_server
From: $_email_from
To: $_email_to
Cc: $_email_cc
Sourceaddr: $_email_ipaddr
Subject: From router $routername: Process terminated

process name: $process_name
subsystem: $sub_system
exit status: $exit_status
respawn count: $respawn_count
```

Exported Tcl Command Extensions

- smtp_send_email, page 202
- smtp_subst, page 203

smtp_send_email

Given the text of an e-mail template file with all global variables already substituted, sends the e-mail out using Simple Mail Transfer Protocol (SMTP). The e-mail template specifies the candidate mail server addresses, To addresses, CC addresses, From address, subject line, and e-mail body.

Note A list of candidate e-mail servers can be provided so that the library will try to connect the servers on the list one by one until it can successfully connect to one of them.

Syntax

```
smtp_send_email text
```

Arguments

text	(Mandatory) The text of an e-mail template file with all global variables already substituted.

Result String

None

Set _cerrno

- Wrong 1st line format—Mailservername:list of server names.
- Wrong 2nd line format—From:from-address.
- Wrong 3rd line format—To:list of to-addresses.
- Wrong 4th line format—CC:list of cc-addresses.
- Error connecting to mail server:—$sock closed by remote server (where $sock is the name of the socket opened to the mail server).
- Error connecting to mail server:—$sock reply code is $k instead of the service ready greeting (where $sock is the name of the socket opened to the mail server; $k is the reply code of $sock).
- Error connecting to mail server:—cannot connect to all the candidate mail servers.
- Error disconnecting from mail server:—$sock closed by remote server (where $sock is the name of the socket opened to the mail server).

Sample Scripts

After all needed global variables in the e-mail template are defined:

```
if [catch {smtp_subst [file join $tcl_library email_template_sm]} result] {
    puts stderr $result
    exit 1
}
if [catch {smtp_send_email $result} result] {
    puts stderr $result
    exit 1
}
```

smtp_subst

Given an e-mail template file e-mail_template, substitutes each global variable in the file by its user-defined value. Returns the text of the file after substitution.

Syntax

```
smtp_subst e-mail_template
```

Arguments

e-mail_template	(Mandatory) Name of an e-mail template file in which global variables need to be substituted by a user-defined value. An example filename could be /disk0://example.template which represents a file named example.template in a top-level directory on an ATA flash disk in slot 0.

Result String

The text of the e-mail template file with all the global variables substituted.

Set _cerrno

- cannot open e-mail template file
- cannot close e-mail template file

CLI Library Command Extensions

All command-line interface (CLI) library command extensions belong to the `::cisco::eem` namespace.

This library provides users the ability to run CLI commands and get the output of the commands in Tcl. Users can use commands in this library to spawn an exec and open a virtual terminal channel to it, write the command to execute to the channel so that the command will be executed by exec, and read back the output of the command.

There are two types of CLI commands: interactive commands and non-interactive commands.

For interactive commands, after the command is entered, there will be a "Q&A" phase in which the router will ask for different user options, and the user is supposed to enter the answer for each question. Only after all the questions have been answered properly will the command run according to the user's options until completion.

For noninteractive commands, once the command is entered, the command will run to completion. To run different types of commands using an EEM script, different CLI library command sequences should be used, which are documented in the "Using the CLI Library to Run a Noninteractive Command" section on page 217 and in the "Using the CLI Library to Run an Interactive Command" section on page 217.

The vty lines are allocated from the pool of vty lines that are configured using the **line vty** CLI configuration command. EEM will use a vty line when a vty line is not being used by EEM and there are available vty lines. EEM will also use a vty line when EEM is already using a vty line and there are three or more vty lines available. Be aware that the connection will fail when fewer than three vty lines are available, preserving the remaining vty lines for Telnet use.

In Cisco IOS Release 12.4(22)T, and later releases, XML-PI support was introduced. For details about the XML-PI support, the new CLI library command extensions, and some examples of how to implement XML-PI, see "CLI Library XML-PI Support" section on page 220.

Exported Tcl Command Extensions

cli_close

Closes the exec process and releases the vty and the specified channel handler connected to the command-line interface (CLI).

Syntax

```
cli_close fd tty_id
```

Arguments

fd	(Mandatory) The CLI channel handler.
tty_id	(Mandatory) The TTY ID returned from the **cli_open** command extension.

Result String

None

Set _cerrno

Cannot close the channel.

cli_exec

Writes the command to the specified channel handler to execute the command. Then reads the output of the command from the channel and returns the output.

Syntax

```
cli_exec fd cmd
```

Arguments

fd	(Mandatory) The command-line interface (CLI) channel handler.
cmd	(Mandatory) The CLI command to execute.

Result String

The output of the CLI command executed.

Set _cerrno

Error reading the channel.

cli_get_ttyname

Returns the real and pseudo TTY names for a given TTY ID.

Syntax

```
cli_get_ttyname tty_id
```

Arguments

tty_id	(Mandatory) The TTY ID returned from the **cli_open** command extension.

Result String

```
pty %s tty %s
```

Set _cerrno

None

cli_open

Allocates a vty, creates an EXEC command-line interface (CLI) session, and connects the vty to a channel handler. Returns an array including the channel handler.

Note Each call to **cli_open** initiates a Cisco IOS EXEC session that allocates a Cisco IOS vty line. The vty remains in use until the **cli_close** routine is called. The vty lines are allocated from the pool of vty lines that are configured using the **line vty** CLI configuration command. EEM will use a vty line when a vty line is not being used by EEM and there are available vty lines. EEM will also use a vty line when EEM is already using a vty line and there are three or more vty lines available. Be aware that the connection will fail when fewer than three vty lines are available, preserving the remaining vty lines for Telnet use

Syntax

```
cli_open
```

Arguments

None

Result String

```
"tty_id {%s} pty {%d} tty {%d} fd {%d}"
```

Event Type	Description
tty_id	TTY ID.
pty	PTY device name.
tty	TTY device name.
fd	CLI channel handler.

Set _cerrno

- Cannot get pty for EXEC.
- Cannot create an EXEC CLI session.
- Error reading the first prompt.

cli_read

Reads the command output from the specified command-line interface (CLI) channel handler until the pattern of the router prompt occurs in the contents read. Returns all the contents read up to the match.

Syntax

```
cli_read fd
```

Arguments

fd	(Mandatory) The CLI channel handler.

Result String

All the contents read.

Set _cerrno

Cannot get router name.

Note This Tcl command extension will block waiting for the router prompt to show up in the contents read.

cli_read_drain

Reads and drains the command output of the specified command-line interface (CLI) channel handler. Returns all the contents read.

Syntax

```
cli_read_drain fd
```

Arguments

fd	(Mandatory) The CLI channel handler.

Result String

All the contents read.

Set _cerrno

None

cli_read_line

Reads one line of the command output from the specified command-line interface (CLI) channel handler. Returns the line read.

Syntax

```
cli_read_line fd
```

Arguments

fd	(Mandatory) The CLI channel handler.

Result String

The line read.

Set _cerrno

None

Note This Tcl command extension will block waiting for the end of line to show up in the contents read.

cli_read_pattern

Reads the command output from the specified command-line interface (CLI) channel handler until the pattern that is to be matched occurs in the contents read. Returns all the contents read up to the match.

Note The pattern matching logic attempts a match by looking at the command output data as it is delivered from the Cisco IOS command. The match is always done on the most recent 256 characters in the output buffer unless there are fewer characters available, in which case the match is done on fewer characters. If more than 256 characters in the output buffer are required for the match to succeed, the pattern will not match.

Syntax

```
cli_read_pattern fd ptn
```

Arguments

fd	(Mandatory) The CLI channel handler.
ptn	(Mandatory) The pattern to be matched when reading the command output from the channel.

Result String

All the contents read.

Set _cerrno

None

Note This Tcl command extension will block waiting for the specified pattern to show up in the contents read.

cli_run

Iterates over the items in the clist and assumes that each one is a command-line-interface (CLI) command to be executed in the enable mode. On success, returns the output of all executed commands and on failure, returns error from the failure.

Syntax

```
cli_run clist
```

Arguments

clist	(Mandatory) The list of commands to be executed.

Result String

Output of all the commands that are executed or an error message.

Set _cerrno

None.

Sample Usage

The following example shows how to use the **cli_run** command extension.

```
set clist [list {sh run} {sh ver} {sh event man pol reg}]
cli_run { clist }
```

cli_run_interactive

Provides a sublist to the clist which has four items. On success, returns the output of all executed commands and on failure, returns error from the failure. Also uses arrays when possible as a way of making things easier to read later by keeping expect and reply separated.

Syntax

```
cli_run_interactive clist
```

Arguments

clist	(Mandatory) Sublist which has four items and each item has four subitems: • command – expect – an expected question – reply – reply to this question • a command to run – expect – an expected question – reply – reply to this question • responses – expect – an expected question – reply – reply to this question • a list of what to expect and what to reply. – expect – an expected question – reply – reply to this question

Result String

Output of all the commands that are executed or an error message.

Set _cerrno

None.

Sample Usage

The following example shows how to use the cli_ru_ interactive command extension.

```
set cmd1 "first command"
set cmd1_exp1 {[confirm]}
```

```
set cmd1_rep1 {y}
set cmd1_response [list [list expect $cmd1_exp1 reply $cmd1_rep1]]

set cmd2 "second command"
set cmd2_exp1 {save config}
set cmd2_rep1 {no}
set cmd2_exp2 {[confirm]}
set cmd2_rep2 {y}
set cmd2_response [list [list expect $cmd2_exp1 reply $cmd2_rep1] [list expect $cmd2_exp2
reply $cmd2_rep2]]

set cmd3 "third command"
set cmd3_exp1 {are you sure}
set cmd3_rep1 {yes}
set cmd3_exp2 {destination file}
set cmd3_rep2 {test.txt}
set cmd2_response [list [list expect $cmd3_exp1 reply $cmd3_rep1] [list expect $cmd3_exp2
reply $cmd3_rep2]]

set clist [list " command $cmd1 responses $cmd1_response" " command $cmd2 responses
$cmd2_response" " command $cmd3 responses $cmd3_response"]
cli_run_interactive { clist }
```

cli_write

Writes the command that is to be executed to the specified CLI channel handler. The CLI channel handler executes the command.

Syntax

```
cli_write fd cmd
```

Arguments

fd	(Mandatory) The CLI channel handler.
cmd	(Mandatory) The CLI command to execute.

Result String

None

Set _cerrno

None

Sample Usage

As an example, use configuration CLI commands to bring up Ethernet interface 1/0:

```
if [catch {cli_open} result] {
puts stderr $result
exit 1
} else {
array set cli1 $result
}
if [catch {cli_exec $cli1(fd) "en"} result] {
puts stderr $result
exit 1
}
if [catch {cli_exec $cli1(fd) "config t"} result] {
puts stderr $result
exit 1
}
if [catch {cli_exec $cli1(fd) "interface Ethernet1/0"} result] {
puts stderr $result
exit 1
}
if [catch {cli_exec $cli1(fd) "no shut"} result] {
puts stderr $result
exit 1
}
if [catch {cli_exec $cli1(fd) "end"} result] {
puts stderr $result
exit 1
}
if [catch {cli_close $cli1(fd) $cli1(tty_id)} } result] {
puts stderr $result
exit 1
```

Using the CLI Library to Run a Noninteractive Command

To run a noninteractive command, use the **cli_exec** command extension to issue the command, and then wait for the complete output and the router prompt. For example, the following shows the use of configuration CLI commands to bring up Ethernet interface 1/0:

```
if [catch {cli_open} result] {
error $result $errorInfo
} else {
set fd $result
}
if [catch {cli_exec $fd "en"} result] {
error $result $errorInfo
}
if [catch {cli_exec $fd "config t"} result] {
error $result $errorInfo
}
if [catch {cli_exec $fd "interface Ethernet1/0"} result] {
error $result $errorInfo
}
if [catch {cli_exec $fd "no shut"} result] {
error $result $errorInfo
}
if [catch {cli_exec $fd "end"} result] {
error $result $errorInfo
}
if [catch {cli_close $fd} result] {
error $result $errorInfo
}
```

Using the CLI Library to Run an Interactive Command

To run interactive commands, three phases are needed:

- Phase 1: Issue the command using the **cli_write** command extension.
- Phase 2: Q&A Phase. Use the **cli_read_pattern** command extension to read the question (the regular pattern that is specified to match the question text) and the **cli_write** command extension to write back the answers alternately.
- Phase 3: Noninteractive phase. All questions have been answered, and the command will run to completion. Use the **cli_read** command extension to wait for the complete output of the command and the router prompt.

For example, use CLI commands to do squeeze bootflash: and save the output of this command in the Tcl variable cmd_output.

```
if [catch {cli_open} result] {
error $result $errorInfo
} else {
array set cli1 $result
}
if [catch {cli_exec $cli1(fd) "en"} result] {
error $result $errorInfo
}

# Phase 1: issue the command
if [catch {cli_write $cli1(fd) "squeeze bootflash:"} result] {
error $result $errorInfo
}

# Phase 2: Q&A phase
# wait for prompted question:
# All deleted files will be removed. Continue? [confirm]
if [catch {cli_read_pattern $cli1(fd) "All deleted"} result] {
```

```
error $result $errorInfo
}
# write a newline character
if [catch {cli_write $cli1(fd) "\n"} result] {
error $result $errorInfo
}
# wait for prompted question:
# Squeeze operation may take a while. Continue? [confirm]
if [catch {cli_read_pattern $cli1(fd) "Squeeze operation"} result] {
error $result $errorInfo
}
# write a newline character
if [catch {cli_write $cli1(fd) "\n"} result] {
error $result $errorInfo
}

# Phase 3: noninteractive phase
# wait for command to complete and the router prompt
if [catch {cli_read $cli1(fd) } result] {
error $result $errorInfo
} else {
set cmd_output $result
}
if [catch {cli_close $cli1(fd) $cli1(tty_id)} result] {
error $result $errorInfo
}
```

The following example causes a router to be reloaded using the CLI **reload** command. Note that the EEM **action_reload** command accomplishes the same result in a more efficient manner, but this example is presented to illustrate the flexibility of the CLI library for interactive command execution.

```
# 1. execute the reload command
if [catch {cli_open} result] {
    error $result $errorInfo
} else {
    array set cli1 $result
}
if [catch {cli_exec $cli1(fd) "en"} result] {
    error $result $errorInfo
}
if [catch {cli_write $cli1(fd) "reload"} result] {
    error $result $errorInfo
} else {
    set cmd_output $result
}
if [catch {cli_read_pattern $cli1(fd) ".*(System configuration has been modified. Save\\\?
\\\[yes/no\\\]: )"} result] {
    error $result $errorInfo
} else {
    set cmd_output $result
}
if [catch {cli_write $cli1(fd) "no"} result] {
    error $result $errorInfo
} else {
    set cmd_output $result
}
if [catch {cli_read_pattern $cli1(fd) ".*(Proceed with reload\\\? \\\[confirm\\\])"}
result] {
    error $result $errorInfo
} else {
    set cmd_output $result
}
```

```
if [catch {cli_write $cli1(fd) "y"} result] {
    error $result $errorInfo
} else {
    set cmd_output $result
}
if [catch {cli_close $cli1(fd) $cli1(tty_id)} result] {
    error $result $errorInfo
}
```

CLI Library XML-PI Support

XML Programmatic Interface (XML-PI) was introduced in Cisco IOS Release 12.4(22)T. XML-PI provides a programmable interface which encapsulates IOS command-line interface (CLI) show commands in XML format in a consistent way across different Cisco products. Customers using XML-PI will be able to parse IOS show command output from within Tcl scripts using well-known keywords instead of having to depend on the use of regular expression support to "screen-scrape" output.

The benefit of using the XML-PI command extensions is to facilitate the extraction of specific output information that is generated using a CLI **show** command. Most show commands return many fields within the output and currently a regular expression has to be used to extract specific information information that may appear in the middle of a line. XML-PI support provides a set of Tcl library functions to facilitate the parsing of output from the IOS CLI format extension in the form of:

```
show <show-command> | format {spec-file}
```

where a spec-file is a concatenation of all Spec File Entries (SFE) for each **show** command currently supported. As part of the XML-PI project a default spec-file will be included in the IOS Release 12.4(22)T images. The default spec-file will have a small set of commands and the SFE for the commands will have a subset of the possible tags. If no spec-file is provided with the format command, the default spec-file is used.

For examples of implementing the XML-PI feature, see the "Sample Usage of the XML-PI feature" section on page 224.

For more general details about XML-PI, see the "XML-PI" chapter of the *Cisco IOS Network Management Configuration Guide*.

The following Tcl command extensions were introduced to support XML-PI:

xml_pi_exec

Writes the XML-PI command specified using the cmd argument to the channel whose handler is specified using the fd argument and the spec-file specified by the spec_file argument to execute the command. The raw XML output data of the command is then read from the channel and the XML output is returned.

Syntax

```
xml_pi_show fd cmd [spec_file]
```

Arguments

fd	(Mandatory) The CLI library file descriptor obtained from cli_open.
cmd	(Mandatory) IOS show command.
spec_file	(Optional) IOS CLI show command spec_file.

Result String

Result of IOS show command in XML format.

Set _cerrno

Possible error raised:

1. error reading the channel

xml_pi_parse

Processes the XML show command raw output passed into this function as xml_data and retrieve those fields that are specified by xml_tags_list. The following processing occurs:

Step 1: The XML tag list is validated as a Tcl list. An XML tag can be specified as the low order XML tag name or as a fully qualified XML tag name in case the low order name is ambiguous for a given command.

Example tags:
<Interface>
<ShowIpInterfaceBrief><IPInterfaces><entry><Interface>

Step 2: The xml_data is validated as valid XML and parsed into an XML parse tree.

Step 3: A walk is made through the XML parse tree and each tag is compared with entries in the XML tag list. When a match occurs it is determined if the tag name matches a Tcl procedure defined within the current Tcl scope. If so, that Tcl procedure will be called with the current result. If not, the tag name and the data associated with that tag name will be appended to the current result.

Syntax

```
xml_pi_parse fd xml_show_cmd_output xml_tags_list
```

Arguments

fd	(Mandatory) The CLI library file descriptor obtained from cli_open.
xml_show_cmd_output	(Mandatory) Output of xml_pi_show command extension in xml format.
xml_tags_list	(Mandatory) List of interesting tags.

Result String

Data in a Tcl array indexed by XML tag name.

Note The current result is reset after Tcl procedure calls.

Set _cerrno

Possible errors raised:

1. error splitting the XML tags list
2. null XML tag list specified
3. XML tag tree exceeds 20 levels
4. called Tcl procedure returned an error
5. memory allocation failure
6. XML parse failure
7. failed to create XML domain

xml_pi_read

Reads the XML-PI command output (from the specified show command) from the CLI channel whose handler is given by the file descriptor until the pattern of the router prompt occurs in the contents that are read. Returns all the contents read up to the match in XML format.

Syntax

```
xml_pi_read fd
```

Arguments

fd	(Mandatory) The CLI library file descriptor obtained from cli_open.

Result String

All the contents that are read in XML format.

Set _cerrno

Possible errors raised:

1. cannot get router name
2. command error

xml_pi_write

Writes the XML-PI command specified using the cmd argument to the channel whose handler is given by the fd argument and the spec file specified by the spec_file argument.

Syntax

```
xml_pi_write fd cmd spec_file
```

Arguments

fd	(Mandatory) The CLI library file descriptor obtained from cli_open.
cmd	(Mandatory) IOS show command.
spec_file	(Optional) IOS CLI show command spec_file.

Result String

None

Set _cerrno

None

Sample Usage of the XML-PI feature

The following EEM policy (sample.tcl) presents one example that illustrates five different implementations of the new EEM XML-PI functionality. The odm spec-file (required for Example 2) follows this policy.

```
::cisco::eem::event_register_none maxrun 60
namespace import ::cisco::eem::*
namespace import ::cisco::lib::*
# open the cli_lib.tcl channel
if [catch {cli_open} result] {
error $result $errorInfo
} else {
array set cli1 $result
}
# enter "enable" privilege mode
if [catch {cli_exec $cli1(fd) "en"} result] {
error $result $errorInfo
}
# Example 1:
#
# Detect if XML-PI is present in this image
# Invoke xml_pi_exec with the default spec file for the "show inventory"
# command. After the command executes $result contains the raw XML data if
# the command is successful.
if [catch {xml_pi_exec $cli1(fd) "show inventory" ""} result] {
puts "Example 1: XML-PI support is not present in this image - exiting"
exit
} else {
puts "Example 1: XML-PI support is present in this image"
}
# Example 2:
#
# In the next example we demonstrate how to extract two data elements
# from the "show version" command using the specified XML-PI spec file.
# The raw output from this command is as follows:
#
```

```
# router#show version | format disk2:speceemtest.odm
# <?xml version="1.0" encoding="UTF-8"?>
# <ShowVersion>
# <Version>12.4(20071029:194217)</Version>
# <Compiled>Thu 08-Nov-07 11:28</Compiled>
# <ROM>System Bootstrap, Version 12.2(20030826:190624) [BLD-npeg1_rommon_r11 102],
DEVELOPMENT</ROM>
# <uptime>17 minutes</uptime>
# <processor>NPE-G1</processor>
# <bytesofmemory>983040K/65536K</bytesofmemory>
# <CPU>700MHz</CPU>
# <L2Cache>0.2</L2Cache>
# <GigabitEthernetinterfaces>3</GigabitEthernetinterfaces>
# <bytesofNVRAM>509K</bytesofNVRAM>
# <bytesofATAPCMCIAcard>125952K</bytesofATAPCMCIAcard>
# <Sectorsize>512 bytes</Sectorsize>
# <bytesofFlashinternalSIMM>16384K</bytesofFlashinternalSIMM>
# <Configurationregister>0x2100</Configurationregister>
# </ShowVersion>
#
# Invoke xml_pi_exec with the spec file "disk2:speceemtest.odm" for the
# "show version" command. After the command executes $result contains
# the raw XML data.
if [catch {xml_pi_exec $cli1(fd) "show version" "disk2:speceemtest.odm"} result] {
error $result $errorInfo
} else {
# Pass the raw XML data to the xml_pi_parse routine to extract fields
# of interest:
# we ask that only the <processor> and <CPU> fields be returned.
array set xml_result [xml_pi_parse $cli1(fd) $result "<processor> <CPU>"]
puts "Example 2: Processor is $xml_result(<processor>) CPU is $xml_result(<CPU>)"
}
# Example 3:
#
# In the next example we demonstrate how to extract two data elements
# from the multi-record "show inventory" command using the default built-in
# XML-PI spec file. Sample raw output from this command is as follows:
#
# router#show inventory | format
# <?xml version="1.0" encoding="UTF-8"?>
# <ShowInventory>
# <SpecVersion>built-in</SpecVersion>
# <InventoryEntry>
# <ChassisName>&quot;Chassis&quot;</ChassisName>
# <Description>&quot;Cisco 7206VXR, 6-slot chassis&quot;</Description>
# <PID>CISCO7206VXR</PID>
# <VID>
# </VID>
# <SN>31413378 </SN>
# </InventoryEntry>
# <InventoryEntry>
# <ChassisName>&quot;NPE-G1 0&quot;</ChassisName>
# <Description>&quot;Cisco 7200 Series Network Processing Engine
NPE-G1&quot;</Description>
# <PID>NPE-G1</PID>
# <VID>
# </VID>
# <SN>31493825 </SN>
# </InventoryEntry>
# <InventoryEntry>
# <ChassisName>&quot;disk2&quot;</ChassisName>
# <Description>&quot;128MB Compact Flash Disk for NPE-G1&quot;</Description>
# <PID>MEM-NPE-G1-FLD128</PID>
# <VID>
```

```
# </VID>
# <SN>NAME: &quot;module 1&quot;</SN>
# </InventoryEntry>
# <InventoryEntry>
# <ChassisName>&quot;module 1&quot;</ChassisName>
# <Description>&quot;Dual Port FastEthernet (RJ45)&quot;</Description>
# <PID>PA-2FE-TX</PID>
# <VID>
# </VID>
# <SN>JAE0827NGKX</SN>
# </InventoryEntry>
# <InventoryEntry>
# <ChassisName>&quot;Power Supply 2&quot;</ChassisName>
# <Description>&quot;Cisco 7200 AC Power Supply&quot;</Description>
# <PID>PWR-7200-AC</PID>
# <VID>
# </VID>
# </InventoryEntry>
# </ShowInventory>
#
# Define a procedure to be called every time the <InventoryEntry> tag
# is processed. Since this tag precedes each new output record, the data
# that is passed into this procedure contains the fields that have been
# requested via xml_pi_parse since the previous time this procedure was
# called.
proc <InventoryEntry> {xml_line} {
global num
# The first time that this function is called there is no data and
# xml_line will be null.
if [string length $xml_line] {
array set xml_result $xml_line
incr num
set output [format "Example 3: Item %2d %-18s %s" \
$num $xml_result(<PID>) $xml_result(<Description>)]
puts $output
}
}
set num 0
# Invoke xml_pi_exec with the default built-in spec file for the
# "show inventory" command. After the command executes $result contains
# the raw XML data.
if [catch {xml_pi_exec $cli1(fd) "show inventory"} result] {
error $result $errorInfo
} else {
# Pass the raw XML data to the xml_pi_parse routine to extract fields
# of interest:
# we ask that only the <PID> and <Description> fields be returned.
# If an XML tag name is requested and a Tcl proc exists with that name,
# the Tcl proc will be called every time that tag is encountered in the
# output data. Specify the <InventoryEntry> tag and define the proc
# before executing the xml_pi_parse statement.
array set xml_result [xml_pi_parse $cli1(fd) $result \
"<InventoryEntry> <PID> <Description>"]
# Display the data from the last record.
incr num
set output [format "Example 3: Item %2d %-18s %s" \
$num $xml_result(<PID>) $xml_result(<Description>)]
puts $output
}
# Example 4:
#
# In the next example we demonstrate how to extract two data elements
# from the multi-record "show ip interface brief" command using the default
# built-in XML-PI spec file. Sample raw output from this command is as
```

```
# follows:
#
# router#show ip interface brief | format
# <?xml version="1.0" encoding="UTF-8"?>
# <ShowIpInterfaceBrief>
# <SpecVersion>built-in</SpecVersion>
# <IPInterfaces>
# <entry>
# <Interface>GigabitEthernet0/1</Interface>
# <IP-Address>172.19.209.34</IP-Address>
# <OK>YES</OK>
# <Method>NVRAM</Method>
# <Status>up</Status>
# <Protocol>up</Protocol>
# </entry>
# <entry>
# <Interface>GigabitEthernet0/2</Interface>
# <IP-Address>unassigned</IP-Address>
# <OK>YES</OK>
# <Method>NVRAM</Method>
# <Status>administratively down</Status>
# <Protocol>down</Protocol>
# </entry>
# <entry>
# <Interface>GigabitEthernet0/3</Interface>
# <IP-Address>unassigned</IP-Address>
# <OK>YES</OK>
# <Method>NVRAM</Method>
# <Status>administratively down</Status>
# <Protocol>down</Protocol>
# </entry>
# <entry>
# <Interface>FastEthernet1/0</Interface>
# <IP-Address>unassigned</IP-Address>
# <OK>YES</OK>
# <Method>NVRAM</Method>
# <Status>administratively down</Status>
# <Protocol>down</Protocol>
# </entry>
# <entry>
# <Interface>FastEthernet1/1</Interface>
# <IP-Address>unassigned</IP-Address>
# <OK>YES</OK>
# <Method>NVRAM</Method>
# <Status>administratively down</Status>
# <Protocol>down</Protocol>
# </entry>
# </IPInterfaces>
# </ShowIpInterfaceBrief>
#
# Define a procedure to be called every time the fully qualified name
# <ShowIpInterfaceBrief><IPInterfaces><entry> tag is processed. Since
# this tag precedes each new output record, the data that is passed into
# this procedure contains the fields that have been requested via
# xml_pi_parse since the previous time this procedure was called.
proc <ShowIpInterfaceBrief><IPInterfaces><entry> {xml_line} {
global num
# The first time that this function is called there is no data and
# xml_line will be null.
if [string length $xml_line] {
array set xml_result $xml_line
incr num
set output [format "Example 4: Interface %2d %-30s %s" \
$num $xml_result(<Interface>) $xml_result(<Status>)]
```

```
puts $output
} else {
puts "Example 4: Display All Interfaces"
}
}
set num 0
# Invoke xml_pi_exec with the default built-in spec file for the
# "show ip interface brief" command. After the command executes $result
# contains the raw XML data.
if [catch {xml_pi_exec $cli1(fd) "show ip interface brief"} result] {
error $result $errorInfo
} else {
# Pass the raw XML data to the xml_pi_parse routine to extract fields
# of interest:
# we ask that only the <Interface> and <Status> fields be returned.
# If an XML tag name is requested and a Tcl proc exists with that name,
# the Tcl proc will be called every time that tag is encountered in the
# output data. Specify the <entry> tag and define the proc
# before executing the xml_pi_parse statement.
array set xml_result [xml_pi_parse $cli1(fd) $result \
"<ShowIpInterfaceBrief><IPInterfaces><entry> <Interface> <Status>"]
# Display the data from the last record.
incr num
set output [format "Example 4: Interface %2d %-30s %s" \
$num $xml_result(<Interface>) $xml_result(<Status>)]
puts $output
}
# Example 5:
#
# In the next example we demonstrate how to extract two data elements
# from the multi-record "show ip interface brief" command using the default
# built-in XML-PI spec file. Sample raw output from this command is as
# follows:
#
# router#show ip interface brief | format
# <?xml version="1.0" encoding="UTF-8"?>
# <ShowIpInterfaceBrief>
# <SpecVersion>built-in</SpecVersion>
# <IPInterfaces>
# <entry>
# <Interface>GigabitEthernet0/1</Interface>
# <IP-Address>172.19.209.34</IP-Address>
# <OK>YES</OK>
# <Method>NVRAM</Method>
# <Status>up</Status>
# <Protocol>up</Protocol>
# </entry>
# <entry>
# <Interface>GigabitEthernet0/2</Interface>
# <IP-Address>unassigned</IP-Address>
# <OK>YES</OK>
# <Method>NVRAM</Method>
# <Status>administratively down</Status>
# <Protocol>down</Protocol>
# </entry>
# <entry>
# <Interface>GigabitEthernet0/3</Interface>
# <IP-Address>unassigned</IP-Address>
# <OK>YES</OK>
# <Method>NVRAM</Method>
# <Status>administratively down</Status>
# <Protocol>down</Protocol>
# </entry>
# <entry>
```

```
# <Interface>FastEthernet1/0</Interface>
# <IP-Address>unassigned</IP-Address>
# <OK>YES</OK>
# <Method>NVRAM</Method>
# <Status>administratively down</Status>
# <Protocol>down</Protocol>
# </entry>
# <entry>
# <Interface>FastEthernet1/1</Interface>
# <IP-Address>unassigned</IP-Address>
# <OK>YES</OK>
# <Method>NVRAM</Method>
# <Status>administratively down</Status>
# <Protocol>down</Protocol>
# </entry>
# </IPInterfaces>
# </ShowIpInterfaceBrief>
#
# Note: This example is the same as Example 4 with the exception that
# the new record procedure is called by the un-qualified tag name. The
# ability to specify the un-qualified tag names is simplier but only works
# if the un-qualified name is used once per Tcl program. In this example
# the unqualified new record tag name is "<entry>" which is a very
# common name in the Cisco spec file.
# Define a procedure to be called every time the <entry> tag
# is processed. Since this tag precedes each new output record, the data
# that is passed into this procedure contains the fields that have been
# requested via xml_pi_parse since the previous time this procedure was
# called.
proc <entry> {xml_line} {
global num
# The first time that this function is called there is no data and
# xml_line will be null.
if [string length $xml_line] {
array set xml_result $xml_line
incr num
if ([string equal $xml_result(<Status>) "up"]) {
set output [format "Example 5: Interface %2d %-30s %s" \
$num $xml_result(<Interface>) $xml_result(<Status>)]
puts $output
}
} else {
puts "Example 5: Display All Interfaces That Are Up"
}
}
set num 0
# Invoke xml_pi_exec with the default built-in spec file for the
# "show ip interface brief" command. After the command executes $result
# contains the raw XML data.
if [catch {xml_pi_exec $cli1(fd) "show ip interface brief"} result] {
error $result $errorInfo
} else {
# Pass the raw XML data to the xml_pi_parse routine to extract fields
# of interest:
# we ask that only the <Interface> and <Status> fields be returned.
# If an XML tag name is requested and a Tcl proc exists with that name,
# the Tcl proc will be called every time that tag is encountered in the
# output data. Specify the <entry> tag and define the proc
# before executing the xml_pi_parse statement.
array set xml_result [xml_pi_parse $cli1(fd) $result \
"<entry> <Interface> <Status>"]
# Display the data from the last record.
incr num
if ([string equal $xml_result(<Status>) "up"]) {
```

```
set output [format "Example 5: Interface %2d %-30s %s" \
$num $xml_result(<Interface>) $xml_result(<Status>)]
puts $output
}
}
```

Sample XML-PI spec eemtest.odm ODM File:

```
###
show version
<?xml version='1.0' encoding='utf-8'?>
<ODMSpec>
<Command>
<Name>show version</Name>
</Command>
<OS>ios</OS>
<DataModel>
<Container name="ShowVersion">
<Property name="Version" distance = "1.0" length = "1" type = "IpAddress"/>
<Property name="Technical Support" distance = "1.0" length = "1" type = "IpAddress"/>
<Property name="Compiled" distance = "1.0" length = "3" type = "String"/>
<Property name="ROM" distance = "1.0" length = "7" type = "IpAddress"/>
<Property name="uptime" distance = "2" length = "8" type = "String"/>
<Property name="image" distance = "4" length = "1" type = "IpAddress"/>
<Property name="processor" distance = "-1" length = "1" type = "String"/>
<Property name="bytes of memory" distance = "-1" length = "1" type = "Port"/>
<Property name="CPU" distance = "2" length = "1" end-delimiter = "," type = "String"/>
<Property name="L2 Cache" distance = "-2" length = "1" end-delimiter = "," type =
"String"/>
<Property name="Gigabit Ethernet interfaces" distance = "-1" length = "1" type =
"Integer"/>
<Property name="bytes of NVRAM" distance = "-1" length = "1" type = "String"/>
<Property name="bytes of ATA PCMCIA card" distance = "-1" length = "1" type = "String"/>
<Property name="Sector size" distance = "1.0" length = "2" end-delimiter = ")" type =
"String"/>
<Property name="bytes of Flash internal SIMM" distance = "-1" length = "1" type =
"String"/>
<Property name="Configuration register" distance = "2" length = "1" type = "String"/>
</Container>
</DataModel>
</ODMSpec>
```

Example sample.tcl Run:

```
router#config t
Enter configuration commands, one per line. End with CNTL/Z.
router(config)#event manager policy sample.tcl
router(config)#end
router#
Oct 10 20:21:26: %SYS-5-CONFIG_I: Configured from console by console
router#event manager run sample.tcl
Example 1: XML-PI support is present in this image
Example 2: Processor is NPE-G1 CPU is 700MHz
Example 3: Item 1 CISCO7206VXR "Cisco 7206VXR, 6-slot chassis"
Example 3: Item 2 NPE-G1 "Cisco 7200 Series Network Processing Engine NPE-G1"
Example 3: Item 3 MEM-NPE-G1-FLD128 "128MB Compact Flash Disk for NPE-G1"
Example 3: Item 4 PA-2FE-TX "Dual Port FastEthernet (RJ45)"
Example 3: Item 5 PWR-7200-AC "Cisco 7200 AC Power Supply"
Example 4: Display All Interfaces
Example 4: Interface 1 GigabitEthernet0/1 up
Example 4: Interface 2 GigabitEthernet0/2 administratively down
Example 4: Interface 3 GigabitEthernet0/3 administratively down
Example 4: Interface 4 FastEthernet1/0 administratively down
Example 4: Interface 5 FastEthernet1/1 administratively down
```

```
Example 4: Interface 6 SSLVPN-VIF0 up
Example 5: Display All Interfaces That Are Up
Example 5: Interface 1 GigabitEthernet0/1 up
Example 5: Interface 6 SSLVPN-VIF0 up
```

Tcl Context Library Command Extensions

All the Tcl context library command extensions belong to the ::cisco::eem namespace.

Exported Commands

context_retrieve

Retrieves Tcl variable(s) identified by the given context name, and possibly the scalar variable name, the array variable name, and the array index. Retrieved information is automatically deleted.

Note Once saved information is retrieved, it is automatically deleted. If that information is needed by another policy, the policy that retrieves it (using the **context_retrieve** command extension) should also save it again (using the **context_save** command extension).

Syntax

```
context_retrieve ctxt [var] [index_if_array]
```

Arguments

ctxt	(Mandatory) Context name.
var	(Optional) Scalar variable name or array variable name. Defaults to a null string if this argument is not specified.
index_if_array	(Optional) The array index.

Note The index_if_array argument will be ignored when the var argument is a scalar variable.

If var is unspecified, retrieves the whole variable table saved in the context.

If var is specified and index_if_array is not specified, or if index_if_array is specified but var is a scalar variable, retrieves the value of var.

If var is specified, and index_if_array is specified, and var is an array variable, retrieves the value of the specified array element.

Result String

Resets the Tcl global variables to the state that they were in when the save was performed.

Set _cerrno

- A string displaying _cerrno, _cerr_sub_num, _cerr_sub_err, _cerr_posix_err, _cerr_str due to appl_reqinfo error.
- Variable is not in the context.

Sample Usage

The following examples show how to use the **context_save** and **context_retrieve** command extension functionality to save and retrieve data. The examples are shown in save and retrieve pairs.

Example 1: Save

If var is unspecified or if a pattern if specified, saves multiple variables to the context.

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

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

```
set testvara 123
set testvarb 345
set testvarc 789
if {[catch {context_save TESTCTX "testvar*"} errmsg]} {
      action_syslog msg "context_save failed: $errmsg"
} else {
      action_syslog msg "context_save succeeded"
}
```

Example 1: Retrieve

If var is unspecified, retrieves multiple variables from the context.

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

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

if {[catch {foreach {var value} [context_retrieve TESTCTX] {set $var $value}} errmsg]} {
      action_syslog msg "context_retrieve failed: $errmsg"
} else {
      action_syslog msg "context_retrieve succeeded"
}
if {[info exists testvara]} {
      action_syslog msg "testvara exists and is $testvara"
} else {
      action_syslog msg "testvara does not exist"
}
if {[info exists testvarb]} {
      action_syslog msg "testvarb exists and is $testvarb"
} else {
      action_syslog msg "testvarb does not exist"
}
if {[info exists testvarc]} {
      action_syslog msg "testvarc exists and is $testvarc"
} else {
      action_syslog msg "testvarc does not exist"
}
```

Example 2: Save

If var is specified, saves the value of var.

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

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

set testvar 123
if {[catch {context_save TESTCTX testvar} errmsg]} {
      action_syslog msg "context_save failed: $errmsg"
} else {
      action_syslog msg "context_save succeeded"
}
```

Example 2: Retrieve

If var is specified and index_if_array is not specified, or if index_if_array is specified but var is a scalar variable, retrieves the value of var.

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

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

```
if {[catch {set testvar [context_retrieve TESTCTX testvar]} errmsg]} {
      action_syslog msg "context_retrieve failed: $errmsg"
} else {
      action_syslog msg "context_retrieve succeeded"
}
if {[info exists testvar]} {
      action_syslog msg "testvar exists and is $testvar"
} else {
      action_syslog msg "testvar does not exist"
}
```

Example 3: Save

If var is specified, saves the value of var even if it is an array.

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

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

array set testvar "testvar1 ok testvar2 not_ok"
if {[catch {context_save TESTCTX testvar} errmsg]} {
      action_syslog msg "context_save failed: $errmsg"
} else {
      action_syslog msg "context_save succeeded"
}
```

Example 3: Retrieve

If var is specified, and index_if_array is not specified, and var is an array variable, retrieves the entire array.

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

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

if {[catch {array set testvar [context_retrieve TESTCTX testvar]} errmsg]} {
      action_syslog msg "context_retrieve failed: $errmsg"
} else {
      action_syslog msg "context_retrieve succeeded"
}
if {[info exists testvar]} {
      action_syslog msg "testvar exists and is [array get testvar]"
} else {
      action_syslog msg "testvar does not exist"
}
```

Example 4: Save

If var is specified, saves the value of var even if it is an array.

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

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

array set testvar "testvar1 ok testvar2 not_ok"
if {[catch {context_save TESTCTX testvar} errmsg]} {
      action_syslog msg "context_save failed: $errmsg"
} else {
      action_syslog msg "context_save succeeded"
}
```

Example 4: Retrieve

If var is specified, and index_if_array is specified, and var is an array variable, retrieves the specified array element value.

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

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

if {[catch {set testvar [context_retrieve TESTCTX testvar testvar1]} errmsg]} {
      action_syslog msg "context_retrieve failed: $errmsg"
} else {
      action_syslog msg "context_retrieve succeeded"
}
if {[info exists testvar]} {
      action_syslog msg "testvar exists and is $testvar"
} else {
      action_syslog msg "testvar doesn't exist"
}
```

context_save

Saves Tcl variables that match a given pattern in current and global namespaces with the given context name as identification. Use this Tcl command extension to save information outside of a policy. Saved information can be retrieved by a different policy using the **context_retrieve** command extension.

Note Once saved information is retrieved, it is automatically deleted. If that information is needed by another policy, the policy that retrieves it (using the **context_retrieve** command extension) should also save it again (using the **context_save** command extension).

Syntax

```
context_save ctxt [pattern]
```

Arguments

ctxt	(Mandatory) Context name.
pattern	(Optional) The glob-style pattern as used by the **string match** Tcl command. If this argument is not specified, the pattern defaults to the wildcard *. There are three constructs used in glob patterns: • * = all characters • ? = 1 character • [abc] = match one of a set of characters

Result String

None

Set _cerrno

A string displaying _cerrno, _cerr_sub_num, _cerr_sub_err, _cerr_posix_err, _cerr_str due to appl_setinfo error.

Sample Usage

For examples showing how to use the **context_save** and **context_retrieve** command extension functionality to save and retrieve data, see the "Sample Usage" section on page 233.

Feature Information for Writing Embedded Event Manager Policies Using Tcl

Table 16 lists the features in this module and provides links to specific configuration information. Only features that were introduced or modified in Cisco IOS Releases 12.3(14)T, 12.2(25)S, 12.0(26)S, 12.2(18)SXF4, 12.2(28)SB, 12.2(33)SRA, 12.2(33)SXH, 12.2(33)SXI, 12.4(20)T, 12.4(22)T, 15.0(1)M, 12.2(33)SRE or a later release appear in the table.

Not all commands may be available in your Cisco IOS software release. For release information about a specific command, see the command reference documentation.

Use Cisco Feature Navigator to find information about platform support and software image support. Cisco Feature Navigator enables you to determine which Cisco IOS and Catalyst OS software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Note Table 16 lists only the Cisco IOS software release that introduced support for a given feature in a given Cisco IOS software release train. Unless noted otherwise, subsequent releases of that Cisco IOS software release train also support that feature.

Table 16 Feature Information for Writing Embedded Event Manager Policies Using Tcl

Feature Name	Releases	Feature Configuration Information
Embedded Event Manager 2.1	12.3(14)T 12.2(18)SXF5 12.2(28)SB 12.2(33)SRA	This document was introduced to support the ability to create policies using Tool Command Language (Tcl) that was introduced in the Embedded Event Manager 2.1 feature in Cisco IOS Release 12.3(14)T.
Embedded Event Manager 2.1 (Software Modularity)	12.2(18)SXF4 Cisco IOS Software Modularity images	EEM 2.1 for Software Modularity images introduced the GOLD, system manager, and WDSysMon (Cisco IOS Software Modularity watchdog) event detectors and the ability to display Cisco IOS Software Modularity processes and process matrix. Six new sample policies were also introduced. The following sections provide information about this feature: • EEM Policy Tcl Command Extension Categories, page 3 • Displaying Software Modularity Process Reliability Metrics Using EEM, page 18 • Modifying the Sample EEM Policies, page 20 • EEM Policy Tcl Command Extension Reference, page 55 The following commands were introduced by this feature: **event gold**, **event process**, **show event manager metric process**. **Note** EEM 2.1 for Software Modularity images also supports the resource and RF event detectors introduced in EEM 2.2, but it does not support the enhanced object tracking event detector or the actions to read and set tracked objects.
Embedded Event Manager 2.2	12.4(2)T 12.2(31)SB3 12.2(33)SRB	EEM 2.2 introduced the enhanced object tracking, resource, and RF event detectors. The actions of reading and setting the state of a tracked object were also introduced. The following sections provide information about this, feature: • EEM Policy Tcl Command Extension Categories, page 3 • Modifying the Sample EEM Policies, page 20 • EEM Policy Tcl Command Extension Reference, page 55 The following commands were introduced or modified by this feature: **action track read**, **action track set**, **default-state**, **event resource**, **event rf**, **event track**, **show track**, **track stub-object**.
SNMP event detector delta environment variable[1]	12.4(11)T	A new SNMP event detector environment variable, delta_val, was introduced.

Table 16 Feature Information for Writing Embedded Event Manager Policies Using Tcl (continued)

Feature Name	Releases	Feature Configuration Information
Embedded Event Manager 2.3	12.2(33)SXH 12.2(33)SB	EEM 2.3 introduced some new features relative to the Generic Online Diagnostics (GOLD) Event Detector on the Cisco Catalyst 6500 Series switches. The **event gold** command was enhanced in addition to the Tcl keywords—**action-notify**, **testing-type**, **test-name**, **test-id**, **consecutive-failure**, **platform-action**, and **maxrun**—for improved reaction to GOLD test failures and conditions. The following sections were updated to describe the enhanced functionality of the **event gold** command: • EEM Event Registration Tcl Command Extensions, page 56 Optional arguments were added to the **event_register_gold** EEM Event Registration Tcl command extension to support additional event configuration options. • EEM Event Information Tcl Command Extension, page 117 Event types were added under the **event_reqinfo** EEM Event Information Tcl command extension to provide access to platform-wide and test-specific GOLD EEM Tcl policy information for a detected event.
Embedded Event Manager 2.4	12.4(20)T 12.2(33)SXI 12.2(33)SRE	EEM 2.4 is supported in Cisco IOS Release 12.4(20)T and later releases, and introduced several new features. The following sections were updated: • EEM Policy Tcl Command Extension Categories, page 3 • Modifying the Sample EEM Policies, page 20 • EEM Policy Tcl Command Extension Reference, page 55 The following commands were introduced by this feature: **attribute (EEM)**, **correlate**, **event manager detector rpc**, **event manager directory user repository**, **event manager update user policy**, **event manager scheduler clear**, **event manager update user policy**, **event owner**, **event rpc**, **event snmp-notification**, **show event manager detector**, **show event manager version**, **trigger (EEM)**.

Table 16 Feature Information for Writing Embedded Event Manager Policies Using Tcl (continued)

Feature Name	Releases	Feature Configuration Information
Embedded Event Manger 3.0	12.4(22)T 12.2(33)SRE	EEM 3.0 is supported in Cisco IOS Release 12.4(22)T and later releases, and introduced several new features. The following sections provide information about this feature: • EEM Policy Tcl Command Extension Categories, page 3 • Modifying the Sample EEM Policies, page 20 • EEM Policy Tcl Command Extension Reference, page 55 The following commands were introduced or modified by this feature: **action add**, **action append**, **action break**, **action comment**, **action context retrieve**, **action context save**, **action continue**, **action decrement**, **action divide**, **action else**, **action elseif**, **action end**, **action exit**, **action foreach**, **action gets**, **action if**, **action if goto**, **action increment**, **action info type interface-names**, **action info type snmp getid**, **action info type snmp inform**, **action info type snmp oid**, **action info type snmp trap**, **action info type snmp var**, **action multiply**, **action puts**, **action regexp**, **action set (EEM)**, **action string compare**, **action string equal**, **action string first**, **action string index**, **action string last**, **action string length**, **action string match**, **action string range**, **action string replace**, **action string tolower**, **action string toupper**, **action string trim**, **action string trimleft**, **action string trimright**, **action subtract**, **action while**, **event cli**, **event ipsla**, **event manager detector routing**, **event manager scheduler**, **event manager scheduler clear**, **event manager scheduler hold**, **event manager scheduler modify**, **event manager scheduler release**, **event nf**, **event routing**, **show event manager policy active**, **show event manager policy pending**, and **show event manager scheduler**.

Table 16 Feature Information for Writing Embedded Event Manager Policies Using Tcl (continued)

Feature Name	Releases	Feature Configuration Information
Embedded Event Manager 3.1	15.0(1)M	EEM 3.1 is supported in Cisco IOS Release 15.0(1)M and later releases, and introduced several new features. The following sections provide information about this feature: • EEM Policy Tcl Command Extension Categories, page 3 • Modifying the Sample EEM Policies, page 20 • EEM Policy Tcl Command Extension Reference, page 55 The following commands were introduced or modified by this feature: **action syslog**, **description**, **event manager applet**, **event manager policy**, **event snmp-notification**, **event snmp-object**, **show event manager policy registered**, and **show event manager policy available**.

1. This is a minor enhancement. Minor enhancements are not typically listed in Feature Navigator.

Embedded Resource Manager (ERM)

Embedded Resource Manager (ERM)

First Published: December 07, 2004
Last Updated: November 20, 2009

The Embedded Resource Manager (ERM) feature allows you to monitor internal system resource utilization for specific resources such as the buffer, memory, and CPU. ERM monitors resource utilization from the perspective of various subsystems within the Cisco IOS software such as resource owners (ROs) and resource users (RUs). ERM allows you to configure threshold values for system resources.

The ERM infrastructure is designed to allow for granular monitoring on a task basis within the Cisco IOS software. Network administrators can define thresholds to create notifications according to the real-time resource consumption. ERM goes beyond simply monitoring for total CPU utilization. Through the use of ERM, network administrators and operators can gain a better understanding of the device's operational characteristics, leading to better insight into system scalability and improved system availability.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the "Feature Information for Embedded Resource Manager" section on page 55.

Use Cisco Feature Navigator to find information about platform support and Cisco IOS and Catalyst OS software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Contents

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

Prerequisites for Embedded Resource Manager

You must be running Cisco IOS Release 12.4(6)T or a later release to use the Packet Memory Reclamation functionality.

Restrictions for Embedded Resource Manager

Additional instructions from a Cisco technical support representative may be required.

Information About Embedded Resource Manager

ERM promotes resource availability by providing the infrastructure to track resource usage.

To configure threshold values for resource manager entities, you should understand the following concepts:

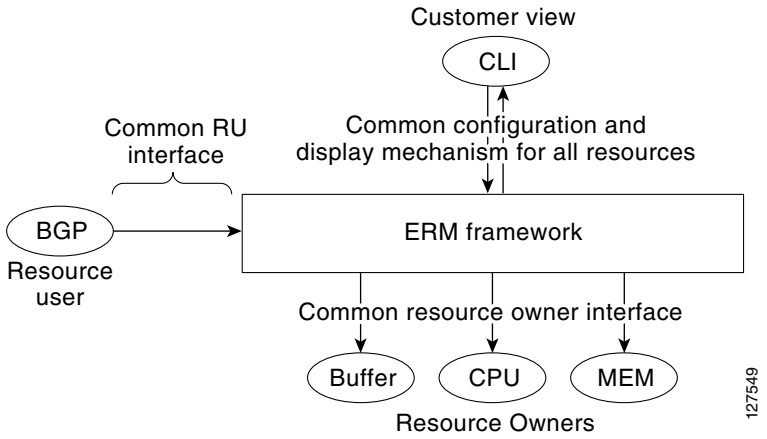
Benefits of the Embedded Resource Manager

The ERM framework tracks resource utilization and resource depletion by monitoring finite resources. Support for monitoring CPU, buffer, and memory utilization at a global or IOS-process level is available.

The ERM framework provides a mechanism to send notifications whenever the specified threshold values are exceeded by any resource user. This notification helps network administrators diagnose any CPU, buffer, and memory utilization issues.

The ERM architecture is illustrated in Figure 1.

Figure 1 ERM Architecture



ERM provides a framework for monitoring any finite resource within the Cisco IOS software and provides information that a user can analyze to better understand how network changes might impact system operation. ERM helps in addressing infrastructure problems such as reloads, memory allocation failure, and high CPU utilization by performing the following functions:

- Monitoring system resource usage.
- Setting the resource threshold at a granular level.
- Generating alerts when resource utilization reaches the specified level.
- Generating internal events using the Cisco IOS Embedded Event Manager feature.

Resource Accounting and Thresholds Tracking in ERM

ERM tracks the resource usage for each RU internally. An RU is a subsystem or process task within the Cisco IOS software; for example, the Open Shortest Path First (OSPF) hello process is a resource user. Threshold limits are used to notify network operators of specific conditions. The ERM infrastructure provides a means to notify the internal RU subsystem of threshold indications as well. The resource accounting is performed by individual ROs. ROs are part of the Cisco IOS software and are responsible for monitoring certain resources such as the memory, CPU, and buffer. When the utilization for each RU exceeds the threshold value you have set, the ROs send internal notifications to the RUs and to network administrators in the form of system logging (syslog) messages or Simple Network Management Protocol (SNMP) alerts.

You can set rising and falling values for critical, major, and minor levels of thresholds. When the resource utilization exceeds the rising threshold level, an Up notification is sent. When the resource utilization falls below the falling threshold level, a Down notification is sent.

ERM provides for three types of thresholds to be defined:

- The System Global Threshold is the point when the entire resource reaches a specified value. A notification is sent to all RUs once the threshold is exceeded.
- The User Local Threshold is the point when a specified RUs utilization exceeds the configured limit.
- The User Global Threshold is the point when the entire resource reaches a configured value. A notification is sent to the specified RU once the threshold is exceeded.

System Resources Monitored by the Embedded Resource Manager

ERM monitors CPU, buffer, and memory utilization at a global and task-based level. To avoid infrastructure issues and promote the availability of system resources, the resource owners described in the following sections are monitored:

- CPU Resource Owner, page 4
- Memory Resource Owner, page 5
- Buffer Resource Owner, page 7

CPU Resource Owner

The ERM feature uses the existing loadometer process to calculate the load information displayed by the **show processes cpu** command. This method generates a report of the extended load statistics and adds it to a circular buffer every five seconds. You can obtain a record of the load statistics for the past one minute through the CLI. This feature also provides an intelligent CPUHOG profiling mechanism that helps to reduce the time required to diagnose error conditions.

The functions described in the following sections help in load monitoring.

- Loadometer Process, page 4
- Scheduler, page 4
- Snapshot Management Using Event Trace, page 4
- Automatic CPUHOG Profiling, page 4

Loadometer Process

The loadometer process generates an extended load monitor report every five seconds. The loadometer function, which calculates process CPU usage percentage, is enhanced to generate the loadometer process reports.

Scheduler

The scheduler collects data when a process is executed, which enables the loadometer to generate reports. The scheduler collects data when the process is launched or when the process transfers control to the scheduler.

Snapshot Management Using Event Trace

Snapshot management manages the buffer in which snapshots of reports are stored. The snapshot management infrastructure stores, displays, and releases the snapshots.

Automatic CPUHOG Profiling

The timer Interrupt Service Routine (ISR) provides automatic CPUHOG profiling. The timer ISR begins profiling a process when it notices that the process has exceeded the configured value or a default of twice the maximum scheduling quantum (maximum time taken for the execution of a task).

On beginning the profiling, the timer ISR saves the interrupted program counter (pc) and return address (ra) in a preallocated buffer. This process provides information that can help the user analyze the CPUHOG.

The profiling continues until the CPUHOG is reported or the buffer is full. To analyze the computation of a long running process you must specify a process ID (PID) and a threshold to start the profiling. When this process takes up more than the specified time (in milliseconds), the profiling begins.

When the data belonging to a particular process exceeds the default size of the buffer, it is reported as a CPUHOG. The default size of the buffer is 1250 entries and can store up to five seconds of profiling data.

Memory Resource Owner

The Embedded Resource Manager feature enhances the memory manager in Cisco IOS devices. The enhancements are described in the following sections:

- Memory Usage History, page 5
- Memory Accounting, page 5
- Interface Wedging and Packet Memory Leaks, page 5
- Memory Resource Reclamation for Interfaces, page 6
- Memory Leak Reclamation, page 6
- I/O Memory, page 6

Memory Usage History

The Embedded Resource Manager feature helps in maintaining memory fragmentation information and thus reduces the need for maintenance of separate scripts for collecting such information.

Memory Accounting

ERM performs the accounting of information for memory by tracking the memory usage of individual RUs. When a process is created, a corresponding RU is also created, against which the usage of memory is recorded. The process of RU creation helps the user to migrate from a process-based accounting to a resource user-based accounting scheme for memory.

The memory RO maintains a global threshold and a per-RU memory usage threshold that can be configured through the ERM infrastructure. The memory RO also tracks the global free memory. When a particular RU's memory usage exceeds the global free memory, a notification is sent to the registered resource monitors (RMs). Similarly when a particular RU exceeds its threshold of memory usage, a notification is sent to that RU. These notifications are sent using the ERM infrastructure.

A memory RO has the intelligence to assign memory to a RU. When a memory RO receives an allocation request, the memory is assigned to the current RU. When a free request is received, the memory RO reduces the memory assigned to the RU.

Interface Wedging and Packet Memory Leaks

In certain situations, errors in the system accounting of incoming packets can occur, leading to a "memory leak" caused by the input queue. When there is a leak in an interface's input queue, gradually the queue reaches its maximum permitted value, causing the interface to become "wedged." A wedged interface may no longer process incoming packets. Packet memory leaks can cause interface input queue wedges.

The Packet Memory Reclamation functionality improves the infrastructure for preventing wedged interface input queues, and it provides a method for changing the defaults of that infrastructure. The Embedded Resource Manager provides the Packet Memory Reclamation functionality for "unwedging" interface input queues and configuring the system to detect and rectify packet leaks.

Note To use the Packet Memory Reclamation functionality, you must be running Cisco IOS Release 12.4(6)T or a later release. Additional troubleshooting (debugging) commands were introduced by this enhancement for use by technical support representatives in specific situations.

Memory Resource Reclamation for Interfaces

The Garbage Detection process works in conjunction with the Memory RO in achieving interface unwedging (for more details, see the *Memory Leak Dectector* feature guide that is part of the *Cisco IOS Configuration Fundamentals Configuration Guide*).

As part of the reclamation process, incoming packets that belong to a leaked input queue can be deallocated and reused. This feature provides a command (**critical rising**) that can be used to fine-tune memory resource reclamation.

Note Configuration of this feature will typically be needed only as part of a troubleshooting process with a Cisco Technical Support representative. Additional configuration tasks or special technical support commands may be required before this feature can be effectively used. Additional **memory debug leak internal service** commands are made available to Cisco Technical Support engineers for use in specific situations.

The deallocation procedure is triggered when a check is made to see if packets are using too much memory. Thresholds for the memory RO can be configured using a global policy of any level.

The purpose of configuring this memory policy is to find a balance between the utilization of the Memory Leak Detector (that can become resource intensive) and the need to detect packet memory leaks. Ideally, the system should perform deallocation only when it becomes absolutely necessary.

The **critical rising** command allows you to set a rising and falling threshold percentage for critical levels of I/O memory usage, and to specify an interval for those values. These values trigger the Memory Leak Detector process and, if needed, the deallocation procedure.

For example, if memory usage is more than that of the rising threshold of 75 percent of total I/O memory for more than 5 seconds, the “critical” notification is generated within the system and a callback is issued. As an action in the callback, a check is made to see if the packets are using too much memory. When the packets have used too much memory, the deallocation procedure begins. If the deallocation procedure does not bring memory utilization below the lower threshold value, the deallocation procedure is periodically reattempted. Once the memory usage falls below the configured threshold value, the periodic attempts to deallocate are stopped.

Memory Leak Reclamation

The Packet Memory Reclamation feature uses the ERM infrastructure to clean up and reclaim leaked Cisco IOS packet memory.

This feature uses the Memory Leak Detector process (sometimes referred to as the Garbage Detection or GD process) and the memory-manager RO functionality to reclaim packet memory.

I/O Memory

The I/O memory pool is one of the memory types in Cisco IOS software. The input queue buffers use memory from this pool for processing.

Buffer Resource Owner

The Embedded Resource Manager feature addresses the recurring problems of the Buffer Manager described in the following sections.

- Automatic Buffer Tuning, page 7
- Buffer Leak Detection, page 7
- Buffer Accounting, page 7
- Buffer Usage Thresholding, page 8

Automatic Buffer Tuning

The Embedded Resource Manager feature allows you to automatically tune the buffers using the **buffer tune automatic** command. The buffer RO tunes permanent memory in particle pools based on the usage of the buffer pool.

The buffer RO tracks the number of failures and the availability of memory in the buffer pool. When the number of failures increases above 1 percent of the buffer hits or when no memory is available in the buffer pool, the buffer RO performs an automatic tuning.

Note Ensure that there is sufficient free I/O memory or main memory using the first lines of the **show memory** command output before enabling automatic tuning of buffers.

Here are some keywords from the **buffer tune** command that can help you verify if you have sufficient I/O memory:

- **permanent**: take the number of total buffers in a pool and add 20 percent.
- **min-free**: set the **min-free** keyword to 20 to 30 percent of the permanent number of allocated buffers in the pool.
- **max-free**: set the **max-free** keyword to a value greater than the sum of permanent and minimum values.

However, when there is a traffic burst, the Cisco IOS device may not have enough time to create the new buffers and the number of failures may continue to increase.

The Embedded Resource Manager feature monitors the buffer pool every minute for tuning (that is, for number of hits, number of failures, and the number of counters created). When buffer tuning is enabled, the buffer RO automatically tunes the buffers when required.

Buffer Leak Detection

The Embedded Resource Manager feature allows Cisco IOS devices to detect and diagnose potential buffer leaks. All the buffers in a pool are linked so that they can be traced easily. The number of buffers allocated for incoming and outgoing packets in each buffer pool is tracked and can be displayed in the **show buffers leak** command output.

Buffer Accounting

The Embedded Resource Manager feature consists of mechanisms to account for the usage of buffers. All buffers are owned by the pool manager process (buffer RU). When a RU requests a buffer, the allocated buffer is allotted to that RU. When the RU returns the buffer, it is deducted from the RU's account. The packet type from the output of the **show buffers usage** command indicates the RU to which the packet belongs.

Buffer Usage Thresholding

The Embedded Resource Manager feature provides a facility to manage high buffer utilization. The buffer manager RO registers as a RU with the memory RO. The buffer manager RU is set before a memory allocation is made for creating new buffers. The buffer manager also registers as an RO. When a buffer is allocated, the current RU (if any) is charged with the memory allocation. The buffer manager RO registers for the notifications from the memory manager for the processor and I/O memory pool. If the I/O memory pool is falling short of memory, the buffer manager tries to free the lists of all the buffer pools. If your Cisco IOS device does not support I/O memory, then it registers for notifications from the processor memory.

Cisco IOS software maintains a threshold per buffer pool. When a particular pool exceeds the specified threshold, ERM sends a notification to all the RUs in that pool, so that the RUs can take corrective measures. Thresholds are configured for public buffer pools only.

Global notification is set for every pool in the system; that is, one notification for all pools in the public pool and one notification for each pool in the private pool. Threshold notifications are sent to only those RUs that have registered with the ROs for getting notifications. A list of RUs that have registered with the RO is maintained by the RO. When the threshold of a particular RU is exceeded, then that RU is notified and marked notified. When the buffers are recovered, the notified RUs are moved back to the original list.

For example, an Ethernet driver RU is allocated buffers from some particular private pool. Another RU, Inter Processor Communication (IPC), is added to the list. In this case, when the pool runs low on buffers, the IPC RU gets a notification and it can take corrective measures.

You can configure threshold values as percentages of the total buffers available in the public pool. Total buffer is the sum of maximum allowed buffers and the permanent pools in the public buffer pool. If these values change due to buffer tuning, then the threshold values also change. For example, if the configuration requires that a notification be sent when the IPC RU is holding more than 40 percent of Ethernet buffers and the sum of permanent and maximum allowed for Ethernet buffers is 150 percent, then the Ethernet pool is notified when the IPC RU is holding 60 percent.

Resource Policy Templates

Resource owner policy is a template used by the ROs to associate a RU with a set of thresholds that are configured through the CLI. This template can be used to specify system global, user local, and per user global thresholds. A particular resource group or RU can have only one policy associated with it. The policy template for ROs is maintained by the ERM framework.

When a policy template is associated with a user type and its instance (RUs), the thresholds configured in that policy are applied based on the RU to RO relationship. This method ignores any RO configuration that may not be applicable to the RU.

How to Configure Embedded Resource Manager

This section contains the following procedures.

- Managing Resource Utilization by Defining Resource Policy, page 9 (required)
- Setting Expected Operating Ranges for Buffer Resources, page 10 (required)
- Setting Expected Operating Ranges for CPU Resources, page 12 (required)
- Setting Expected Operating Ranges for Memory Resources, page 17 (required)

Managing Resource Utilization by Defining Resource Policy

Perform this task to configure a resource policy for ERM.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **resource policy**
4. **policy** *policy-name* [**global** | **type** *resource-user-type*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**resource policy** **Example:** `Router(config)# resource policy`	Enters ERM configuration mode.
Step 4	**policy** *policy-name* [**global** | **type** *resource-user-type*] **Example:** `Router(config-erm)# policy policy1 type iosprocess`	Enters ERM policy configuration mode to configure a resource policy. • The *policy-name* argument identifies the name of the resource policy. • The **global** keyword is used when you are configuring a system global policy. • The **type** keyword indicates that you are configuring either a user local or per user global policy. The *resource-user-type* argument identifies the name of the resource user type you want to attach the policy to.

Setting Expected Operating Ranges for Buffer Resources

Perform this task to configure threshold values for buffer RO.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **resource policy**
4. **policy** *policy-name* [**global** | **type** *resource-user-type*]
5. **system**
   or
   slot *slot-number*
6. **buffer public**
7. **critical rising** *rising-threshold-value* [**interval** *interval-value*] [**falling** *falling-threshold-value* [**interval** *interval-value*]] [**global**]
   or
   major rising *rising-threshold-value* [**interval** *interval-value*] [**falling** *falling-threshold-value* [**interval** *interval-value*]] [**global**]
   or
   minor rising *rising-threshold-value* [**interval** *interval-value*] [**falling** *falling-threshold-value* [**interval** *interval-value*]] [**global**]
8. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**resource policy** **Example:** `Router(config)# resource policy`	Enters ERM configuration mode.
Step 4	**policy** *policy-name* [**global** | **type** *resource-user-type*] **Example:** `Router(config-erm)# policy policy1 type iosprocess`	Configures a resource policy and enters ERM policy configuration mode. • The *policy-name* argument identifies the name of the resource policy. • The **global** keyword is used when you are configuring a system global policy. • The **type** keyword indicates that you are configuring either a user local or per user global policy. The *resource-user-type* argument identifies the name of the resource user type you want to attach the policy to.

Step 5	**system**
	
	or
	
	slot *slot-number*
	
	Example:
	`Router(config-erm-policy)# system`
	
	or
	
	Example:
	`Router(config-erm-policy)# slot 1`
		Enters policy node configuration mode with the **system** command.
		
		Enters ERM slot configuration mode with the **slot** *slot-number* command. This command is available only in distributed platforms like the Route Switch Processor (RSP).

Step 6	**buffer public** **Example:** `Router(config-policy-node)# buffer public`	Enters buffer owner configuration mode. Allows you to set the rising and falling values for the critical, major, and minor thresholds.

	Command or Action	Purpose

Step 7	**critical rising** *rising-threshold-value* [**interval** *interval-value*] [**falling** *falling-threshold-value* [**interval** *interval-value*]] [**global**]
	or
	major rising *rising-threshold-value* [**interval** *interval-value*] [**falling** *falling-threshold-value* [**interval** *interval-value*]] [**global**]
	or
	minor rising *rising-threshold-value* [**interval** *interval-value*] [**falling** *falling-threshold-value* [**interval** *interval-value*]] [**global**]
	
	Example:
	`Router(config-owner-buffer)# critical rising 40 falling 20 interval 10 global`
	or
	
	Example:
	`Router(config-owner-buffer)# major rising 30 falling 15 interval 10 global`
	or
	
	Example:
	`Router(config-owner-buffer)# minor rising 20 falling 10 interval 10 global`
		Allows you to set the rising and falling threshold values for critical, major, and minor levels of buffer usage count for the public buffer pools.
		
		Note If you had configured a global policy in Step 4, you do not need to give the **global** keyword while setting the threshold values in Step 7. However, if you have configured a user local or per user global policy (by not specifying the **global** keyword) in Step 4, enter the **global** keyword in Step 7 if you want to configure a per user global threshold.

Step 8	**exit** **Example:** `Router(config-owner-buffer)# exit`	Exits buffer owner configuration mode.

Setting Expected Operating Ranges for CPU Resources

Perform this task to configure threshold values for the CPU RO.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **resource policy**
4. **policy** *policy-name* [**global** | **type** *resource-user-type*]
5. **system**
   or
   slot *slot-number*
6. **cpu interrupt**

7. **critical rising** *rising-threshold-value* [**interval** *interval-value*] [**falling** *falling-threshold-value* [**interval** *interval-value*]] **global**
   or
   major rising *rising-threshold-value* [**interval** *interval-value*] [**falling** *falling-threshold-value* [**interval** *interval-value*]] **global**
   or
   minor rising *rising-threshold-value* [**interval** *interval-value*] [**falling** *falling-threshold-value* [**interval** *interval-value*]] **global**
8. **exit**
9. **cpu process**
10. **critical rising** *rising-threshold-value* [**interval** *interval-value*] [**falling** *falling-threshold-value* [**interval** *interval-value*]] [**global**]
    or
    major rising *rising-threshold-value* [**interval** *interval-value*] [**falling** *falling-threshold-value* [**interval** *interval-value*]] [**global**]
    or
    minor rising *rising-threshold-value* [**interval** *interval-value*] [**falling** *falling-threshold-value* [**interval** *interval-value*]] [**global**]
11. **exit**
12. **cpu total**
13. **critical rising** *rising-threshold-value* [**interval** *interval-value*] [**falling** *falling-threshold-value* [**interval** *interval-value*]] **global**
    or
    major rising *rising-threshold-value* [**interval** *interval-value*] [**falling** *falling-threshold-value* [**interval** *interval-value*]] **global**
    or
    minor rising *rising-threshold-value* [**interval** *interval-value*] [**falling** *falling-threshold-value* [**interval** *interval-value*]] **global**
14. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**resource policy** **Example:** `Router(config)# resource policy`	Enters ERM configuration mode.

	Command or Action	Purpose
Step 4	**policy** *policy-name* [**global** | **type** *resource-user-type*] **Example:** `Router(config-erm)# policy policy1 type iosprocess`	Configures a resource policy and enters ERM policy configuration mode. • The *policy-name* argument identifies the name of the resource policy. • The **global** keyword is used when you are configuring a system global policy. • The **type** keyword indicates that you are configuring either a user local or per user global policy. The *resource-user-type* argument identifies the name of the resource user type you want to attach the policy to.

Step 5	**system**
	
	or
	
	slot *slot-number*
	
	Example:
	`Router(config-erm-policy)# system`
	
	or
	
	Example:
	`Router(config-erm-policy)# slot 1`
		Enters policy node configuration mode with the **system** command.
		
		Enters ERM slot configuration mode with the **slot** *slot-number* command. This command is available only in distributed platforms like the RSP.

Step 6	**cpu interrupt** **Example:** `Router(config-policy-node)# cpu interrupt`	(Optional) Enters CPU owner configuration mode. Allows you to set the rising and falling values for the critical, major, and minor thresholds.

	Command or Action	Purpose

Step 7	**critical rising** *rising-threshold-value* [**interval** *interval-value*] [**falling** *falling-threshold-value* [**interval** *interval-value*]] **global**
	or
	major rising *rising-threshold-value* [**interval** *interval-value*] [**falling** *falling-threshold-value* [**interval** *interval-value*]] **global**
	or
	minor rising *rising-threshold-value* [**interval** *interval-value*] [**falling** *falling-threshold-value* [**interval** *interval-value*]] **global**
	
	Example:
	`Router(config-owner-cpu)# critical rising 40 falling 20 interval 10 global`
	or
	
	Example:
	`Router(config-owner-cpu)# major rising 30 falling 15 interval 10 global`
	or
	
	Example:
	`Router(config-owner-cpu)# minor rising 20 falling 10 interval 10 global`
		Allows you to set the rising and falling threshold values for critical, major, and minor levels of percentages of CPU interrupt utilization.
		
		Note If you had configured a global policy in Step 4, you do not need to give the **global** keyword while setting the threshold values in Step 7. However, if you have configured a user local or per user global policy (by not specifying the **global** keyword) in Step 4, enter the **global** keyword in Step 7 if you want to configure a per user global threshold.
		
		For interrupt CPU utilization, you can configure either global thresholds or per user global thresholds. Hence, you must enter the **global** keyword either in Step 4 or in Step 7.

Step 8	**exit** **Example:** `Router(config-owner-cpu)# exit`	Exits the CPU owner configuration mode.
Step 9	**cpu process** **Example:** `Router(config-policy-node)# cpu process`	(Optional) Enters CPU owner configuration mode. Allows you to set the rising and falling values for the critical, major, and minor thresholds.

	Command or Action	Purpose

Step 10	**critical rising** *rising-threshold-value* [**interval** *interval-value*] [**falling** *falling-threshold-value* [**interval** *interval-value*]] [**global**]
	or
	major rising *rising-threshold-value* [**interval** *interval-value*] [**falling** *falling-threshold-value* [**interval** *interval-value*]] [**global**]
	or
	minor rising *rising-threshold-value* [**interval** *interval-value*] [**falling** *falling-threshold-value* [**interval** *interval-value*]] [**global**]
	Example:
	`Router(config-owner-cpu)# critical rising 40 falling 20 interval 10 global`
	or
	Example:
	`Router(config-owner-cpu)# major rising 30 falling 15 interval 10 global`
	or
	Example:
	`Router(config-owner-cpu)# minor rising 20 falling 10 interval 10 global`
		Allows you to set the rising and falling threshold values for critical, major, and minor levels of percentages of process CPU utilization.
		Note If you had configured a global policy in Step 4, you do not need to give the **global** keyword while setting the threshold values in Step 10. However, if you have configured a user local or per user global policy (by not specifying the **global** keyword) in Step 4, enter the **global** keyword in Step 10 if you want to configure a per user global threshold.
		For process CPU utilization, you can configure global thresholds, per user global thresholds or user local thresholds.

Step 11	**exit** **Example:** `Router(config-owner-cpu)# exit`	Exits the CPU owner configuration mode.
Step 12	**cpu total** **Example:** `Router(config-policy-node)# cpu total`	(Optional) Enters CPU owner configuration mode. Allows you to set the rising and falling values for the critical, major, and minor thresholds.

	Command or Action	Purpose

Step 13	**critical rising** *rising-threshold-value* [**interval** *interval-value*] [**falling** *falling-threshold-value* [**interval** *interval-value*]] **global**
	or
	major rising *rising-threshold-value* [**interval** *interval-value*] [**falling** *falling-threshold-value* [**interval** *interval-value*]] **global**
	or
	minor rising *rising-threshold-value* [**interval** *interval-value*] [**falling** *falling-threshold-value* [**interval** *interval-value*]] **global**
	
	Example:
	`Router(config-owner-cpu)# critical rising 40 falling 20 interval 10 global`
	or
	
	Example:
	`Router(config-owner-cpu)# major rising 30 falling 15 interval 10 global`
	or
	
	Example:
	`Router(config-owner-cpu)# minor rising 20 falling 10 interval 10 global`
		Allows you to set the rising and falling threshold values for critical, major, and minor levels of percentages of total CPU utilization.
		
		Note If you had configured a global policy in Step 4, you do not need to give the **global** keyword while setting the threshold values in Step 13. However, if you have configured a user local or per user global policy (by not specifying the **global** keyword) in Step 4, enter the **global** keyword in Step 13 if you want to configure a per user global threshold.
		
		For total CPU utilization, you can configure either global thresholds or per user global thresholds. Hence, you must enter the **global** keyword either in Step 4 or in Step 13.

Step 14	**exit** **Example:** `Router(config-owner-cpu)# exit`	Exits CPU owner configuration mode.

Setting Expected Operating Ranges for Memory Resources

Perform this task to configure threshold values for the memory RO.

Note When the Packet Memory Reclamation functionality is enabled, and the violation of the configured threshold value for the memory RO occurs, the system verifies whether the memory is hogged by the buffers. If 70 percent of the memory is used by the buffers, the system activates the Memory Leak Detector process (sometimes referred to as the "Garbage Detection" or "GD" process) to clean up the memory. (For more details, see the Memory Leak Dectector feature guide that is part of the *Cisco IOS Configuration Fundamentals Configuration Guide*).

SUMMARY STEPS

1. **enable**
2. **configure terminal**

3. **resource policy**
4. **policy** *policy-name* [**global** | **type** *resource-user-type*]
5. **system**
   or
   slot *slot-number*
6. **memory io**
7. **critical rising** *rising-threshold-value* [**interval** *interval-value*] [**falling** *falling-threshold-value* [**interval** *interval-value*]] [**global**]
   or
   major rising *rising-threshold-value* [**interval** *interval-value*] [**falling** *falling-threshold-value* [**interval** *interval-value*]] [**global**]
   or
   minor rising *rising-threshold-value* [**interval** *interval-value*] [**falling** *falling-threshold-value* [**interval** *interval-value*]] [**global**]
8. **exit**
9. **memory processor**
10. **critical rising** *rising-threshold-value* [**interval** *interval-value*] [**falling** *falling-threshold-value* [**interval** *interval-value*]] [**global**]
    or
    major rising *rising-threshold-value* [**interval** *interval-value*] [**falling** *falling-threshold-value* [**interval** *interval-value*]] [**global**]
    or
    minor rising *rising-threshold-value* [**interval** *interval-value*] [**falling** *falling-threshold-value* [**interval** *interval-value*]] [**global**]
11. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**resource policy** **Example:** `Router(config)# resource policy`	Enters ERM configuration mode.

	Command or Action	Purpose
Step 4	**policy** *policy-name* [**global** | **type** *resource-user-type*] **Example:** `Router(config-erm)# policy policy1 type iosprocess`	Configures a resource policy and enters ERM policy configuration mode. • The *policy-name* argument identifies the name of the resource policy. • The **global** keyword is used when you are configuring a system global policy. • The **type** keyword indicates that you are configuring either a user local or per user global policy. The *resource-user-type* argument identifies the name of the resource user type you want to attach the policy to.

Step 5	**system**
	or
	slot *slot-number*
	
	Example:
	`Router(config-erm-policy)# system`
	or
	
	Example:
	`Router(config-erm-policy)# slot 1`
		Enters policy node configuration mode with the **system** command.
		Enters ERM slot configuration mode with the **slot** *slot-number* command. This command is available only in distributed platforms like the RSP.

Step 6	**memory io** **Example:** `Router(config-policy-node)# memory io`	(Optional) Enters memory owner configuration mode. Allows you to set the rising and falling values for the critical, major, and minor thresholds.

	Command or Action	Purpose

Step 7	**critical rising** *rising-threshold-value* [**interval** *interval-value*] [**falling** *falling-threshold-value* [**interval** *interval-value*]] [**global**]
	or
	major rising *rising-threshold-value* [**interval** *interval-value*] [**falling** *falling-threshold-value* [**interval** *interval-value*]] [**global**]
	or
	minor rising *rising-threshold-value* [**interval** *interval-value*] [**falling** *falling-threshold-value* [**interval** *interval-value*]] [**global**]
	Example:
	`Router(config-owner-memory)# critical rising 40 falling 20 interval 10 global`
	or
	Example:
	`Router(config-owner-memory)# major rising 30 falling 15 interval 10 global`
	or
	Example:
	`Router(config-owner-memory)# minor rising 20 falling 10 interval 10 global`
		Allows you to set the rising and falling threshold values for critical, major, and minor levels of percentages of I/O memory usage.
		Note If you had configured a global policy in Step 4, you do not need to give the **global** keyword while setting the threshold values in Step 7. However, if you have configured a user local or per user global policy (by not specifying the **global** keyword) in Step 4, enter the **global** keyword in Step 7 if you want to configure a per user global threshold.

Step 8	**exit** **Example:** `Router(config-owner-memory)# exit`	Exits memory owner configuration mode.
Step 9	**memory processor** **Example:** `Router(config-policy-node)# memory processor`	(Optional) Enters memory owner configuration mode. Allows you to set the rising and falling values for the critical, major, and minor thresholds.

	Command or Action	Purpose

Step 10	**critical rising** *rising-threshold-value* [**interval** *interval-value*] [**falling** *falling-threshold-value* [**interval** *interval-value*]] [**global**]
	or
	major rising *rising-threshold-value* [**interval** *interval-value*] [**falling** *falling-threshold-value* [**interval** *interval-value*]] [**global**]
	or
	minor rising *rising-threshold-value* [**interval** *interval-value*] [**falling** *falling-threshold-value* [**interval** *interval-value*]] [**global**]
	
	Example:
	`Router(config-owner-memory)# critical rising 40 falling 20 interval 10 global`
	or
	
	Example:
	`Router(config-owner-memory)# major rising 30 falling 15 interval 10 global`
	or
	
	Example:
	`Router(config-owner-memory)# minor rising 20 falling 10 interval 10 global`
		Allows you to set the rising and falling threshold values for critical, major, and minor levels of percentages of processor memory usage.
		
		Note If you had configured a global policy in Step 4, you do not need to give the **global** keyword while setting the threshold values in Step 10. However, if you have configured a user local or per user global policy (by not specifying the **global** keyword) in Step 4, enter the global keyword in Step 10 if you want to configure a per user global threshold.

Step 11	**exit** **Example:** `Router(config-owner-memory)# exit`	Exits memory owner configuration mode.

Enabling Automatic Tuning of Buffers

Perform this task to enable automatic tuning of buffers.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **buffer tune automatic**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**buffer tune automatic** **Example:** `Router(config)# buffer tune automatic`	Enables automatic tuning of buffers.

Managing Memory Usage History

Perform this task to change the number of hours for which the memory log is maintained.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **memory statistics history table** *number-of-hours*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**memory statistics history table** *number-of-hours* **Example:** `Router(config)# memory statistics history table 48`	Changes the time (number of hours) for which the memory log is maintained.

Configuring a CPU Process to Be Included in the Extended Load Monitor Report

Perform this task to configure a process (or processes) to be included in the extended load monitor report.

SUMMARY STEPS

1. **enable**
2. **monitor processes cpu extended** *process-id-list*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**monitor processes cpu extended** *process-id-list* **Example:** `Router# monitor processes cpu extended 1`	Enables the specified process or processes to be monitored for the extended CPU load. You can specify a maximum of eight processes to be monitored.

Managing Extended CPU Load Monitoring

Perform this task to change the history size in the collection report for extended CPU load.

Restrictions

You cannot disable this feature completely. If the command is not configured, the default behavior is to collect a one-minute history. The one-minute history is equivalent to collecting history for a history size 12.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **process cpu extended history** *history-size*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**process cpu extended history** *history-size* **Example:** `Router(config)# process cpu extended history 24`	Enables you to change the history size of the extended collection report. If the command is not configured, the default behavior is to collect a one-minute history, which is equivalent to collecting history for history size 12.

Managing Automatic CPUHOG Profiling

Perform this task to enable automatic profiling of CPUHOGs by the CPU Resource Owner. The CPU Resource Owner predicts when a process could hog CPU and begins profiling that process at the same time. This function is enabled by default.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **processes cpu autoprofile hog**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**processes cpu autoprofile hog** **Example:** `Router(config)# processes cpu autoprofile hog`	Enables automatic profiling of CPUHOG processes. This function is enabled by default.

Applying a Policy to Resource Users

Perform this task to apply a policy or policy template to RUs or resource groups.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **resource policy**
4. **policy** *policy-name* [**global** | **type** *resource-user-type*]
5. **exit**
6. **user** {*resource-instance-name resource-user-type resource-policy-name* | **global** *global-policy-name* | **group** *resource-group-name* **type** *resource-user-type*}
7. **instance** *instance-name*
8. **policy** *policy-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**resource policy** **Example:** `Router(config)# resource policy`	Enters ERM configuration mode.
Step 4	**policy** *policy-name* [**global** | **type** *resource-user-type*] **Example:** `Router(config-erm)# policy policy1 type iosprocess`	Configures a resource policy and enters ERM policy configuration mode. • The *policy-name* argument identifies the name of the resource policy. • The **global** keyword is used when you are configuring a system global policy. • The **type** keyword indicates that you are configuring either a user local or per user global policy. The *resource-user-type* argument identifies the name of the resource user type you want to attach the policy to.
Step 5	**exit** **Example:** `Router(config-erm)# exit`	Exits ERM policy configuration mode.

	Command or Action	Purpose
Step 6	**user** {*resource-instance-name resource-user-type resource-policy-name* | **global** *global-policy-name* | **group** *resource-group-name* **type** *resource-user-type*} **Example:** `Router(config-erm)# user group lowPrioUsers type iosprocess`	Applies a policy system wide (global thresholding), a group of users (group thresholding), or a particular user. **Note** When you apply a group policy to a group of RUs by giving the **group** keyword in this command, the Cisco IOS router enters the resource group configuration mode. Go to Step 7 if you want to add RUs to the resource group. Got to Step 8 if you want to apply a policy to the resource group. • The *resource-instance-name* argument identifies the name of the RU to which you are applying a policy. • The *resource-user-type-name* argument identifies the type of RU. • The *resource-policy-name* argument identifies the name resource policy you are applying to the individual RU. • The *global-policy-name* argument identifies the name of the global policy you are trying to apply. • The *resource-group-name* argument identifies the name of the resource group.
Step 7	**instance** *instance-name* **Example:** `Router(config-res-group)# instance http`	Adds an RU to a resource group. The *instance-name* argument specifies the RU or instance name. **Note** All the RUs added by this command will be grouped together under the resource group and the same thresholding policy will be applied to all the RUs. For example, if you have created a resource group **lowPrioUsers** in Step 6, then all the RUs you add in Step 7 will be part of the resource group **lowPrioUsers** and the same policy is applied to all the RUs.
Step 8	**policy** *policy-name* **Example:** `Router(config-res-group)# policy group-policy1`	Specifies the policy you want to apply to the resource group you created in Step 6. The *policy-name* argument specifies the name of the group policy. This command helps you to set the same threshold policy to a group of RUs grouped under a resource group. For example, if you have some low-priority tasks or RUs like **http** and **snmp** and you want to set a threshold not on these individual RUs, but as a group; then add these RUs to the **lowPrioUsers** group using Step 7 and then apply a threshold policy using Step 8. In this case, if you have set a minor rising threshold of 10 percent (this 10 percent threshold is applied to both **http** and **snmp** in the **lowPrioUsers** group), then a notification is sent to **lowPrioUsers** resource group when the accumulated usage exceeds the 10 percent mark. That is, if http uses 4 percent and snmp uses 7 percent, a notification will be sent to all the RUs in the **lowPrioUsers** resource group.

Setting a Critical Rising Threshold for Global I/O Memory

Perform this task to specify a critical rising threshold value for the global I/O memory pool. If global I/O memory resource consumption meets or exceeds this value, the Memory Leak Detector process will be automatically triggered. This configuration is only needed if you are experiencing a problem and you want to change (fine tune) how often the automatic process occurs (for example, set the threshold lower so that deallocation check occurs more frequently).

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **resource policy**
4. **policy** *policy-name* [**global** | **type** *resource-user-type*]
5. **system**
   or
   slot *slot-number*
6. **memory io**
7. **critical rising** *rising-threshold-value* [**interval** *interval-value*] [**falling** *falling-threshold-value* [**interval** *interval-value*]] [**global**]
8. **exit**

DETAILED STEP

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**resource policy** **Example:** `Router(config)# resource policy`	Enters ERM configuration mode.

	Command or Action	Purpose
Step 4	**policy** *policy-name* [**global** | **type** *resource-user-type*] **Example:** `Router(config-erm)# policy policy1 type iosprocess`	Configures a resource policy and enters ERM policy configuration mode. • The *policy-name* argument identifies the name of the resource policy. • The **global** keyword is used when you are configuring a system global policy. • The **type** keyword indicates that you are configuring either a user local or per-user global policy. The *resource-user-type* argument identifies the name of the resource user type you want to attach the policy to.

Step 5	**system**
	or
	slot *slot-number*
	
	Example:
	`Router(config-erm-policy)# system`
	or
	
	Example:
	`Router(config-erm-policy)# slot 1`
		Enters policy node configuration mode with the **system** command.
		Enters ERM slot configuration mode with the **slot** *slot-number* command. This command is available only in distributed platforms like RSP.

Step 6	**memory io** **Example:** `Router(config-policy-node)# memory io`	(Optional) Enters memory owner configuration mode. • Allows you to set the rising and falling values for the critical, major, and minor thresholds.

	Command or Action	Purpose
Step 7	**critical rising** *rising-threshold-value* [**interval** *interval-value*] [**falling** *falling-threshold-value* [**interval** *interval-value*]] [**global**] **Example:** `Router(config-owner-memory)# critical rising 75 falling 65 interval 10 global`	Allows you to set the rising and falling threshold values for critical levels as percentages of the I/O memory pool, and set the interval of time that must pass before these values are registered. • If the amount of memory held by the resource user exceeds the rising threshold value, a rising threshold notification is generated. • If the falling threshold notification is generated before the interval has passed, then the rising notification is not sent. • The interval following the rising threshold signifies this time period in seconds. • If the amount of memory held by the resource user falls below the falling threshold, the falling threshold notification is sent. • The optional **global** keyword indicates that the threshold is being set on the global memory consumption, not on the memory used by the particular resource user in which the configuration is being applied. **Note** If you had configured a global policy in Step 4, you do not need to give the **global** keyword while setting the threshold values in Step 7. However, if you have configured a user local or per-user global policy (by not specifying the **global** keyword) in Step 4, enter the **global** keyword in Step 7 if you want to configure a per user global threshold.
Step 8	**exit** **Example:** `Router(config-owner-memory)# exit`	Exits memory owner configuration mode.

Verifying ERM Operations

To verify the various ERM operations, perform the following steps.

SUMMARY STEPS

1. **show buffers leak** [**resource user**]
2. **show buffers tune**
3. **show buffers usage** [**pool** *pool-name*]
4. **show memory** [**processor** | **io**] **fragment** [**detail**]
5. **show memory statistics history table**
6. **show monitor event-trace cpu-report** {**brief** {**all** [**detail**] | **back** *time* | **clock** *time* | **from-boot** [*seconds* | **detail**] | **latest** [**detail**]} | **handle** *handle-number*}

7. **show processes cpu autoprofile hog**
8. **show processes cpu extended** [**history**]
9. **show resource all** [**brief | detailed**]
10. **show resource database**
11. **show resource owner** {*resource-owner-name* | **all**} **user** {*resource-user-type-name* | **all**} [**brief | detailed | triggers**]
12. **show resource relationship user** *resource-user-type*
13. **show resource user** {**all** | *resource-user-type*} [**brief | detailed**]

DETAILED STEPS

Step 1 **show buffers leak** [**resource user**]

Use this command without the optional keywords to display the details of all the buffers that are older than one minute in the system, for example:

```
Router# show buffers leak

Header   DataArea  Pool  Size  Link  Enc    Flags      Input     Output  User

6488F464  E000084 Small    74     0    0       10       None       None EEM ED Sy
6488FB5C  E000304 Small    74     0    0       10       None       None EEM ED Sy
648905D0  E0006C4 Small    61     0    0        0       None       None EEM ED Sy
648913C0  E000BC4 Small    74     0    0       10       None       None EEM ED Sy
6489173C  E000D04 Small    74     0    0       10       None       None EEM ED Sy
648921B0  E0010C4 Small    60     0    0        0       None       None Init
6489252C  E001204 Small   103     0    0       10       None       None EEM ED Sy
64892C24  E001484 Small    74     0    0       10       None       None EEM ED Sy
64892FA0  E0015C4 Small    74     0    0       10       None       None EEM ED Sy
64893A14  E001984 Small    74     0    0       10       None       None EEM ED Sy
64893D90  E001AC4 Small    61     0    0        0       None       None EEM ED Sy
64894804  E001E84 Small    61     0    0        0       None       None EEM ED Sy
6517CB64  E32F944 Small    74     0    0       10       None       None EEM ED Sy
6517D25C  E176D44 Small    74     0    0       10       None       None EEM ED Sy
6517D5D8  E176E84 Small    74     0    0       10       None       None EEM ED Sy
6517D954  E209A84 Small    74     0    0       10       None       None EEM ED Sy
6517E744  E209D04 Small    61     0    0        0       None       None EEM ED Sy
6517EE3C  E29CBC4 Small    61     0    0        0       None       None EEM ED Sy
65180324  E177844 Small    74     0    0       10       None       None EEM ED Sy
65180D98  E177C04 Small    61     0    0        0       None       None EEM ED Sy
65E1F3A0  E4431A4 Small   102     0    0        0       None       None EEM ED Sy
64895278  E002644 Middl   191     0    0       10       None       None EEM ED Sy
64895CEC  E003004 Middl   173     0    0       10       None       None EEM ED Sy
64896068  E003344 Middl   176     0    0       10       None       None EEM ED Sy
648963E4  E003684 Middl   191     0    0       10       None       None EEM ED Sy
64896E58  E004044 Middl   109     0    0       10       None       None EEM ED Sy
64897C48  E004D44 Middl   194     0    0       10       None       None EEM ED Sy
65181F04  E330844 Middl   173     0    0       10       None       None EEM ED Sy
65183070  E3C3644 Middl   105     0    0       10       None       None EEM ED Sy
65DF9558  E4746E4 Middl   107     0    0        0       None       None EEM ED Sy
65DFA6C4  E475724 Middl   116     0    0        0       None       None EEM ED Sy
65DFADBC  E475DA4 Middl   115     0    0        0       None       None EEM ED Sy
65DFC620  E477464 Middl   110     0    0        0       None       None EEM ED Sy
64C64AE0        0 FS He     0     0    3        0       None       None Init
64C64E5C        0 FS He     0     0    3        0       None       None Init
64C651D8        0 FS He     0     0    3        0       None       None Init
64C65554        0 FS He     0     0    0        0       None       None Init
64C658D0        0 FS He     0     0    0        0       None       None Init
```

```
64C65C4C        0 FS He     0    0    0        0       None       None Init
64C65FC8        0 FS He     0    0    0        0       None       None Init
64C66344        0 FS He     0    0    0        0       None       None Init
64D6164C        0 FS He     0    0    0        0       None       None Init
64EB9D10        0 FS He     0    0    0        0       None       None Init
6523EE14        0 FS He     0    0    0        0       None       None Init
65413648        0 FS He     0    0    0        0       None       None Init
```

Use this command with the optional keywords to display the details of the buffers of a specified RU that are older than one minute in the system, for example:

```
Router# show buffers leak resource user

Resource User:  EEM ED Syslog count:       32
Resource User:           Init count:        2
Resource User:         *Dead* count:        2
Resource User: IPC Seat Manag count:       11
Resource User:      XDR mcast count:        2
```

Step 2 **show buffers tune**

Use this command to display the details of automatic tuning of buffers, for example:

```
Router# show buffers tune

Tuning happened for the pool Small

Tuning happened at 20:47:25
Oldvalues
permanent:50  minfree:20  maxfree:150
Newvalues
permanet:61  minfree:15  maxfree:76

Tuning happened for the pool Middle
Tuning happened at 20:47:25
Oldvalues
permanent:25  minfree:10  maxfree:150
Newvalues
permanet:36  minfree:9  maxfree:45
```

Step 3 **show buffers usage** [**pool** *pool-name*]

Use this command without the optional keyword and argument to display the details of the buffer usage pattern in a specified buffer pool, for example:

```
Router# show buffers usage

Statistics for the Small pool
Caller pc    : 0x626BA9E0 count:       20
Resource User: EEM ED Sys count:       20
Caller pc    : 0x60C71F8C count:        1
Resource User:       Init count:        1
Number of Buffers used by packets generated by system:   62
Number of Buffers used by incoming packets:               0

Statistics for the Middle pool
Caller pc    : 0x626BA9E0 count:       12
Resource User: EEM ED Sys count:       12
Number of Buffers used by packets generated by system:   41
Number of Buffers used by incoming packets:               0

Statistics for the Big pool
Number of Buffers used by packets generated by system:   50
Number of Buffers used by incoming packets:               0
```

```
Statistics for the VeryBig pool
Number of Buffers used by packets generated by system:   10
Number of Buffers used by incoming packets:               0

Statistics for the Large pool
Number of Buffers used by packets generated by system:    0
Number of Buffers used by incoming packets:               0

Statistics for the Huge pool
Number of Buffers used by packets generated by system:    0
Number of Buffers used by incoming packets:               0

Statistics for the IPC pool
Number of Buffers used by packets generated by system:    2
Number of Buffers used by incoming packets:               0

Statistics for the Header pool
Number of Buffers used by packets generated by system:  511
Number of Buffers used by incoming packets:               0

Statistics for the FS Header pool
Caller pc    : 0x608F68FC count:        9
Resource User:       Init count:       12
Caller pc    : 0x61A21D3C count:        1
Caller pc    : 0x60643FF8 count:        1
Caller pc    : 0x61C526C4 count:        1
Number of Buffers used by packets generated by system:   28
Number of Buffers used by incoming packets:               0
```

Use this command with the optional keyword and argument to display the details of the buffer usage pattern in a small buffer pool, for example:

```
Router# show buffers usage pool small

Statistics for the Small pool
Caller pc    : 0x626BA9E0 count:       20
Resource User: EEM ED Sys count:       20
Caller pc    : 0x60C71F8C count:        1
Resource User:       Init count:        1
Number of Buffers used by packets generated by system:   62
Number of Buffers used by incoming packets:               0
```

Step 4 **show memory [processor | io] fragment [detail]**

Use this command without the optional keywords to display the block details of every allocated block for both I/O memory and processor memory, for example:

```
Router# show memory fragment

Processor memory

Free memory size : 211014448 Number of free blocks:      139
Allocator PC Summary for allocated blocks in pool: Processor

    PC          Total     Count  Name
0x6189A438     318520       1  RTPSPI
0x6205711C     237024       2  CCH323_CT
0x6080BE38      98416       2  Exec
0x606AD988      80256       1  Init
0x618F68A8      73784       1  CCSIP_UDP_SOCKET
0x6195AD04      67640       1  QOS_MODULE_MAIN
0x606488C8      65592       1  CEF: Adjacency chunk
0x60635620      65592       1  CEF: 16 path chunk pool
0x615ECE58      65592       1  XTagATM VC chunk
```

```
0x6165ACF8      65592       1  eddri_self_event
0x608DE168      65592       1  MallocLite
0x60857920      51020      11  Normal
0x6203BF88      42480       4  IPv6 CEF fib tables
0x60DC7F14      32824       1  PPP Context Chunks
.
.
.
I/O memory

Free memory size : 14700024 Number of free blocks:       52
Allocator PC Summary for allocated blocks in pool: I/O

    PC          Total   Count   Name
0x60857934    3936000      60  FastEthernet0/
0x60857898     524800       8  FastEthernet0/0
0x601263CC      29120       7  Init
0x6082DB28       9408      23  *Packet Data*
0x60126344       8448       4  Init

Allocator PC Summary for free blocks in pool: I/O

    PC          Total    Count   Name
0x608C5730   29391444       1  (coalesced)
0x608FC1F4       5376      28  (fragment)
0x6082DB28       4288      14  (fragment)
```

Use this command with the **detail** optional keyword to display the block details of every allocated block for both I/O memory and processor memory, for example:

```
Router# show memory fragment detail

Processor memory

Free memory size : 211038812 Number of free blocks:      139
 Address      Bytes     Prev     Next Ref     PrevF    NextF Alloc PC  what
644AAB70 0000001032 644AAB20 644AAFAC 001  -------- -------- 620450F8  Index Table Block
644AAFAC 0000000028 644AAB70 644AAFFC 000  0        6448CB5C 607B2ADC  NameDB String
644AAFFC 0000000076 644AAFAC 644AB07C 001  -------- -------- 60818DE0  Init
6448CB0C 0000000028 6448CABC 6448CB5C 001  -------- -------- 607F8380  Cond Debug
definition
6448CB5C 0000000028 6448CB0C 6448CBAC 000  644AAFAC 6489F158 607B2ADC  NameDB String
6448CBAC 0000000028 6448CB5C 6448CBFC 001  -------- -------- 607F8380  Cond Debug
definition
6489EF8C 0000000408 6489DBCC 6489F158 001  -------- -------- 60857920  Normal
6489F158 0000000064 6489EF8C 6489F1CC 000  6448CB5C 6448CABC 607B2ADC  NameDB String
6489F1CC 0000005004 6489F158 648A058C 001  -------- -------- 60857920  Normal
6448CA6C 0000000028 6448C9AC 6448CABC 001  -------- -------- 607D72FC  Parser Linkage
6448CABC 0000000028 6448CA6C 6448CB0C 000  6489F158 644949C8 607B2ADC  NameDB String
6448CB0C 0000000028 6448CABC 6448CB5C 001  -------- -------- 607F8380  Cond Debug
definition
64494978 0000000028 64494928 644949C8 001  -------- -------- 607D72FC  Parser Linkage
644949C8 0000000028 64494978 64494A18 000  6448CABC 654F2868 607B2ADC  NameDB String
64494A18 0000000028 644949C8 64494A68 001  -------- -------- 607D72FC  Parser Linkage
654F27E8 0000000076 654F2768 654F2868 001  -------- -------- 60818DE0  Init
654F2868 0000000076 654F27E8 654F28E8 000  644949C8 654F1BE8 60818DE0  Init
.
.
.
I/O memory

Free memory size : 14700024 Number of free blocks:       52
 Address      Bytes     Prev     Next Ref     PrevF    NextF Alloc PC  what
0E000000 0000000056 00000000 0E00006C 000  0        E176F4C  00000000  (fragment)
```

```
0E00006C 0000000268 0E000000 0E0001AC 001  -------- -------- 6082DB28  *Packet Data*
0E176E0C 0000000268 0E176CCC 0E176F4C 001  -------- -------- 6082DB28  *Packet Data*
0E176F4C 0000000076 0E176E0C 0E176FCC 000  E000000  E209F4C  6082DB28  (fragment)
0E176FCC 0000002060 0E176F4C 0E17780C 001  -------- -------- 60126344  Init
0E209E0C 0000000268 0E209CCC 0E209F4C 001  -------- -------- 6082DB28  *Packet Data*
0E209F4C 0000000076 0E209E0C 0E209FCC 000  E176F4C  E29CF4C  6082DB28  (fragment)
0E209FCC 0000002060 0E209F4C 0E20A80C 001  -------- -------- 60126344  Init
0E29CE0C 0000000268 0E29CCCC 0E29CF4C 001  -------- -------- 6082DB28  *Packet Data*
0E29CF4C 0000000076 0E29CE0C 0E29CFCC 000  E209F4C  E32FF4C  6082DB28  (fragment)
0E29CFCC 0000002060 0E29CF4C 0E29D80C 001  -------- -------- 60126344  Init
0E32FE0C 0000000268 0E32FCCC 0E32FF4C 001  -------- -------- 6082DB28  *Packet Data*
0E32FF4C 0000000076 0E32FE0C 0E32FFCC 000  E29CF4C  0        6082DB28  (fragment)
0E32FFCC 0000002060 0E32FF4C 0E33080C 001  -------- -------- 60126344  Init
0E177FCC 0000004108 0E177E4C 0E17900C 001  -------- -------- 601263CC  Init
0E17900C 0000000140 0E177FCC 0E1790CC 000  0        E18910C  601263CC  (fragment)
```

Use this command with **detail** optional keyword to display the block details of every allocated block for processor memory, for example:

```
Router# show memory processor fragment detail

Processor memory

Free memory size : 65566148 Number of free blocks:      230
 Address      Bytes     Prev     Next Ref     PrevF    NextF Alloc PC  what
645A8148 0000000028 645A80F0 645A8194 001  -------- -------- 60695B20  Init
645A8194 0000000040 645A8148 645A81EC 000  0        200B4300 606B9614  NameDB String
645A81EC 0000000260 645A8194 645A8320 001  -------- -------- 607C2D20  Init
200B42B4 0000000028 200B4268 200B4300 001  -------- -------- 62366C80  Init
200B4300 0000000028 200B42B4 200B434C 000  645A8194 6490F7E8 60976574  AAA Event Data
200B434C 0000002004 200B4300 200B4B50 001  -------- -------- 6267D294  Coproc Request
Structures
6490F79C 0000000028 6490F748 6490F7E8 001  -------- -------- 606DDA04  Parser Linkage
6490F7E8 0000000028 6490F79C 6490F834 000  200B4300 6491120C 606DD8D8  Init
6490F834 0000006004 6490F7E8 64910FD8 001  -------- -------- 607DF5BC  Process Stack
649111A0 0000000060 64911154 6491120C 001  -------- -------- 606DE82C  Parser Mode
6491120C 0000000028 649111A0 64911258 000  6490F7E8 500770F0 606DD8D8  Init
64911258 0000000200 6491120C 64911350 001  -------- -------- 603F0E38  Init
.
20000000 0000000828 5C3AEB24 2000036C 001  -------- -------- 60734010  *Packet Header*
6500BF94 0000000828 6500BC28 6500C300 001  -------- -------- 60734010  *Packet Header*
6500C300 0004760912 6500BF94 50000000 000  5C3AEB24 2C42E310 6071253C  (coalesced)
50000000 0000000828 6500C300 5000036C 001  -------- -------- 60734010  *Packet Header*
2C42E0B4 0000000556 2C429430 2C42E310 001  -------- -------- 60D4A0B4  Virtual Exec
2C42E310 0062725312 2C42E0B4 00000000 000  6500C300 0        6071253C  (coalesced)
```

Use this command with **detail** optional keyword to display the block details of every allocated block for I/O memory, for example:

```
Router# show memory io fragment detail

0E3F8BAC 0000000204 0E3F8AAC 0E3F8CAC 001  -------- -------- 608C5730  test memory
0E3F8CAC 0000000204 0E3F8BAC 0E3F8DAC 000  0        E3F8AAC  608C5730  test memory
0E3F8DAC 0000000204 0E3F8CAC 0E3F8EAC 001  -------- -------- 608C5730  test memory
0E3F89AC 0000000204 0E3F88AC 0E3F8AAC 001  -------- -------- 608C5730  test memory
0E3F8AAC 0000000204 0E3F89AC 0E3F8BAC 000  E3F8CAC  E3F88AC  608C5730  test memory
0E3F8BAC 0000000204 0E3F8AAC 0E3F8CAC 001  -------- -------- 608C5730  test memory
0E3F87AC 0000000204 0E3F86AC 0E3F88AC 001  -------- -------- 608C5730  test memory
0E3F88AC 0000000204 0E3F87AC 0E3F89AC 000  E3F8AAC  E3F86AC  608C5730  test memory
0E3F89AC 0000000204 0E3F88AC 0E3F8AAC 001  -------- -------- 608C5730  test memory
0E3F85AC 0000000204 0E3F826C 0E3F86AC 001  -------- -------- 608C5730  test memory
0E3F86AC 0000000204 0E3F85AC 0E3F87AC 000  E3F88AC  0        608C5730  test memory
0E3F87AC 0000000204 0E3F86AC 0E3F88AC 001  -------- -------- 608C5730  test memory
0E3F4E6C 0000000268 0E3F4D2C 0E3F4FAC 000  0        E3F5BEC  608C5730  test memory
```

```
0E3F5BEC 0000000268 0E3F5AAC 0E3F5D2C 000  E3F4E6C  E3EE56C  608C5730  test memory
0E3EE46C 0000000204 0E3EE12C 0E3EE56C 001  -------- -------- 608C5730  test memory
0E3EEFAC 0000000204 0E3EEE6C 0E3EF0AC 001  -------- -------- 608C5730  test memory
0E3F06EC 0000000204 0E3F03AC 0E3F07EC 001  -------- -------- 608C5730  test memory
0E3F8DAC 0000000204 0E3F8CAC 0E3F8EAC 001  -------- -------- 608C5730  test memory
```

Step 5 show memory statistics history table

Use this command to display the history of memory consumption, for example:

```
Router# show memory statistics history table

History for Processor memory

Time: 15:48:56.806
Used(b): 422748036 Largest(b): 381064952 Free blocks :291
Maximum memory users for this period
Process Name           Holding   Num Alloc
Virtual Exec             26992          37
TCP Protocols            14460           6
IP Input                  1212           1

Time: 14:42:54.506
Used(b): 422705876 Largest(b): 381064952 Free blocks :296
Maximum memory users for this period
Process Name           Holding   Num Alloc
Exec                  400012740          24
Dead                   1753456          90
Pool Manager            212796         257

Time: 13:37:26.918
Used(b): 20700520 Largest(b): 381064952 Free blocks :196
Maximum memory users for this period
Process Name           Holding   Num Alloc
Exec                      8372           5

Time: 12:39:44.422
Used(b): 20701436 Largest(b): 381064952 Free blocks :193

Time: 11:46:25.135
Used(b): 20701436 Largest(b): 381064952 Free blocks :193
Maximum memory users for this period
Process Name           Holding   Num Alloc
CDP Protocol              3752          25
.
.
.
History for I/O memory

Time: 15:48:56.809
Used(b):  7455520 Largest(b): 59370080 Free blocks :164

Time: 14:42:54.508
Used(b):  7458064 Largest(b): 59370080 Free blocks :165
Maximum memory users for this period
Process Name           Holding   Num Alloc
Pool Manager            141584         257

Time: 13:37:26.920
Used(b):  7297744 Largest(b): 59797664 Free blocks :25

Time: 12:39:44.424
Used(b):  7297744 Largest(b): 59797664 Free blocks :25
.
```

```
.
.
Time: 09:38:53.040
Used(b):  7297744 Largest(b): 59797664 Free blocks :25

Time: 01:02:05.533
Used(b):  7308336 Largest(b): 59797664 Free blocks :23

Time: 00:00:17.937
Used(b):  7308336 Largest(b): 59797664 Free blocks :23
Maximum memory users for this period
Process Name           Holding   Num Alloc
Init                   7296000         214
Pool Manager               816           3
```

Step 6 **show monitor event-trace cpu-report** {**brief** {**all** [**detail**] | **back** *time* | **clock** *time* | **from-boot** [*seconds* | **detail**] | **latest** [**detail**]} | **handle** *handle-number*}

Use this command to view a brief CPU report details for event tracing on a networking device, for example:

```
Router# show monitor event-trace cpu-report brief all

Timestamp   : Handle Name               Description
00:01:07.320:  1     CPU                None
```

Use this command to view a brief CPU report details for event tracing on a networking device, for example:

```
Router# show monitor event-trace cpu-report handle 1

00:01:07.320:  1     CPU                None
################################################################################
Global Statistics
-----------------
5 sec CPU util 0%/0% Timestamp 21:03:56
Queue Statistics
----------------
             Exec Count    Total CPU  Response Time       Queue Length
                                        (avg/max)           (avg/max)
Critical           1          0          0/0                 1/1
High               5          0          0/0                 1/1
Normal           178          0          0/0                 2/9
Low               15          0          0/0                 2/3
Common Process Information
------------------------------
 PID   Name         Prio Style
------------------------------
  10 AAA high-capacit M  New
 133 RADIUS TEST CMD  M  New
  47 VNM DSPRM MAIN   H  New
  58 TurboACL         M  New
  97 IP Background    M  New
  99 CEF: IPv4 proces L  New
 112 X.25 Background  M  New
 117 LFDp Input Proc  M  New
   3 Init             M  Old
CPU Intensive processes
--------------------------------------------------------------------------------
 PID Total       Exec    Quant         Burst  Burst size  Schedcall  Schedcall
     CPUms      Count   avg/max        Count avg/max(ms)      Count Per avg/max
--------------------------------------------------------------------------------
   3   820          6   136/236         1     24/24              18  887/15172
Priority Suspends
```

```
-----------------------------------
 PID Exec Count Prio-Susps
-----------------------------------
   3          6          1

Latencies
-------------------------
 PID     Exec Count Latency
                    avg/max
-------------------------
  10          1 15192/15192
 133          1 15192/15192
  58          1 15192/15192
 112          1 15192/15192
 117          1 15192/15192
  99          1 15172/15172
  47          1 15172/15172
  97          1 15172/15172
################################################################################
################################################################################
Global Statistics
-----------------
5 sec CPU util 0%/0% Timestamp 00:00:00
Queue Statistics
----------------
         Exec Count  Total CPU    Response Time           Queue Length
                                    (avg/max)                (avg/max)
Critical      0          0              0/0                    0/0
High          0          0              0/0                    0/0
Normal        0          0              0/0                    0/0
Low           0          0              0/0                    0/0

Common Process Information
------------------------------
 PID Name             Prio Style
------------------------------

CPU Intensive processes
--------------------------------------------------------------------------------
 PID Total       Exec    Quant         Burst  Burst size  Schedcall  Schedcall
     CPUms      Count   avg/max        Count avg/max(ms)      Count Per avg/max
--------------------------------------------------------------------------------
Priority Suspends
-----------------------------------
 PID Exec Count Prio-Susps
-----------------------------------
Latencies
-------------------------
 PID Exec Count   Latency
                  avg/max
-------------------------
################################################################################
```

Step 7 **show processes cpu autoprofile hog**

Use this command to view the CPUHOG autoprofile data, for example:

```
Router# show processes cpu autoprofile hog

0x6075DD40 0x60755638
0x6075DD24 0x60755638
0x6075563C 0x60755638
0x60755638 0x60755638
0x60755638 0x60755638
```

```
0x6075DD10 0x60755638
0x6075DD40 0x60755638
0x6075DD40 0x60755638
0x6075563C 0x60755638
0x6075DCE0 0x60755638
0x6075DD44 0x60755638
0x6075DCCC 0x60755638
0x6075DD10 0x60755638
.
.
.
0x6075DD3C 0x60755638
0x6075DD38 0x60755638
0x6075DD10 0x60755638
0x6075DCCC 0x60755638
0x6075DCDC 0x60755638
0x6075563C 0x60755638
0x6075DD3C 0x60755638
0x6075DD20 0x60755638
0x6075DD58 0x60755638
0x6075DD1C 0x60755638
0x6075DD10 0x60755638
0x6075DCDC 0x60755638
0x6075DCF8 0x60755638
```

Step 8 **show processes cpu extended** [**history**]

Use this command to view an extended CPU load report, for example:

```
Router# show processes cpu extended

################################################################################
Global Statistics
-----------------
5 sec CPU util 0%/0% Timestamp 21:03:56
Queue Statistics
----------------

             Exec Count  Total CPU    Response Time         Queue Length
                                       (avg/max)             (avg/max)
Critical           1          0          0/0                   1/1
High               5          0          0/0                   1/1
Normal           178          0          0/0                   2/9
Low               15          0          0/0                   2/3
Common Process Information
--------------------------------
 PID Name            Prio Style
--------------------------------
CPU Intensive processes
--------------------------------------------------------------------------------
 PID Total       Exec    Quant         Burst  Burst size  Schedcall  Schedcall
     CPUms      Count   avg/max        Count avg/max(ms)      Count Per avg/max
--------------------------------------------------------------------------------
Priority Suspends
-----------------------------------
 PID Exec Count Prio-Susps
-----------------------------------
Latencies
-------------------------
 PID Exec Count   Latency
                  avg/max
-------------------------

################################################################################
```

Step 9 **show resource all [brief | detailed]**

Use this command without the optional keywords to display the resource details, for example:

```
Router# show resource all

Resource Owner: cpu
Resource User Type: iosprocess
Resource User: Init(ID: 0x1000001)
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
16777217           0         0          0  0.00%  0.00%  0.00% Init
  Resource User: Scheduler(ID: 0x1000002)
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
16777218           0         0          0  0.00%  0.00%  0.00% Scheduler
  Resource User: Dead(ID: 0x1000003)
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
16777219           0         0          0  0.00%  0.00%  0.00% Dead
  Resource User: Interrupt(ID: 0x1000004)
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
16777220           0         0          0  0.00%  0.00%  0.00% Interrupt
  Resource User: Memory RO RU(ID: 0x1000005)
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
16777221           0         0          0  0.00%  0.00%  0.00% Memory RO RU
  Resource User: Chunk Manager(ID: 0x1000006)
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
16777222           0        13          0  0.00%  0.00%  0.00% Chunk Manager
  Resource User: Load Meter(ID: 0x1000007)
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
16777223        2872     36029         79  0.00%  0.00%  0.00% Load Meter
  Resource User: Check heaps(ID: 0x1000009)
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
16777225      352744     33446      10546  0.00%  0.20%  0.17% Check heaps
  Resource User: Pool Manager(ID: 0x100000A)
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
16777226           0         1          0  0.00%  0.00%  0.00% Pool Manager
  Resource User: Buffer RO RU(ID: 0x100000B)
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
16777227           0         0          0  0.00%  0.00%  0.00% Buffer RO RU
  Resource User: Timers(ID: 0x100000C)
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
16777228           0         2          0  0.00%  0.00%  0.00% Timers
  Resource User: Serial Background(ID: 0x100000D)
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
16777229           0         2          0  0.00%  0.00%  0.00% Serial Backgroun
  Resource User: AAA_SERVER_DEADTIME(ID: 0x100000E)
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
16777230           0         1          0  0.00%  0.00%  0.00% AAA_SERVER_DEADT
  Resource User: AAA high-capacity counters(ID: 0x100000F)
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
16777231           0         2          0  0.00%  0.00%  0.00% AAA high-capacit
  Resource User: Policy Manager(ID: 0x1000010)
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
16777232           0         1          0  0.00%  0.00%  0.00% Policy Manager
  Resource User: Crash writer(ID: 0x1000011)
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
16777233           0         1          0  0.00%  0.00%  0.00% Crash writer
  Resource User: RO Notify Timers(ID: 0x1000012)
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
16777234           0         1          0  0.00%  0.00%  0.00% RO Notify Timers
  Resource User: RMI RM Notify Watched Policy(ID: 0x1000013)
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
16777235           0         1          0  0.00%  0.00%  0.00% RMI RM Notify Wa
  Resource User: EnvMon(ID: 0x1000014)
```

```
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
16777236       11164     92859        120  0.00%  0.00%  0.00% EnvMon
  Resource User: IPC Dynamic Cache(ID: 0x1000015)
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
16777237           0      3004          0  0.00%  0.00%  0.00% IPC Dynamic Cach
  Resource User: IPC Periodic Timer(ID: 0x1000017)
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
16777239           0    180082          0  0.00%  0.00%  0.00% IPC Periodic Tim
  Resource User: IPC Managed Timer(ID: 0x1000018)
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
16777240         572     79749          7  0.00%  0.00%  0.00% IPC Managed Time
  Resource User: IPC Deferred Port Closure(ID: 0x1000019)
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
16777241           4    180088          0  0.00%  0.00%  0.00% IPC Deferred Por
  Resource User: IPC Seat Manager(ID: 0x100001A)
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
16777242       97560   1408799         69  0.23%  0.02%  0.00% IPC Seat Manager
  Resource User: IPC Session Service(ID: 0x100001B)
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
16777243           0         1          0  0.00%  0.00%  0.00% IPC Session Serv
  Resource User: ARP Input(ID: 0x100001C)
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
16777244          20      3082          6  0.00%  0.00%  0.00% ARP Input
  Resource User: EEM ED Syslog(ID: 0x100001D)
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
16777245           0        49          0  0.00%  0.00%  0.00% EEM ED Syslog
  Resource User: DDR Timers(ID: 0x100001E)
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
16777246           0         2          0  0.00%  0.00%  0.00% DDR Timers
  Resource User: Dialer event(ID: 0x100001F)
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
16777247           0         2          0  0.00%  0.00%  0.00% Dialer event
  Resource User: Entity MIB API(ID: 0x1000020)
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
16777248          28        16       1750  0.00%  0.00%  0.00% Entity MIB API
.
.
.
Resource User: draco-oir-process:slot 2(ID: 0x100011E)
Getbufs  Retbufs  Holding  RU Name
0        0        0        draco-oir-proces

  Resource User: SCP async: Draco-LC4(ID: 0x1000125)
Getbufs  Retbufs  Holding  RU Name
35849    243101   4294760044 SCP async: Draco

  Resource User: IFCOM Msg Hdlr(ID: 0x1000127)
Getbufs  Retbufs  Holding  RU Name
2        2        0        IFCOM Msg Hdlr

  Resource User: IFCOM Msg Hdlr(ID: 0x1000128)
Getbufs  Retbufs  Holding  RU Name
28       28       0        IFCOM Msg Hdlr

  Resource User: Exec(ID: 0x100012C)
Getbufs  Retbufs  Holding  RU Name
912      912      0        Exec

Resource Owner: test_mem
 Resource User Type: test_process
 Resource User Type: mem_rut
Resource Owner: test_cpu
 Resource User Type: test_process
 Resource User Type: cpu_rut
```

Step 10 **show resource database**

Use this command to display the resource database details, for example:

```
Router# show resource database

List of all Resource Owners :
Owner: cpu                      Id:0x1
Owner's list of monitors is empty.
Owner: memory                   Id:0x2
Owner's list of monitors is empty.
Owner: Buffer                   Id:0x3
Owner's list of monitors is empty.
Owner: test_mem                 Id:0x4
Owner's list of monitors is empty.
Owner: test_cpu                 Id:0x5
Owner's list of monitors is empty.
Owner: test_RO0                 Id:0x7
Owner's list of monitors is empty.
Owner: test_RO1                 Id:0x8
Owner's list of monitors is empty.
Owner: test_RO2                 Id:0x9
Owner's list of monitors is empty.
Owner: test_RO3                 Id:0xA
Owner's list of monitors is empty.
.
.
.
Resource Monitor: test_ROM0, ID: 0x1B
 Not Watching any Relations.
 Not Watching any Policies.
Resource Monitor: test_ROM1, ID: 0x1C
 Not Watching any Relations.
 Not Watching any Policies.
Resource Monitor: test_ROM2, ID: 0x1D
 Not Watching any Relations.
 Not Watching any Policies.
```

Step 11 **show resource owner** {*resource-owner-name* | **all**} **user** {*resource-user-type-name* | **all**} [**brief** | **detailed** | **triggers**]

Use this command to display the resource owner details, for example:

```
Router# show resource owner all user all

Resource Owner: cpu
 Resource User Type: iosprocess
  Resource User: Init(ID: 0x1000001)
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
16777217           0         0          0  0.00%  0.00%  0.00% Init
  Resource User: Scheduler(ID: 0x1000002)
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
16777218           0         0          0  0.00%  0.00%  0.00% Scheduler
  Resource User: Dead(ID: 0x1000003)
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
16777219           0         0          0  0.00%  0.00%  0.00% Dead
  Resource User: Interrupt(ID: 0x1000004)
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
16777220           0         0          0  0.00%  0.00%  0.00% Interrupt
  Resource User: Memory RO RU(ID: 0x1000005)
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
16777221           0         0          0  0.00%  0.00%  0.00% Memory RO RU
  Resource User: Chunk Manager(ID: 0x1000006)
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
16777222           4         3       1333  0.00%  0.00%  0.00% Chunk Manager
```

```
  Resource User: Load Meter(ID: 0x1000007)
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
16777223           4       292         13  0.00%  0.00%  0.00% Load Meter
  Resource User: Check heaps(ID: 0x1000009)
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
16777225         376       192       1958  0.00%  0.02%  0.00% Check heaps
  Resource User: Pool Manager(ID: 0x100000A)
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
16777226           0         1          0  0.00%  0.00%  0.00% Pool Manager
  Resource User: Buffer RO RU(ID: 0x100000B)
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
16777227           0         0          0  0.00%  0.00%  0.00% Buffer RO RU
  Resource User: Timers(ID: 0x100000C)
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
16777228           0         2          0  0.00%  0.00%  0.00% Timers
  Resource User: Serial Background(ID: 0x100000D)
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
16777229           0         2          0  0.00%  0.00%  0.00% Serial Backgroun
  Resource User: ALARM_TRIGGER_SCAN(ID: 0x100000E)
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
16777230           0       268          0  0.00%  0.00%  0.00% ALARM_TRIGGER_SC
  Resource User: AAA_SERVER_DEADTIME(ID: 0x100000F)
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
16777231           0         1          0  0.00%  0.00%  0.00% AAA_SERVER_DEADT
  Resource User: AAA high-capacity counters(ID: 0x1000010)
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
.
.
.
Resource User Type: test_RUT143

Resource User Type: test_RUT144
Resource User Type: test_RUT145

Resource User Type: test_RUT146
Resource User Type: test_RUT147
```

Step 12 **show resource relationship user** *resource-user-type*

Use this command to display the relationship details between different resource owners, for example:

```
Router# show resource relationship

Resource User Type: iosprocess (ID: 0x1)
 -> Resource Owner: cpu (ID: 0x1)
 -> Resource Owner: memory (ID: 0x2)
 -> Resource Owner: Buffer (ID: 0x3)
 -> Resource User: Init (ID: 0x1000001)
 -> Resource User: Scheduler (ID: 0x1000002)
 -> Resource User: Dead (ID: 0x1000003)
 -> Resource User: Interrupt (ID: 0x1000004)
 -> Resource User: Memory RO RU (ID: 0x1000005)
 -> Resource User: Chunk Manager (ID: 0x1000006)
 -> Resource User: Load Meter (ID: 0x1000007)
 -> Resource User: Check heaps (ID: 0x1000009)
 -> Resource User: Pool Manager (ID: 0x100000A)
 -> Resource User: Buffer RO RU (ID: 0x100000B)
 -> Resource User: Timers (ID: 0x100000C)
 -> Resource User: Serial Background (ID: 0x100000D)
 -> Resource User: ALARM_TRIGGER_SCAN (ID: 0x100000E)
 -> Resource User: AAA_SERVER_DEADTIME (ID: 0x100000F)
 -> Resource User: AAA high-capacity counters (ID: 0x1000010)
 -> Resource User: Policy Manager (ID: 0x1000011)
 -> Resource User: Crash writer (ID: 0x1000012)
```

```
 -> Resource User: RO Notify Timers (ID: 0x1000013)
 -> Resource User: RMI RM Notify Watched Policy (ID: 0x1000014)
 -> Resource User: EnvMon (ID: 0x1000015)
 -> Resource User: OIR Handler (ID: 0x1000016)
 -> Resource User: IPC Dynamic Cache (ID: 0x1000017)
 -> Resource User: IPC Zone Manager (ID: 0x1000018)
 -> Resource User: IPC Periodic Timer (ID: 0x1000019)
 -> Resource User: IPC Managed Timer (ID: 0x100001A)
 -> Resource User: IPC Deferred Port Closure (ID: 0x100001B)
 -> Resource User: IPC Seat Manager (ID: 0x100001C)
 -> Resource User: IPC Session Service (ID: 0x100001D)
 -> Resource User: Compute SRP rates (ID: 0x100001E)
 -> Resource User: ARP Input (ID: 0x100001F)
 -> Resource User: DDR Timers (ID: 0x1000020)
 -> Resource User: Dialer event (ID: 0x1000021)
 -> Resource User: Entity MIB API (ID: 0x1000022)
 -> Resource User: SERIAL A'detect (ID: 0x1000023)
 -> Resource User: GraphIt (ID: 0x1000024)
 -> Resource User: HC Counter Timers (ID: 0x1000025)
 -> Resource User: Critical Bkgnd (ID: 0x1000026)
 -> Resource User: Net Background (ID: 0x1000027)
 -> Resource User: Logger (ID: 0x1000028)
.
.
.
Resource User Type: test_RUT141 (ID: 0x92)
 -> Resource Owner: test_RO0 (ID: 0x7)
 Resource User Type: test_RUT142 (ID: 0x93)
 -> Resource Owner: test_RO0 (ID: 0x7)
 Resource User Type: test_RUT143 (ID: 0x94)
 -> Resource Owner: test_RO0 (ID: 0x7)
 Resource User Type: test_RUT144 (ID: 0x95)
 -> Resource Owner: test_RO0 (ID: 0x7)
 Resource User Type: test_RUT145 (ID: 0x96)
 -> Resource Owner: test_RO0 (ID: 0x7)
 Resource User Type: test_RUT146 (ID: 0x97)
 -> Resource Owner: test_RO0 (ID: 0x7)
 Resource User Type: test_RUT147 (ID: 0x98)
 -> Resource Owner: test_RO0 (ID: 0x7)
 Resource User Type: test_RUT148 (ID: 0x99)
 -> Resource Owner: test_RO0 (ID: 0x7)
 Resource User Type: test_RUT149 (ID: 0x9A)
 -> Resource Owner: test_RO0 (ID: 0x7)
```

Step 13 **show resource user** {**all** | *resource-user-type*} [**brief** | **detailed**]

Use this command to display the relationship details between different ROs, for example:

```
Router# show resource user all

Resource User Type: iosprocess
Resource Grp: Init
Resource Owner: memory
Processor memory
Allocated   Freed  Holding   Blocks
27197780  8950144 18247636     6552

I/O memory
Allocated   Freed  Holding   Blocks
 7296000     9504  7286496      196

Resource Owner: cpu
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
16777224       14408       116     124206 100.40%  8.20%  1.70% Init
```

```
Resource Owner: Buffer
Getbufs  Retbufs  Holding  RU Name
332      60       272      Init

Resource User: Init
Resource User: Scheduler
Resource Owner: memory
Processor memory
Allocated   Freed  Holding   Blocks
   77544        0    77544        2

Resource Owner: cpu
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
16777218           0         0          0  0.00%  0.00%  0.00% Scheduler
Resource Owner: Buffer
Getbufs  Retbufs  Holding  RU Name
0        0        0        Scheduler

Resource User: Dead
Resource Owner: memory
Processor memory
Allocated   Freed  Holding   Blocks
 1780540      260  1780280      125
.
.
.

 Resource User: BGP Scanner
  Resource Owner: memory
Processor memory
Allocated   Freed  Holding   Blocks
    9828     9828        0        0

  Resource Owner: cpu
    RUID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min Res Usr
16777406         660       659       1001  0.00%  0.00%  0.00% BGP Scanner
  Resource Owner: Buffer
Getbufs  Retbufs  Holding  RU Name
0        0        0        BGP Scanner
Resource User Type: test_process
Resource User Type: mem_rut
Resource User Type: cpu_rut
```

Troubleshooting Tips

To trace and troubleshoot the notification and registration activities for resources using the Embedded Resource Manager feature, use the following suggested techniques.

- Enable debugging of resource registration using the **debug resource policy registration** command in privileged EXEC mode.
- Enable debugging of resource manager notification using the **debug resource policy notification** command in privileged EXEC mode.

SUMMARY STEPS

1. **enable**
2. **debug resource policy registration**
3. **debug resource policy notification** [**owner** *resource-owner-name*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**debug resource policy registration** **Example:** `Router# debug resource policy registration`	Enables debugging on resource policy registration.
Step 3	**debug resource policy notification** [**owner** *resource-owner-name*] **Example:** `Router# debug resource policy notification owner cpu`	Enables notification debugging on ROs.

Examples

Use the **debug resource policy registration** command to trace the resource manager registration information, for example:

```
Router# debug resource policy registration

Registrations debugging is on

When a Resource User is created
*Mar  3 09:35:58.304: resource_user_register: RU: ruID: 0x10000B8, rutID: 0x1, rg_ID: 0x0
name: usrr1

When a Resource User is deleted
*Mar  3 09:41:09.500: resource_user_unregister: RU: ruID: 0x10000B8, rutID: 0x1, rg_ID:
0x0 name: usrr1
```

Use the **debug resource policy notification** [**owner** *resource-owner-name*] command to trace the resource policy notification information, for example:

```
Router# debug resource policy notification

Enabled notif. debugs on all owners
```

When a threshold is exceeded, you would see these messages:

```
*Mar  3 09:50:44.081: Owner: 'memory' initiated a notification:
*Mar  3 09:50:44.081: %SYS-4-RESMEMEXCEED: Resource user usrr1 has exceeded the Major
memory threshold
Pool: Processor Used: 42932864 Threshold :42932860
*Mar  3 09:50:46.081: Notification from Owner: 'memory' is dispatched for User: 'usrr1'
(ID: 0x10000B9)
*Mar  3 09:50:46.081: %SYS-4-RESMEMEXCEED: Resource user usrr1 has exceeded the Major
memory threshold
Pool: Processor Used: 42932864 Threshold :42932860

Router# no debug resource policy notification

Disabled notif. debugs on all owners
```

```
Router# debug resource policy notification owner cpu

Enabled notif. debugs on owner 'cpu'

Router# no debug resource policy notification owner cpu

Disabled notif. debugs on owner 'cpu'

Router# debug resource policy notification owner memory

Enabled notif. debugs on owner 'memory'

Router# no debug resource policy notification owner memory

Disabled notif. debugs on owner 'memory'

Router# debug resource policy notification owner Buffer

Enabled notif. debugs on owner 'Buffer'

Router# no debug resource policy notification owner Buffer

Disabled notif. debugs on owner 'Buffer'
```

Configuration Examples for Embedded Resource Manager

This section provides the following configuration examples:

Managing Resource Utilization by Defining Resource Policy: Example

The following example shows how to configure a global resource policy with the policy name system-global-pc1:

```
configure terminal
resource policy
policy system-global-pc1 global
```

The following example shows how to configure a per user global resource policy with the policy name per-user-global-pc1 and the resource type as iosprocess:

```
configure terminal
resource policy
policy per-user-global-pc1 type iosprocess
```

The following example shows how to configure a user local resource policy with the policy name user-local-pc1 and the resource type as iosprocess:

```
configure terminal
resource policy
policy user-local-pc1 type iosprocess
```

Setting Expected Operating Ranges for Resource Owners: Example

The following example shows how to configure various thresholds for buffer, CPU, and memory ROs.

Configuring System Global Thresholding Policy for Buffer RO

The following example shows how to configure a global policy with the policy name as system-global-pc1 for public buffer with critical threshold values of 90 percent as rising at an interval of 12 seconds, 20 percent as falling at an interval of 10 seconds, major threshold values of 70 percent as rising at an interval of 12 seconds, 15 percent as falling at an interval of 10 seconds, and minor threshold values of 60 percent as rising at an interval of 12 seconds, 10 percent as falling at an interval of 10 seconds:

```
configure terminal
resource policy
policy system-global-pc1 global
system
buffer public
critical rising 90 interval 12 falling 20 interval 10
major rising 70 interval 12 falling 15 interval 10
minor rising 60 interval 12 falling 10 interval 10
```

Configuring Per User Global Thresholding Policy for Buffer RO

The following example shows how to configure a per user global policy with the policy name as per-user-global-pc1 for public buffer with critical threshold values of 90 percent as rising at an interval of 12 seconds, 20 percent as falling at an interval of 10 seconds, major threshold values of 70 percent as rising at an interval of 12 seconds, 15 percent as falling at an interval of 10 seconds, and minor threshold values of 60 percent as rising at an interval of 12 seconds, 10 percent as falling at an interval of 10 seconds:

```
configure terminal
resource policy
policy per-user-global-pc1 type iosprocess
system
buffer public
critical rising 90 interval 12 falling 20 interval 10 global
major rising 70 interval 12 falling 15 interval 10 global
minor rising 60 interval 12 falling 10 interval 10 global
```

Configuring User Local Thresholding Policy for Buffer RO

The following example shows how to configure a user local policy with the policy name as user-local-pc1 for public buffer with critical threshold values of 90 percent as rising at an interval of 12 seconds, 20 percent as falling at an interval of 10 seconds, major threshold values of 70 percent as rising at an interval of 12 seconds, 15 percent as falling at an interval of 10 seconds, and minor threshold values of 60 percent as rising at an interval of 12 seconds, 10 percent as falling at an interval of 10 seconds:

```
configure terminal
resource policy
policy user-local-pc1 type iosprocess
system
buffer public
critical rising 70 interval 12 falling 20 interval 10
major rising 70 interval 12 falling 15 interval 10
minor rising 60 interval 12 falling 10 interval 10
```

Configuring System Global Thresholding Policy for I/O Memory RO

The following example shows how to configure a global policy with the policy name as system-global-pc1 for I/O memory with critical threshold values of 90 percent as rising at an interval of 12 seconds, 20 percent as falling at an interval of 10 seconds, major threshold values of 70 percent as rising at an interval of 12 seconds, 15 percent as falling at an interval of 10 seconds, and minor threshold values of 60 percent as rising at an interval of 12 seconds, 10 percent as falling at an interval of 10 seconds:

```
configure terminal
resource policy
policy system-global-pc1 global
system
memory io
critical rising 90 interval 12 falling 20 interval 10
major rising 70 interval 12 falling 15 interval 10
minor rising 60 interval 12 falling 10 interval 10
```

Configuring Per User Global Thresholding Policy for I/O Memory RO

The following example shows how to configure a per user global policy with the policy name as per-user-global-pc1 for I/O memory with critical threshold values of 90 percent as rising at an interval of 12 seconds, 20 percent as falling at an interval of 10 seconds, major threshold values of 70 percent as rising at an interval of 12 seconds, 15 percent as falling at an interval of 10 seconds, and minor threshold values of 60 percent as rising at an interval of 12 seconds, 10 percent as falling at an interval of 10 seconds:

```
configure terminal
resource policy
policy per-user-global-pc1 type iosprocess
system
memory io
critical rising 90 interval 12 falling 20 interval 10 global
major rising 70 interval 12 falling 15 interval 10 global
minor rising 60 interval 12 falling 10 interval 10 global
```

Configuring User Local Thresholding Policy for I/O Memory RO

The following example shows how to configure a user local policy with the policy name as user-local-pc1 for I/O memory with critical threshold values of 90 percent as rising at an interval of 12 seconds, 20 percent as falling at an interval of 10 seconds, major threshold values of 70 percent as rising at an interval of 12 seconds, 15 percent as falling at an interval of 10 seconds, and minor threshold values of 60 percent as rising at an interval of 12 seconds, 10 percent as falling at an interval of 10 seconds:

```
configure terminal
resource policy
policy user-local-pc1 type iosprocess
system
memory io
critical rising 90 interval 12 falling 20 interval 10
major rising 70 interval 12 falling 15 interval 10
minor rising 60 interval 12 falling 10 interval 10
```

Configuring System Global Thresholding Policy for Processor Memory RO

The following example shows how to configure a user system global policy with the policy name as system-global-pc1 for processor memory with critical threshold values of 90 percent as rising at an interval of 12 seconds, 20 percent as falling at an interval of 10 seconds, major threshold values of 70

percent as rising at an interval of 12 seconds, 15 percent as falling at an interval of 10 seconds, and minor threshold values of 60 percent as rising at an interval of 12 seconds, 10 percent as falling at an interval of 10 seconds:

```
configure terminal
resource policy
policy system-global-pc1 global
system
memory processor
critical rising 90 interval 12 falling 20 interval 10
major rising 70 interval 12 falling 15 interval 10
minor rising 60 interval 12 falling 10 interval 10
```

Configuring Per User Global Thresholding Policy for Processor Memory RO

The following example shows how to configure a per user global policy with the policy name as user-global-pc1 and the resource type as iosprocess for processor memory with critical threshold values of 90 percent as rising at an interval of 12 seconds, 20 percent as falling at an interval of 10 seconds, major threshold values of 70 percent as rising at an interval of 12 seconds, 15 percent as falling at an interval of 10 seconds, and minor threshold values of 60 percent as rising at an interval of 12 seconds, 10 percent as falling at an interval of 10 seconds:

```
configure terminal
resource policy
policy user-global-pc1 type iosprocess
system
memory processor
critical rising 90 interval 12 falling 20 interval 10
major rising 70 interval 12 falling 15 interval 10
minor rising 60 interval 12 falling 10 interval 10
```

Configuring User Local Thresholding Policy for Processor Memory RO

The following example shows how to configure a user local policy with the policy name as user-local-pc1 and the resource type as iosprocess for processor memory with critical threshold values of 90 percent as rising at an interval of 12 seconds, 20 percent as falling at an interval of 10 seconds, major threshold values of 70 percent as rising at an interval of 12 seconds, 15 percent as falling at an interval of 10 seconds, and minor threshold values of 60 percent as rising at an interval of 12 seconds, 10 percent as falling at an interval of 10 seconds:

```
configure terminal
resource policy
policy user-local-pc1 type iosprocess
system
memory processor
critical rising 90 interval 12 falling 20 interval 10
major rising 70 interval 12 falling 15 interval 10
minor rising 60 interval 12 falling 10 interval 10
```

Configuring System Global Thresholding Policy for Interrupt CPU RO

The following example shows how to configure a global policy with the policy name as system-global-pc1 for interrupt CPU with critical threshold values of 90 percent as rising at an interval of 12 seconds, 20 percent as falling at an interval of 10 seconds, major threshold values of 70 percent as rising at an interval of 12 seconds, 15 percent as falling at an interval of 10 seconds, and minor threshold values of 60 percent as rising at an interval of 12 seconds, 10 percent as falling at an interval of 10 seconds:

```
configure terminal
resource policy
policy system-global-pc1 global
```

```
system
cpu interrupt
critical rising 90 interval 12 falling 20 interval 10
major rising 70 interval 12 falling 15 interval 10
minor rising 60 interval 12 falling 10 interval 10
```

Configuring Per User Global Thresholding Policy for Interrupt CPU RO

The following example shows how to configure a per user global policy with the policy name as per-user-global-pc1 and the resource type as iosprocess for interrupt CPU with critical threshold values of 90 percent as rising at an interval of 12 seconds, 20 percent as falling at an interval of 10 seconds, major threshold values of 70 percent as rising at an interval of 12 seconds, 15 percent as falling at an interval of 10 seconds, and minor threshold values of 60 percent as rising at an interval of 12 seconds, 10 percent as falling at an interval of 10 seconds:

```
configure terminal
resource policy
policy per-user-global-pc1 type iosprocess
system
cpu interrupt
critical rising 90 interval 12 falling 20 interval 10 global
major rising 70 interval 12 falling 15 interval 10 global
minor rising 60 interval 12 falling 10 interval 10 global
```

Configuring User Local Thresholding Policy for Interrupt CPU RO

The following example shows how to configure a user local policy with the policy name as user-local-pc1 and the resource type as iosprocess for interrupt CPU with critical threshold values of 90 percent as rising at an interval of 12 seconds, 20 percent as falling at an interval of 10 seconds, major threshold values of 70 percent as rising at an interval of 12 seconds, 15 percent as falling at an interval of 10 seconds, and minor threshold values of 60 percent as rising at an interval of 12 seconds, 10 percent as falling at an interval of 10 seconds:

```
configure terminal
resource policy
policy user-local-pc1 global type iosprocess
system
cpu interrupt
critical rising 90 interval 12 falling 20 interval 10
major rising 70 interval 12 falling 15 interval 10
minor rising 60 interval 12 falling 10 interval 10
```

Configuring System Global Thresholding Policy for Process CPU RO

The following example shows how to configure a global policy with the policy name as system-global-pc1 for process CPU with critical threshold values of 90 percent as rising at an interval of 12 seconds, 20 percent as falling at an interval of 10 seconds, major threshold values of 70 percent as rising at an interval of 12 seconds, 15 percent as falling at an interval of 10 seconds, and minor threshold values of 60 percent as rising at an interval of 12 seconds, 10 percent as falling at an interval of 10 seconds:

```
configure terminal
resource policy
policy system-global-pc1 global
system
cpu process
critical rising 90 interval 12 falling 20 interval 10
major rising 70 interval 12 falling 15 interval 10
minor rising 60 interval 12 falling 10 interval 10
```

Configuring Per User Global Thresholding Policy for Process CPU RO

The following example shows how to configure a per user global policy with the policy name as per-user-global-pc1 and the resource type as iosprocess for process CPU with critical threshold values of 90 percent as rising at an interval of 12 seconds, 20 percent as falling at an interval of 10 seconds, major threshold values of 70 percent as rising at an interval of 12 seconds, 15 percent as falling at an interval of 10 seconds, and minor threshold values of 60 percent as rising at an interval of 12 seconds, 10 percent as falling at an interval of 10 seconds:

```
configure terminal
resource policy
resource policy per-user-global-pc1 type iosprocess
system
cpu process
critical rising 90 interval 12 falling 20 interval 10 global
major rising 70 interval 12 falling 15 interval 10 global
minor rising 60 interval 12 falling 10 interval 10 global
```

Configuring User Local Thresholding Policy for Process CPU RO

The following example shows how to configure a user local policy with the policy name as user-local-pc1 and the resource type as iosprocess for process CPU with critical threshold values of 90 percent as rising at an interval of 12 seconds, 20 percent as falling at an interval of 10 seconds, major threshold values of 70 percent as rising at an interval of 12 seconds, 15 percent as falling at an interval of 10 seconds, and minor threshold values of 60 percent as rising at an interval of 12 seconds, 10 percent as falling at an interval of 10 seconds:

```
configure terminal
resource policy
policy user-local-pc1 global type iosprocess
system
cpu process
critical rising 90 interval 12 falling 20 interval 10
major rising 70 interval 12 falling 15 interval 10
minor rising 60 interval 12 falling 10 interval 10
```

Configuring System Global Thresholding Policy for Total CPU RO

The following example shows how to configure a global policy with the policy name as system-global-pc1 for total CPU with critical threshold values of 90 percent as rising at an interval of 12 seconds, 20 percent as falling at an interval of 10 seconds, major threshold values of 70 percent as rising at an interval of 12 seconds, 20 percent as falling at an interval of 10 seconds, and minor threshold values of 60 percent as rising at an interval of 12 seconds, 10 percent as falling at an interval of 10 seconds:

```
configure terminal
resource policy
policy system-global-pc1 global
system
cpu total
critical rising 90 interval 12 falling 20 interval 10
major rising 70 interval 12 falling 15 interval 10
minor rising 60 interval 12 falling 10 interval 10
```

Configuring Per User Global Thresholding Policy for Total CPU RO

The following example shows how to configure a per user global policy with the policy name as per-user-global-pc1 and the resource type as iosprocess for total CPU with critical threshold values of 90 percent as rising at an interval of 12 seconds, 20 percent as falling at an interval of 10 seconds, major

threshold values of 70 percent as rising at an interval of 12 seconds, 15 percent as falling at an interval of 10 seconds, and minor threshold values of 60 percent as rising at an interval of 12 seconds, 10 percent as falling at an interval of 10 seconds:

```
configure terminal
resource policy
policy per-user-global-pc1 type iosprocess
system
cpu total
critical rising 90 interval 12 falling 20 interval 10 global
major rising 70 interval 12 falling 15 interval 10 global
minor rising 60 interval 12 falling 10 interval 10 global
```

Configuring User Local Thresholding Policy for Total CPU RO

The following example shows how to configure a user local policy with the policy name as user-local-pc1 and the resource type as iosprocess for total CPU with critical threshold values of 90 percent as rising at an interval of 12 seconds, 20 percent as falling at an interval of 10 seconds, major threshold values of 70 percent as rising at an interval of 12 seconds, 15 percent as falling at an interval of 10 seconds, and minor threshold values of 60 percent as rising at an interval of 12 seconds, 10 percent as falling at an interval of 10 seconds:

```
configure terminal
resource policy
policy user-local-pc1 type iosprocess
system
cpu total
critical rising 90 interval 12 falling 20 interval 10
major rising 70 interval 12 falling 15 interval 10
minor rising 60 interval 12 falling 10 interval 10
```

Applying a Policy: Example

The following example shows how to apply a per user thresholding policy for the resource instance EXEC, resource user type iosprocess, and policy name policy-test1:

```
configure terminal
resource policy
policy policy-test1 type iosprocess
exit
user EXEC iosprocess policy-test1
```

The following example shows how to apply a global thresholding policy with the policy name global-global-test1:

```
configure terminal
resource policy
policy global-global-test1 global
exit
user global global-global-test1
```

The following example shows how to apply a group thresholding policy with the group name gr1 and resource type as iosprocess:

```
configure terminal
resource policy
policy group-test1
exit

user group gr1 type iosprocess
```

```
instance http
policy group-test1
```

Setting a System Global Thresholding Policy for I/O Memory: Example

The following example shows the configuration of a global memory thresholding policy for I/O memory. In this example, the policy is given the name “system-global-io”, and the threshold for critical I/O memory usage is defined as being usage of over 90 percent of the globally available I/O memory pool for 12 consecutive seconds.

The critical falling threshold is also defined in this example (less than 20 percent of the globally available I/O memory pool for 10 seconds or more); however, only the critical rising level will affect when the automatic deallocation procedure is triggered.

```
configure terminal
resource policy
policy system-global-io global
system
memory io
critical rising 90 interval 12 falling 20 interval 10
```

Additional References

The following sections provide references related to the Embedded Resource Manager.

Related Documents

Related Topic	Document Title
Cisco IOS commands	*Cisco IOS Master Commands List, All Releases*
Configuration fundamentals commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*Cisco IOS Configuration Fundamentals Command Reference*
Network management commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*Cisco IOS Network Management Command Reference*
Embedded Event Manager configuration tasks	*Cisco IOS Embedded Event Manager*
Memory Leak Detector	*Memory Leak Dectector*

Standards

Standards	Title
No new or modified standards are supported by this feature.	—

MIBs

MIBs	MIBs Link
CISCO-ERM-MIB.my	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFCs	Title
No new or modified RFCs are supported by this feature.	—

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for Embedded Resource Manager

Table 1 lists the features in this module and provides links to specific configuration information. Only features that were introduced or modified in Cisco IOS Release 12.3(14)T or a later release appear in the table.

Not all commands may be available in your Cisco IOS software release. For release information about a specific command, see the command reference documentation.

Use Cisco Feature Navigator to find information about platform support and software image support. Cisco Feature Navigator enables you to determine which Cisco IOS and Catalyst OS software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Note Table 1 lists only the Cisco IOS software release that introduced support for a given feature in a given Cisco IOS software release train. Unless noted otherwise, subsequent releases of that Cisco IOS software release train also support that feature.

Table 1 Feature Information for Embedded Resource Manager

Feature Name	Releases	Feature Information
Embedded Resource Manager	12.3(14)T 12.2(33)SRB 12.2(33)SB	The Embedded Resource Manager (ERM) feature allows you to monitor internal system resource utilization for finite resources such as the buffer, memory, and CPU. ERM monitors resource utilization from the perspective of various subsystems within the Cisco IOS software such as resource owners (ROs) and resource users (RUs). ERM allows you to configure threshold values for system resources, leading to better insight into system scalability and improved system availability. The following sections provide information about this feature: • Prerequisites for Embedded Resource Manager, page 2 • How to Configure Embedded Resource Manager, page 8

Table 1 Feature Information for Embedded Resource Manager (continued)

Feature Name	Releases	Feature Information
Embedded Resource Manager MIB	15.0(1)M 12.2(33)SRB 12.2(33)SB	The ERM MIB feature introduces MIB support for the Embedded Resource Manager (ERM) feature. The ERM feature tracks resource usage information for every registered resource owner and resource user. ERM ensures efficient usage of available resources.The ERM MIB feature allows you to monitor the usage of resources by gathering resource usage information using MIB objects. The network manager can use the information collected by the ERM MIB objects to ensure the optimal use of the resources. The following command was introduced by this feature: **snmp-server enable traps resource-policy**.
Packet Memory Reclamation	12.4(6)T 12.2(33)SRE	The Packet Memory Reclamation functionality utilizes the ERM infrastructure to cleanup and reclaim leaked Cisco IOS packet memory using the Memory Leak Detector process (sometimes referred to as the “Garbage Detection” or “GD” process). The following section provides information about this feature: • Memory Resource Owner, page 5

Glossary

CPUHOG—Each process is allocated a quantum of time, which is equivalent to 200 ms. If a process is running for more than 2 seconds, the process is hogging the CPU. This condition is called CPUHOG.

RM—resource usage monitors. Applications that wants to monitor resource utilization of resources by the resource users.

RO—resource owners. Provides resources to the resource users. For example, CPU, buffer, memory and so on.

RU—resource users. Applications or clients (like HTTP, SNMP, telnet, and so on) that use the resources and receive notifications to throttle when thresholds exceed the current values.

Configuring Embedded Resource Manager-MIB

First Published: February 19, 2007
Last Updated: November 20, 2009

The Embedded Resource Manager (ERM)-MIB feature introduces MIB support for the ERM feature. The ERM feature tracks resource usage information for every registered resource owner and resource user. The ERM-MIB feature allows you to monitor the usage of resources by gathering resource usage information using MIB objects. The network manager can use the information collected by the ERM-MIB objects to ensure the optimal use of the resources.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the "Feature Information for ERM-MIB" section on page 15.

Use Cisco Feature Navigator to find information about platform support and Cisco IOS and Catalyst OS software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Contents

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

Prerequisites for ERM-MIB

Simple Network Management Protocol (SNMP) must be enabled on the router before notifications (traps) can be configured or before SNMP GET operations can be performed.

Information About ERM-MIB

The ERM-MIB feature introduces network management support for ERM through the use of ERM-MIB table entries, MIB objects, and MIB trap notification objects that are defined in CISCO-ERM-MIB.my.

To use the ERM-MIB feature, you should understand the following concepts:

- ERM Show MIB Objects, page 2
- ERM Configuration MIB Objects, page 7
- ERM Notification MIB Objects, page 9

ERM Show MIB Objects

The ERM Show MIB objects are read-only objects. You can use these MIB objects to obtain information about resource owners, resource user type, resource users or groups, resource owner and resource user relationships, and resource monitors in the system.

Table 1 describes the ERM Show MIB objects.

Table 1 ERM Show MIB objects

ERM Show MIB Objects	Purpose
cermResOwnerTable	Obtains the details of all resource owners in the system.
cermResOwnerSubTypeTable	Obtains the details of the resource owner sub-types in the system.
cermResOwnerSubTypeThresholdTable	Obtains the details of the threshold value defined for each resource owner sub-type in the system.
cermResUserTypeTable	Obtains the details of the resource user types in the system.
cermResUserTable	Obtains the details of each resource user in the system.
cermResGroupTable	Obtains the details of each resource group in the system.
cermResGroupResUserTable	Obtains the details of resource users available in a specific resource group.
cermResOwnerResUserOrGroupTable	Obtains the details of all the resource owners, resource users, and group relationships defined in the system.
cermResOwnerResUserOrGroupThresholdTable	Obtains the details of the threshold value defined for each resource owner sub-type, resource user or resource group relationship in the system.

Table 1 ERM Show MIB objects

ERM Show MIB Objects	Purpose
cermResUserTypeResOwnerTable	Obtains the details of resource owners present in a specific resource user type.
cermResMonitorTable	Obtains the details of resource monitors in the system.
cermResMonitorResOwnerResUserTable	Obtains the details of resource owners, resource users, and resource owner and resource user relationships that are monitored by a resource monitor.
cermResMonitorPolicyTable	Obtains the details of resource policies that are monitored by a resource monitor.

Obtaining Information About Resource Owners

You can use cermResOwnerTable to obtain information about all resource owners in the system. The index entries for cermResOwnerTable are entPhysicalIndex, cermResOwnerSubEntityId, and cermResOwnerId.

The cermResOwnerTable defines the following MIB objects:

- cermResOwnerSubEntityId
- cermResOwnerId
- cermResOwnerName
- cermResOwnerMeasurementUnit
- cermResOwnerThresholdIsConfigurable
- cermResOwnerResUserCount
- cermResOwnerResGroupCount

Obtaining Sub-type Specific Information

You can use cermResOwnerSubTypeTable to obtain sub-type specific information. The cermResOwnerSubTypeTable is an extension of the cermResOwnerTable. The index entries for cermResOwnerSubTypeTable are entPhysicalIndex, cermResOwnerSubEntityId, cermResOwnerId, and cermResOwnerSubTypeId.

Each resource owner will have one or more entries in this table. For example, the CPU resource owner has three sub-types: process, interrupt, and total.

Some resource owners may not have any sub-types, such as the IPC resource owner. In such cases this table will contain a single entry with cermResOwnerSubTypeId as 0 and cermResOwnerSubTypeName as an empty string.

You can obtain all sub-type related information specified in this table by querying the corresponding resource owner.

The cermResOwnerSubTypeTable defines the following objects:

- cermResOwnerSubTypeId
- cermResOwnerSubTypeName

- cermResOwnerSubTypeUsagePct
- cermResOwnerSubTypeUsage
- cermResOwnerSubTypeMaxUsage
- cermResOwnerSubTypeGlobNotifSeverity

Obtaining Applied System Global Threshold Details

You can use cermResOwnerSubTypeThresholdTable to obtain applied threshold details for each resource owner sub-type. This object is an extension of the cermResOwnerSubTypeTable.

The index entries for cermResOwnerSubTypeThresholdTable are entPhysicalIndex, cermResOwnerSubEntityId, cermResOwnerId, cermResOwnerSubTypeId, and cermResOwnerSubTypeThreshSeverity. You can obtain all threshold details corresponding to a resource owner sub-type by querying the corresponding resource owner.

The cermResOwnerSubTypeThresholdTable defines the following objects:

- cermResOwnerSubTypeThreshSeverity
- cermResOwnerSubTypeRisingThresh
- cermResOwnerSubTypeRisingInterval
- cermResOwnerSubTypeFallingThresh
- cermResOwnerSubTypeFallingInterval

Obtaining Information About a Resource User Type

You can use cermResUserTypeTable to obtain information about a resource user type. Each resource user type in the system has an entry in cermResUserTypeTable. The index entries for this object are entPhysicalIndex, cermResUserTypeSubEntityId, and cermResUserTypeId.

The cermResUserTypeTable defines the following objects:

- cermResUserTypeSubEntityId
- cermResUserTypeId
- cermResUserTypeName
- cermResUserTypeResOwnerCount
- cermResUserTypeResUserCount
- cermResUserTypeResGroupCount

Obtaining Resource User-Specific Information

You can use cermResUserTable to obtain information about each resource user in the system. This object is an extension of cermResUserTypeTable. The index entries for cermResUserTable are entPhysicalIndex, cermResUserTypeSubEntityId, cermResUserTypeId, and ermResUserId.

The cermResUserTable defines the following objects:

- cermResUserId
- cermResUserName
- cermResUserPriority

- cermResUserResGroupId

Obtaining Information About Resource Groups

You can use cermResGroupTable to obtain information about every resource group available in the system. This object is an extension of cermResUserTypeTable. The index entries for cermResGroupTable are entPhysicalIndex, cermResUserTypeSubEntityId, cermResUserTypeId, and cermResGroupId.

The cermResGroupTable defines the following objects:

- cermResGroupId
- cermResGroupName
- cermResGroupUserInstanceCount

Obtaining Information About Resource Users in a Particular Resource Group

You can use cermResGroupResUserTable to obtain the list of resource users available in a particular resource group. This object is an extension of cermResGroupTable. The index entries for cermResGroupResUserTable are entPhysicalIndex, cermResUserTypeSubEntityId, cermResUserTypeId, cermResGroupId, and cermResGroupResUserId.

The cermResGroupResUserTable defines the following object:

- cermResGroupResUserId

Obtaining Information About Resource Owner and User Relationships

You can use cermResOwnerResUserOrGroupTable to obtain information about each resource owner-user relationship or resource owner-group relationship in the system. This object is an extension of cermResOwnerSubTypeTable.

The index entries for cermResOwnerResUserOrGroupTable are entPhysicalIndex, cermResOwnerSubEntityId, cermResOwnerId, cermResOwnerSubTypeId, cermResOwnerResUserTypeId, and cermResOwnerResUserOrGroupId.

This table can be used for the following tasks:

- To obtain the list of resource users registered for a specific resource owner.
- To obtain usage, max-usage, user local and per user global current notification levels for a given resource owner sub-type and resource user relation.

The cermResOwnerResUserOrGroupTable defines the following objects:

- cermResOwnerResUserTypeId
- cermResOwnerResUserOrGroupId
- cermResUserOrGroupFlag
- cermResUserOrGroupUsagePct
- cermResUserOrGroupUsage
- cermResUserOrGroupMaxUsage
- cermResUserOrGroupNotifSeverity
- cermResUserOrGroupGlobNotifSeverity

Obtaining Threshold Information About Each Resource Owner Sub-type and Resource User Relationship

You can use cermResOwnerResUserOrGroupThresholdTable to obtain threshold information about each resource owner sub-type and resource user relationship. This object is an extension of the cermResOwnerResUserOrGroupTable.

The index entries for cermResOwnerResUserOrGroupThresholdTable are entPhysicalIndex, cermResOwnerSubEntityId, cermResOwnerId, cermResOwnerSubTypeId, cermResOwnerResUserTypeId, cermResOwnerResUserOrGroupId, cermResUserOrGroupThreshIsUserGlob, and cermResUserOrGroupThreshSeverity.

The cermResOwnerResUserOrGroupThresholdTable defines the following objects:

- cermResUserOrGroupThreshIsUserGlob
- cermResUserOrGroupThreshSeverity
- cermResUserOrGroupThreshFlag
- cermResUserOrGroupRisingThresh
- cermResUserOrGroupRisingInterval
- cermResUserOrGroupFallingThresh
- cermResUserOrGroupFallingInterval

Obtaining Information About Resource Owners Present in a Resource User Type

You can use cermResUserTypeResOwnerTable to obtain the list of resource owners present in a resource user type. This object is an extension of the cermResUserTypeTable.

The index entries for cermResUserTypeResOwnerTable are entPhysicalIndex, cermResUserTypeSubEntityId, cermResUserTypeId, and cermResUserTypeResOwnerId.

The cermResUserTypeResOwnerTable defines the following objects:

- cermResUserTypeResOwnerId

Obtaining Information About Resource Monitors

You can use cermResMonitorTable to obtain the list of resource monitors in the system. The index entries for this object are entPhysicalIndex, cermResMonitorSubEntityId, and cermResMonitorId.

The cermResMonitorTable defines the following objects:

- cermResMonitorSubEntityId
- cermResMonitorId
- cermResMonitorName

Obtaining Resource Information About Resource Owner and User Relationships that are Monitored

You can use cermResMonitorResOwnerResUserTable to obtain resource-related information that is tracked by a resource monitor. This object is an extension of cermResMonitorTable.

The index entries for cermResMonitorResOwnerResUserTable are entPhysicalIndex, cermResMonitorSubEntityId, cermResMonitorId, cermResMonitorResOwnerId, cermResMonitorResUserTypeId, and cermResMonitorResUserId.

The cermResMonitorResOwnerResUserTable defines the following objects:

- cermResMonitorResOwnerId
- cermResMonitorResUserTypeId
- cermResMonitorResUserId
- cermResMonitorResPolicyName

Obtaining Information About Resource Policies that are Monitored by a Resource Monitor

You can use cermResMonitorPolicyTable to obtain the list of resource policies that are tracked by a resource monitor. This object is an extension of the cermResMonitorTable. The index entries for cermResMonitorPolicyTable are entPhysicalIndex, cermResMonitorSubEntityId, cermResMonitorId, and cermResMonitorPolicyName.

The cermResMonitorPolicyTable defines the following object:

- cermResMonitorPolicyName

ERM Configuration MIB Objects

You can use the ERM Configuration MIB objects to perform the following tasks:

- Creating, Modifying, or Deleting a Resource Policy, page 8
- Configuring Threshold Values and Intervals for Resource Owner Sub-types in a Resource Policy, page 8
- Creating or Deleting a Resource Group, page 9
- Creating or Deleting a User Instance in a Resource Group, page 9
- Applying an Existing Resource Policy to a Resource User or Group, page 9

Table 2 describes the ERM Configuration MIB objects.

Table 2 ERM Configuration MIB Objects

ERM Configuration MIB Objects	Purpose
cermScalarsGlobalPolicyName (scalar object)	Identifies and indicates the global resource policy applied in the system.
cermConfigPolicyTable	Creates, modifies, or deletes a resource policy.
cermConfigPolicyResOwnerThreshTable	Configures threshold values and intervals for resource owner sub-types.
cermConfigResGroupTable	Creates or deletes a resource group.
cermConfigResGroupUserTable	Creates or deletes a user instance in a resource group.
cermConfigPolicyApplyTable	Applies an existing resource policy to a resource user or group.

Verifying Whether a Global Resource Policy is Applied in the System

You can use the scalar object cermScalarsGlobalPolicyName to impose and indicate if a global resource policy is applied in the system. If no global resource policy is applied in the system, then this object will contain empty string. This object has read-write access permission. Setting this scalar object to an existing global resource policy name, will result in applying the global resource policy to the system.

Creating, Modifying, or Deleting a Resource Policy

You can use cermConfigPolicyTable to create, modify, or delete a resource policy. The index entry for this object is cermPolicyName.

The cermConfigPolicyTable defines the following objects:

- cermPolicyName
- cermPolicyIsGlobal
- cermPolicyUserTypeName
- cermPolicyLoggingEnabled
- cermPolicySnmpNotifEnabled
- cermPolicyStorageType
- cermPolicyRowStatus

Configuring Threshold Values and Intervals for Resource Owner Sub-types in a Resource Policy

You can use cermConfigPolicyResOwnerThreshTable to configure rising or falling threshold values and rising or falling intervals for resource owner sub-types in a resource policy. This object is an extension of the cermConfigPolicyTable.

The index entries for cermConfigPolicyResOwnerThreshTable are cermPolicyName, cermPolicyPhysicalIndex, cermConfigPolicyResOwnerSubEntityId, cermConfigPolicyResOwnerId, cermConfigPolicyResOwnerSubTypeId, ermConfigPolicyIsUserGlobal, and cermConfigPolicyThresholdLevel.

The cermConfigPolicyResOwnerThreshTable defines the following objects:

- cermPolicyPhysicalIndex
- cermConfigPolicyResOwnerSubEntityId
- cermPolicyResOwnerId
- cermPolicyResOwnerSubTypeId
- cermPolicyIsUserGlobal
- cermPolicyThresholdLevel
- cermPolicyRisingThreshold
- cermPolicyRisingInterval
- cermPolicyFallingThreshold
- cermPolicyFallingInterval
- cermPolicyResOwnerThreshStorageType
- cermPolicyResOwnerRowStatus

Creating or Deleting a Resource Group

You can use cermConfigResGroupTable to create or delete a resource group in the system. The index entry for this object is cermConfigResGroupName.

The cermConfigResGroupTable defines the following objects:

- cermConfigResGroupName
- cermConfigResGroupUserTypeName
- cermConfigResGroupStorageType
- cermConfigResGroupRowStatus

Creating or Deleting a User Instance in a Resource Group

You can use cermConfigResGroupUserTable to create or delete a user instance in a given resource group. This object is an extension of the cermConfigResGroupTable.

The index entries for cermConfigResGroupUserTable are cermConfigResGroupName and cermConfigResGroupUserName.

The cermConfigResGroupUserTable defines the following objects:

- cermConfigResGroupUserName
- cermConfigResGroupUserStorageType
- cermConfigResGroupUserRowStatus

Applying an Existing Resource Policy to a Resource User or Group

You can use cermConfigPolicyApplyTable to apply an existing resource policy to a resource user or resource group. The index entries for this object are cermPolicyApplyUserOrGroupName and cermPolicyApplyUserOrGroupFlag.

The cermConfigPolicyApplyTable defines the following objects:

- cermPolicyApplyUserOrGroupName
- cermPolicyApplyUserOrGroupFlag
- cermPolicyApplyPolicyName
- cermPolicyApplyStorageType
- cermPolicyApplyRowStatus

ERM Notification MIB Objects

You can configure ERM Notification MIB objects to receive global or user-specific notification on policy violation. There are three types of ERM Notification MIB objects.

Table 3 describes the ERM Notification MIB objects.

Table 3 ERM Notification MIB Objects

ERM Notification MIB Objects	Purpose
cermNotifsEnabled	Enables ERM notifications.
ciscoErmGlobalPolicyViolation	Specifies the type of notification received on global policy violation.
ciscoErmLocalPolicyViolation	Specifies the type of user-specific notification received on local policy violation.

Controlling the Generation of Traps for ERM Policy Violation Notifications

You can use cermNotifsEnabled to determine if the generation of traps for ERM policy violation notifications is allowed.

When this object is set to true, it allows generation of traps for the ERM policy violation related notifications ciscoErmGlobalPolicyViolation and ciscoErmLocalPolicyViolation.

Receiving a Global Notification on Policy Violation

You can use ciscoErmGlobPolicyViolation to receive global notification on policy violation.

The notification object ciscoErmGlobPolicyViolation defines the following objects:

- cermResOwnerName
- cermResOwnerSubTypeName
- cermNotifsThresholdSeverity
- cermNotifsThresholdValue
- cermNotifsDirection
- cermNotifsPolicyName

Receiving a User-Specific Notification on Policy Violation

You can use ciscoErmUserPolicyViolation to receive a user-specific notification on policy violation.

The notification object ciscoErmUserPolicyViolation contains the following objects:

- cermResOwnerName
- cermResOwnerSubTypeName
- cermResUserTypeName
- cermResUserName
- cermResUserOrGroupThreshFlag
- cermNotifsThresholdIsUserGlob
- cermNotifsThresholdSeverity
- cermNotifsThresholdValue
- cermNotifsDirection
- cermNotifsPolicyName

How to Configure ERM-MIB

This section contains the following procedures:

- Enabling ERM-MIB Notification Traps, page 11 (required)
- Configuring the Router to Send SNMP Notification Traps for ERM to a Host, page 12 (required)

Enabling ERM-MIB Notification Traps

You can enable ERM-MIB notification traps, which are generated when resource usage exceeds the threshold value. The ERM-MIB notification traps will be sent to the host that is configured to receive traps.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp-server enable traps resource-policy**
4. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**snmp-server enable traps resource-policy** **Example:** `Router(config)# snmp-server enable traps resource-policy`	Enables CISCO-ERM-MIB notifications.
Step 4	**end** **Example:** `Router(config)# end`	Returns the router to privileged EXEC mode.

Configuring the Router to Send SNMP Notification Traps for ERM to a Host

Perform this task to enable the router to send SNMP notifications traps defined in ERM-MIB to a host.

Prerequisites

- SNMP must be enabled on your network.
- Create an SNMP server community to receive information on MIB objects and traps using the **snmp-server community** command.

SUMMARY STEPS

1. **enable**
2. **show running-config** [*options*]
3. **configure terminal**
4. **snmp-server host** {*hostname* | *ip-address*} [**vrf** *vrf-name*] [**traps** | **informs**] [**version** {**1** | **2c** | **3** [**auth** | **noauth** | **priv**]}] *community-string* [**udp-port** *port*] [*notification-type*]
5. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show running-config** [*options*] **Example:** `Router# show running-config`	Displays the running configuration to determine if an SNMP agent is already running. • If no SNMP information is displayed, continue with the next step. If any SNMP information is displayed, you can modify the information or change it as needed.
Step 3	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 4	**snmp-server host** {*hostname* | *ip-address*} [**vrf** *vrf-name*] [**traps** | **informs**] [**version** {**1** | **2c** | **3** [**auth** | **noauth** | **priv**]}] *community-string* [**udp-port** *port*] [*notification-type*] **Example:** `Router(config)# snmp-server host 209.165.201.30 traps version 2c priv mycommunitystring isis`	Specifies the recipient (target host) for ERM SNMP notification operations.
Step 5	**end** **Example:** `Router(config)# end`	Returns the router to privileged EXEC mode.

Configuration Examples for ERM-MIB

This section provides the following configuration example:

- Configuring the Router to Send SNMP Notifications for ERM to a Host: Example, page 13

Configuring the Router to Send SNMP Notifications for ERM to a Host: Example

The following example shows how to configure the router to send SNMP notifications for ERM to a host:

```
Router# configure terminal
Router(config)# snmp-server community public rw
Router(config)# snmp-server enable traps resource-policy
Router(config)# snmp-server host 209.165.201.30 version 2c public
Router(config)# end
```

Additional References

The following sections provide references related to the ERM-MIB feature.

Related Documents

Related Topic	Document Title
Cisco IOS commands	*Cisco IOS Master Commands List, All Releases*
Embedded Resource Manager	*Embedded Resource Manager*
Network Management commands: complete command syntax, command mode, command history, defaults, usage guidelines, and examples	*Cisco IOS Network Management Command Reference*

Standards

Standard	Title
None	—

MIBs

MIB	MIBs Link
• CISCO-ERM-MIB.my	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
RFC 1902	*Structure of Management Information for Version 2 of the Simple Network Management Protocol (SNMPv2)*

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for ERM-MIB

Table 4 lists the features in this module and provides links to specific configuration information. Only features that were introduced or modified in Cisco IOS Release 12.2(33)SRB and Cisco IOS Release 15.0(1)M or later releases appear in the table.

Not all commands may be available in your Cisco IOS software release. For release information about a specific command, see the command reference documentation.

Use Cisco Feature Navigator to find information about platform support and software image support. Cisco Feature Navigator enables you to determine which Cisco IOS and Catalyst OS software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Note Table 4 lists only the Cisco IOS software release that introduced support for a given feature in a given Cisco IOS software release train. Unless noted otherwise, subsequent releases of that Cisco IOS software release train also support that feature.

Table 4 Feature Information for ERM-MIB

Feature Name	Releases	Feature Information
Embedded Resource Manager (ERM)-MIB	12.2(33)SB 12.2(33)SRB 12.4(15)T	The ERM-MIB feature introduces MIB support for the Embedded Resource Manager (ERM) feature. The ERM-MIB feature allows you to monitor the usage of resources by gathering resource usage information using MIB objects. The network manager can use the information collected by the ERM-MIB objects to ensure the optimal use of the resources. The following commands were introduced or modified by this feature: **snmp-server enable traps resource-policy**

Embedded Syslog Manager (ESM)

Embedded Syslog Manager (ESM)

First Published: July 28, 2003
Last Updated: November 14, 2008

The Embedded Syslog Manager (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.

Finding Feature Information

Your software release may not support all of the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the "Feature Information for Embedded Syslog Manager" section on page 27.

Use Cisco Feature Navigator to find information about platform support and Cisco IOS and Catalyst OS software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Contents

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

Restrictions for Embedded Syslog Manager

Embedded Syslog Manager (ESM) depends on the Tcl 8.3.4 Cisco IOS subsystem, as ESM filters are written in Tool Command Language (Tcl). ESM is only available in images that support Tcl version 8.3.4 or later. Support for Tcl 8.3.4 is introduced in Cisco IOS Release 12.3(2)T.

ESM filters are written in Tcl. This document assumes the reader is familiar with Tcl programming.

ESM filtering cannot be applied to SNMP "history" logging. In other words, ESM filtering will not be applied to messages logged using the **logging history** and **snmp-server enable traps syslog** commands.

Currently, the ESM filters do not support the debug messages. For example, if debug messages for IP packets are enabled (with the **debug ip packet** command) and an ESM filter is used on the debug messages, the filter will not work.

Information About the Embedded Syslog Manager

To configure the Embedded Syslog Manager, you should understand the following concepts:

Cisco IOS System Message Logging

The Cisco IOS system message logging (syslog) process allows the system to report and save important error and notification messages, either locally or to a remote logging server. These syslog messages include messages in a standardized format (called system logging messages, system error messages, or simply system messages). These messages are generated during network operation to assist users and Cisco TAC engineers with identifying the type and severity of a problem, or to aid users in monitoring router activity. System logging messages can be sent to console connections, monitor (TTY) connections, the system buffer, or to remote hosts.

With the introduction of the Embedded Syslog Manager, system messages can be logged independently as standard messages, XML-formatted messages, or ESM filtered messages. These outputs can be sent to any of the traditional syslog targets. For example, you could enable standard logging to the console connection, XML-formatted message logging to the buffer, and ESM filtered message logging to the monitor. Similarly, each type of output could be sent to different remote hosts. A benefit of separate logging processes is that if, for example, there is some problem with the ESM filter modules, standard logging will not be affected.

System Logging Message Formatting

System logging messages are displayed in the following format:

%<facility>-<severity>-<mnemonic>: <message-text>

For example:

```
%LINK-5-CHANGED: Interface Serial3/3, changed state to administratively down
```

Usually, these messages are preceded by additional text, such as the timestamp and error sequence number:

<sequence-number>: <timestamp>:%<facility>-<severity>-<mnemonic>: <message-text>

For example:

```
000013: Mar 18 14:52:10.039:%LINK-5-CHANGED: Interface Serial3/3, changed state
to administratively down
```

Note The timestamp format used in system logging messages is determined by the **service timestamps** global configuration mode command. The **service sequence-numbers** global configuration command enables or disables the leading sequence number. An asterisk (*) before the time indicates that the time may be incorrect because the system clock has not synchronized to a reliable time source.

Embedded Syslog Manager

The Embedded Syslog Manager (ESM) is a feature integrated in Cisco IOS software that allows complete control over system message logging at the source. ESM provides a programmatic interface to allow you to write custom filters that meet your specific needs in dealing with system logging. Benefits of this feature include:

- Customization—Fully customizable processing of system logging messages, with support for multiple, interfacing syslog collectors.
- Severity escalation for key messages—The ability to configure your own severity levels for syslog messages instead of using the system-defined severity levels.
- Specific message targeting—The ability to route specific messages or message types, based on type of facility or type of severity, to different syslog collectors.
- SMTP-base e-mail alerts—Capability for notifications using TCP to external servers, such as TCP-based syslog collectors or Simple Mail Transfer Protocol (SMTP) servers.
- Message Limiting—The ability to limit and manage syslog “message storms” by correlating device-level events.

The ESM is not a replacement for the current UDP-based syslog mechanism; instead, it is an optional subsystem that can operate in parallel with the current system logging process. For example, you can continue to have the original syslog message stream collected by server A, while the filtered, correlated, or otherwise customized ESM logging stream is sent to server B. All of the current targets for syslog messages (console, monitor, buffer, and syslog host list) can be configured to receive either the original syslog stream or the ESM stream. The ESM stream can be further divided into user-defined streams and routed to collectors accordingly.

Syslog Filter Modules

To process system logging messages, the ESM uses syslog filter modules. Syslog filter modules are merely scripts written in the Tcl script language stored in local system memory or on a remote file server. The ESM is customizable because you can write and reference your own scripts.

Syslog filter modules can be written and stored as plain-text files or as precompiled files. Tcl script pre-compiling can be done with tools such as TclPro. Precompiled scripts allow a measure of security and managed consistency because they cannot be edited.

Note As Tcl script modules contain executable commands, you should manage the security of these files in the same way you manage configuration files.

How to Use the Embedded Syslog Manager

This section contains the following procedures:

- Writing ESM Syslog Filter Modules, page 4
- Configuring the Embedded Syslog Manager, page 10

Writing ESM Syslog Filter Modules

Before referencing syslog filter modules in the ESM configuration, you must write or obtain the modules you wish to apply to system logging messages. Syslog filter modules can be stored in local system memory, or on a remote file server. To write syslog filter modules, you should understand the following concepts:

- The ESM Filter Process, page 4
- Syslog Filter Module Input, page 5
- Normal ESM Filter Processing, page 7
- Background ESM Filter Processing, page 9

The ESM Filter Process

When ESM is enabled, all system logging messages are processed through the referenced syslog filter modules. Syslog filter modules are processed in their order in the filter chain. The position of a syslog filter module in the filter chain is determined by the position tag applied in the **logging filter** global configuration mode command. If a position is not specified, the modules are processed in the order in which they were added to the configuration.

The output of each filter module is used as the input for the next filter module in the chain. In other words, the Tcl global variable containing the original syslog message (::orig_msg) is set to the return value of each filter before calling the next filter in the chain. Thus, if a filter returns NULL, no message will be sent out to the ESM stream. Once all filters have processed the message, the message is enqueued for distribution by the logger.

The console, buffer, monitor, and syslog hosts can be configured to receive a particular message stream (normal, XML, or ESM). The syslog hosts can be further restricted to receive user-defined numbered streams. Each target examines each message and accepts or rejects the message based on its stream tag. ESM filters can change the destination stream by altering the messages' stream tag by changing the Tcl global variable "::stream."

Syslog Filter Module Input

When ESM is enabled, system logging messages are sent to the logging process. Each of the data elements in the system logging message, as well as the formatted syslog message as a whole, are recorded as Tcl global variables. The data elements format for the syslog message are as follows:

<sequence-number>: <timestamp>:%<facility>-<severity>-<mnemonic>: <message-text>

The message-text will often contain message-arguments.

When messages are received on a syslog host a "syslog-count" number is also added:

<syslog-count>: <sequence-number>: <timestamp>:%<facility>-<severity>-<mnemonic>: <message-text>

For example:

```
24:000024:02:18:37:%SYS-5-CONFIG_I:Configured from console by console
```

Table 1 lists the Tcl script input variables used in syslog filter modules. The syslog message data that the filter must operate on are passed as Tcl global namespace variables. Therefore, variables should be prefixed by a double-colon within the script module.

Table 1 Valid Variables for Syslog Filter Modules

Variable Name	Definition
::orig_msg	Full original system logging message as formatted by the system. • If the filter module is just making decisions on whether to send a message or not, return either NULL or the value of this variable ($::orig_msg).
::hostname	The router's hostname. • The hostname can be added to the beginning of syslog messages sent to remote hosts using the **logging origin-id hostname** global configuration mode command.
::buginfseq	The error message sequence number. • The **service sequence-numbers** global configuration command enables or disables the leading sequence number.
::timestamp	The timestamp on the system logging message. • The timestamp format used in system logging messages is determined by the **service timestamps** global configuration mode command.
::facility	The name of the system facility that generated the message. • The FACILITY is a code consisting of two or more uppercase letters that indicate the facility to which the message refers. A facility can be a hardware device, a protocol, or a module of the system software. Common examples include SYS, LINK, LINEPROTO, and so on.
::severity	The severity value. • The SEVERITY is a single-digit code from 0 to 7 that reflects the severity of the condition. The lower the number, the more serious the message. • The syslog filter module should change this variable if the severity is to be escalated.

Table 1 Valid Variables for Syslog Filter Modules

Variable Name	Definition
::mnemonic	The message mnemonic. • The MNEMONIC is a code (usually an abbreviated description) that uniquely identifies the type of error or event. Common examples include CONFIG_I, UPDOWN, and so on.
::format_string	The message-text string. • The format string is used to create the original message. The message text will often contain arguments; for example, in the message "Configured from %s by %s," %s indicates the message arguments. • The message-text string is the message form that can be passed to the Tcl **format** command.
::msg_args	The message-text arguments. • The msg_args variable is the list containing the arguments for the format_string. • For example, in the system logging message "2w0d: %SYS-5-CONFIG_I: Configured from console by console." the format_string is "Configured from %s by %s." and the msg_args are "console, console."
::process	The process name and interrupt level string. • Some system messages describe internal errors and contain trace back information. The following sample output shows the format for process and interrupt level (ipl) information: `-Process= "Net Background", ipl= 2, pid= 82`
::pid	The process ID (PID). • Some system messages include the process ID of the triggering process. The following sample output shows the format for process ID (pid) information: `-Process= "Net Background", ipl= 2, pid= 12345`

::traceback	The traceback string.
	• Some system messages describe internal errors and contain traceback information. This information, when included, will typically appear at the end of an error message. The following sample output shows the format for traceback information:
	`Apr 23 07:14:02: %ATMPA-3-CMDFAIL: ATM2/1/0 Command Failed at ../src-rsp/rsp_vip _atmdx.c - line 113, arg 32784`
	`-Process= "Net Background", ipl= 2, pid= 82`
	`-Traceback= 602D12AC 602CED14 60050B6C 602CFF74`

::syslog_facility	The syslog facility number used in the PRI portion of the syslog message sent to external syslog collectors (syslog hosts). • The syslog facility is given as a number, from 0 to 184. • The default is 184 (local7), but the value can be changed with the **logging facility** global configuration command.
::clear	Contains the string "- event cleared" or "NULL."
::version	The Cisco IOS software version, in the format "SYS_MAJORVERSION. SYS_MINORVERSION."

Table 1 Valid Variables for Syslog Filter Modules

Variable Name	Definition
::module_position	The position of this syslog filter module in the filter chain. The filter chain starts at one (1). • The value of this argument is determined by the order in which the scripts are referenced by the **logging filter** global configuration mode command.
::stream	The ESM message stream number. • The stream number will always be set to 2 (filtered stream) prior to the first filter being executed. • Syslog filter modules can change this value to a user-defined stream number in order to route the message to particular syslog collectors. • Stream numbers are allocated as follows: – Stream 0: Default (standard) syslog stream – Stream 1: XML tagged syslog stream – Stream 2: Default filtered syslog stream – Streams 3-9: Reserved – Streams 10-65536: User defined
::cli_args	The list of optional arguments specified during the filter configuration. A Tcl list containing any optional filter arguments specified when the filter was configured. This is the list of strings specified after the **args** keyword when the filter was configured with the **logging filter** command.
::msg_part	The message part. If an oversized syslog message has been split into multiple messages, this variable contains a number representing the message part (starting with 0).
::truncate	The incomplete message. If an oversized syslog message has been split into multiple messages, this variable will be nonzero if this message is incomplete (truncated).
::sev_prefix	The severity prefix string. Contains the optional severity prefix string.
::msg_prefix	The message prefix string. Contains the optional message prefix string.
::fac_prefix	The optional facility prefix string. Contains the optional facility prefix string.

Normal ESM Filter Processing

Each time a system logging message is generated, the syslog filter modules are called in a series. This series is determined by the ::module_position variable, which in turn is typically the order in which the modules are referenced in the system configuration (the order in which they are configured).

The output of one filter module becomes the input to the next. Because the input to the filters are the Tcl global namespace variables (as listed in Table 1), each filter can change any or all of these variables depending upon the purpose of the filter.

The only Tcl global variables that are automatically updated by the ESM framework between subsequent filter executions are the ::orig_msg and ::cli_args variables. The framework automatically sets the value of ::orig_msg to the return value of the filter module. Thus a filter that is designed to alter or filter the original message must not manually set the value for the ::orig_msg variable; the filter only needs to return the desired value. For example, the following one-line ESM filter

```
return "This is my new syslog message."
```

would ignore any message passed to it, and always change the output to the constant string "This is my new syslog message." If the module was the last filter in the chain, all ESM targets would receive this string as the final syslog message.

The one-line ESM filter

```
return ""
```

would block all syslog messages to the ESM stream. For example, the line

```
return $::orig_msg
```

would do nothing but pass the message along to the next filter in the chain. Thus, an ESM filter designed to suppress unwanted messages would look something like this:

```
if { [my_procedure_to_check_this_message] == 1 } {
     return $::orig_msg
} else {
     return ""
}
```

Depending upon their design, some filters may not use the ::orig_msg variable at all, but rather reconstruct a syslog message from its data elements (using ::format_string, ::msg_args, ::timestamp, and so on). For example, an XML tagging filter will tag the individual data elements, and disregard the original formatted message. It is important for such modules to check the ::orig_msg variable at the beginning of the Tcl script, so that if previous filter indicated that the message should not be sent out (::orig_msg is NULL), it would not bother to process the message, but simply return NULL also.

Cisco IOS commands can also be added to syslog filter modules using the **exec** and **config** Tcl commands. For example, if you wanted to add the source IP address to the syslog messages, and syslog messages were configured to be sent from the Ethernet 2/0 interface (using the **logging source-interface** command) you could issue the **show interface Ethernet 2/0** command during the module initialization by using the **exec** Tcl command within the script:

```
set source_ip_string [exec show ip int E2/0 | inc Internet]

puts $source_ip_string

"  Internet address is 10.4.2.63/24"
```

The script should then pass the output of that command to the syslog message. For further information on scripting within Cisco IOS software, see the "Cisco IOS Scripting with Tcl" feature module on Cisco.com.

Background ESM Filter Processing

In Tcl it is possible to queue commands for processing in the future by using the **after** Tcl command. The most common use of this command is to correlate (gather and summarize) events over a fixed interval of time, called the "correlation window." Once the window of interest expires, the filter will need to "wake up," and calculate or summarize the events that occurred during the window, and often send out a new syslog message to report the events. This background process is handled by the ESM Event Loop process, which allows the Tcl interpreter to execute queued commands after a certain amount of time has passed.

If your syslog filter module needs to take advantage of correlation windows, it must use the **after** Tcl command to call a summary procedure once the correlation window expires (see examples in the "Configuration Examples for the Embedded Syslog Manager" section on page 13). Because there is no normal filter chain processing when background processes are run, in order to produce output these filters must make use of one of two ESM Tcl extensions: **errmsg** or **esm_errmsg**.

During background processing, the commands that have been enqueued by the **after** command are not run in the context of the filter chain (as in normal processing), but rather are autonomous procedures that are executed in series by the Tcl interpreter. Thus, these background procedures should not operate on the normal Tcl global namespace variables (except for setting the global namespace variables for the next filter when using **esm_errmsg**), but should operate on variables stored in their own namespace. If these variables are declared outside of a procedure definition, they will be persistent from call to call.

The purpose of the **errmsg** Tcl command is to create a new message and send it out for distribution, bypassing any other syslog filter modules. The syntax of the **errmsg** command is:

```
errmsg <severity> <stream> <message_string>
```

The purpose of the **esm_errmsg** Tcl command is to create a new message, process it with any syslog filter modules below it in the filter chain, and then send it out for distribution. The syntax of the **esm_errmsg** command is:

```
esm_errmsg <module_position>
```

The key difference between the errmsg() Tcl function and the esm_errmsg() Tcl function is that **errmsg** ignores the filters and directly queues a message for distribution, while **esm_errmsg** will send a syslog message down the chain of filters.

In the following example, a new syslog message is created and sent out tagged as Alert severity 1 to the configured ESM logging targets (stream 2). One can assume the purpose of this filter would be to suppress the individual SYS-5-CONFIG messages over a thirty minute correlation window, and send out a summary message at the end of the window.

```
errmsg 1 2 "*Jan 24 09:34:02.539: %SYS-1-CONFIG_I: There have been 12
configuration changes to the router between Jan 24 09:04:02.539 and Jan 24
09:34:01.324"
```

In order to use **esm_errmsg**, because the remaining filters below this one will be called, this background process must populate the needed Tcl global namespace variables prior to calling **esm_errmsg**. Passing the ::module_position tells the ESM framework which filter to start with. Thus, filters using the **esm_errmsg** command should store their ::module_position (passed in the global namespace variables during normal processing) in their own namespace variable for use in background processing. Here is an example:

```
proc ::my_filter_namespace::my_summary_procedure{}
{
 set ::orig_msg "*Jan 24 09:34:02.539: %SYS-1-CONFIG_I: There have been 12
configuration changes to the router between Jan 24 09:04:02.539 and Jan 24
09:34:01.324"
 set ::timestamp "*Jan 24 09:34:02.539"
```

```
 set ::severity 1
 set ::stream 2
 set ::traceback ""
 set ::pid ""
 set ::process ""
 set ::format_string "There have been %d configuration changes to the router
between %s and %s"
 set ::msg_args {12 "Jan 24 09:04:01.539" "Jan 24 09:34:01.324"}
 esm_errmsg $::my_filter_namespace::my_module_position
}
```

The benefit of setting all the global namespace variables for the **esm_errmsg** command is that your filters will be modular, and it will not matter what order they are used in the ESM framework. For example, if you wish all of the messages destined for the ESM targets to suffixed with the message originator's hostname, you could write a one-line "hostname" filter and place it at the bottom of the filter chain:

```
return "$::orig_msg -- $::hostname"
```

In this example, if any of your filters generate new messages during background processing and they use **esm_errmsg** instead of **errmsg**, these messages will be clearly suffixed with the hostname.

What to Do Next

After creating your syslog filter module, you should store the file in a location accessible to the router. You can copy the file to local system memory, or store it on a network file server.

Configuring the Embedded Syslog Manager

To configure the ESM, specify one or more filters to be applied to generated syslog messages, and specify the syslog message target.

Prerequisites

One or more syslog filter modules must be available to the router.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **logging filter** *filter-url* [*position*] [**args** *filter-arguments*]
4. Repeat Step 3 for each syslog filter module that should be applied to system logging output.
5. **logging** [**console** | **buffered** | **monitor**] **filtered** [*level*]

   or

   logging host {*ip-address* | *host-name*} **filtered** [**stream** *stream-id*]
6. Repeat Step 5 for each desired system logging destination.
7. **logging source-interface** *type number*
8. **logging origin-id** {**hostname** | **ip** | **string** *user-defined-id*}
9. **end**
10. **show logging**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**logging filter** *filter-url* [*position*] [**args** *filter-arguments*] **Example:** `Router(config)# logging filter slot0:/escalate.tcl 1 args CONFIG_I 1`	Specifies one or more syslog filter modules to be applied to generated system logging messages. • Repeat this command for each syslog filter module that should be used. • The *filter-url* argument is the Cisco IOS File System location of the syslog filter module (script). The location can be in local memory, or a remote server using **tftp:**, **ftp:**, or **rcp:**. • The optional *position* argument specifies the order in which the syslog filter modules should be executed. If this argument is omitted, the specified module will be positioned as the last module in the chain. • Filters can be re-ordered on the fly by re-entering the **logging filter** command and specifying a different position. • The optional **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. • To remove a module from the list of modules to be executed, use the **no** form of this command.
Step 4	Repeat Step 3 for each syslog filter module that should be applied to system logging output.	—

	Command or Action	Purpose

Step 5	**logging** [**console** | **buffered** | **monitor**] **filtered** [*level*]
	
	or
	
	logging host {*ip-address* | *host-name*} **filtered** [**stream** *stream-id*]
	
	Example:
	`Router(config)# logging console filtered informational`
	or
	
	Example:
	`Router(config)# logging host 209.165.200.225 filtered stream 20`
		Specifies the target for ESM filtered syslog output.
		• ESM filtered syslog messages can be sent to the console, a monitor (TTY and Telnet connections), the system buffer, or to remote hosts.
		• The optional *level* argument limits the sending of messages to those at or numerically lower than the specified value. For example, if level **1** is specified, only messages at level 1 (alerts) or level 0 (emergencies) will be sent to the specified target. The level can be specified as a keyword or number.
		• When logging to the console, monitor connection, or system buffer, the severity threshold specified by the *level* argument takes precedence over the ESM filtering. In other words, even if the ESM filters return a message to be delivered to ESM targets, if the severity doesn't meet the configured threshold (is numerically higher than the level value), it will not be delivered.
		• When logging to remote hosts, the stream tag allows you to specify a destination based on the type of message. The **stream** *stream-id* syntax allows you to configure the ESM to send only messages that have a specified stream value to a certain host.
		• The stream value is applied to messages by the configured syslog filter modules. For example, all Severity 5 messages could have a stream tag of "20" applied. You can then specify that all messages with a stream tag of "20" be sent to the host at 209.165.200.225 using the command:
		logging host 209.165.200.225 filtered stream 20

Step 6	Repeat Step 5 for each desired system logging destination.	• By issuing the logging host command multiple times, you can specify different targets for different system logging streams. • Similarly, you can configure messages at different severity levels to be sent to the console, monitor connection, or system buffer. For example, you may want to display only very important messages to the screen (using a monitor or console connection) at your network operations center (NOC).
Step 7	**logging source-interface** *type number* **Example:** `Router(config)# logging source-interface ethernet 0`	(Optional) Specifies the source interface for syslog messages sent to remote syslog hosts. • Normally, a syslog messages sent to remote hosts will use whatever interface is available at the time of the message generation. This command forces the router to send syslog messages to remote hosts only from the specified interface.

	Command or Action	Purpose
Step 8	**logging origin-id** {**hostname** | **ip** | **string** *user-defined-id*} **Example:** `Router(config)# logging origin-id string "Domain 2, Router 5"`	(Optional) Allows you to add an origin identifier to syslog messages sent to remote hosts. • The origin identifier is added to the beginning of all syslog messages sent to remote hosts. The identifier can be the hostname, the IP address, or any text that you specify. • The origin identifier is useful for identifying the source of system logging messages in cases where you send syslog output from multiple devices to a single syslog host.
Step 9	**end** **Example:** `Router(config)# end`	Ends your current configuration session and returns the CLI to privileged EXEC mode.
Step 10	**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 shows the data stored in the buffer. • The order in which syslog filter modules are listed in the output of this command is the order in which the filter modules are executed.

Configuration Examples for the Embedded Syslog Manager

This section provides the following configuration examples:

- Configuring the Embedded Syslog Manager: Example, page 13
- Syslog Filter Module: Example, page 14

Configuring the Embedded Syslog Manager: Example

In the following example, ESM filter logging is enabled for the console connection, standard logging is enabled for the monitor connection and for the buffer, and XML-formatted logging is enabled for the host at 209.165.200.225:

```
Router(config)# logging filter tftp://209.165.200.225/ESM/escalate.tcl
Router(config)# logging filter slot0:/email.tcl user@example.com
Router(config)# logging filter slot0:/email_guts.tcl
Router(config)# logging console filtered
Router(config)# logging monitor 4
Router(config)# logging buffered debugging
Router(config)# logging host 209.165.200.225 xml
Router(config)# end
Router# show logging
Syslog logging: enabled (0 messages dropped, 8 messages rate-limited,
                0 flushes, 0 overruns, xml disabled, filtering enabled)
    Console logging: level debugging, 21 messages logged, xml disabled,
                     filtering enabled
    Monitor logging: level warnings , 0 messages logged, xml disabled,
```

```
                    filtering disabled
    Buffer logging: level debugging, 30 messages logged, xml disabled,
                    filtering disabled
    Logging Exception size (8192 bytes)
    Count and timestamp logging messages: disabled

Filter modules:
    tftp://209.165.200.225/ESM/escalate.tcl
    slot0:/email.tcl user@example.com

    Trap logging: level informational, 0 message lines logged
        Logging to 209.165.200.225, 0 message lines logged, xml enabled,
              filtering disabled

Log Buffer (8192 bytes):

*Jan 24 09:34:28.431: %SYS-5-CONFIG_I: Configured from console by console
*Jan 24 09:34:51.555: %SYS-5-CONFIG_I: Configured from console by console
*Jan 24 09:49:44.295: %SYS-5-CONFIG_I: Configured from console by console
Router#
```

Syslog Filter Module: Example

Syslog Script Modules are Tcl scripts. The following examples are provided to assist you in developing your own Syslog Script Modules.

Note These script modules are provided as examples only, and are not supported by Cisco Systems, Inc. No guarantees, expressed or implied, are provided for the functionality or impact of these scripts.

This appendix contains the following syslog filter module examples:

Severity Escalation: Example

This ESM syslog filter module example watches for a single mnemonic (supplied via the first CLI argument) and escalates the severity of the message to that specified by the second CLI argument.

```
#  ====================================================================
#  Embedded Syslog Manager                              ||        ||
#                                                       ||        ||
#  Severity Escalation Filter                          ||||      ||||
#                                                  ..:||||||:..:||||||:..
#                                                 -------------------------
#                                                C i s c o  S y s t e m s
#  ====================================================================
#
# Usage: Set CLI Args to "mnemonic new_severity"
```

```
#
# Namespace: global

# Check for null message

if { [string length $::orig_msg] == 0} {
   return ""
}

if { [info exists ::cli_args] } {
    set args [split $::cli_args]
    if { [ string compare -nocase [lindex $args 0] $::mnemonic ] == 0 } {
        set ::severity [lindex $args 1]
        set sev_index [ string first [lindex $args 0] $::orig_msg ]
        if {  $sev_index >= 2 } {
           incr sev_index -2
           return [string replace $::orig_msg $sev_index $sev_index \
              [lindex $args 1]]
        }
    }
}
return $::orig_msg
```

Message Counting: Example

This ESM syslog filter module example is divided into two files for readability. The first file allows the user to configure those messages that they wish to count and how often to summarize (correlation window) by populating the msg_to_watch array. The actual procedures are in the counting_guts.tcl file. Note the use of the separate namespace “counting” to avoid conflict with other ESM filters that may also perform background processing.

```
#  ========================================================================
#  Embedded Syslog Manager                          ||        ||
#                                                   ||        ||
#  Message Counting Filter                         ||||      ||||
#                                              ..:||||||:..:||||||:..
#                                             ------------------------
#                                             C i s c o  S y s t e m s
# ========================================================================

#
# Usage:
# 1) Define the location for the counting_guts.tcl script
#
# 2) Define message categories to count and how often to dump them (sec)
#    by populating the "msg_to_watch" array below.
#    Here we define category as facility-severity-mnemonic
#    Change dump time to 0 to disable counting for that category
#
# Namespace: counting


namespace eval ::counting {

    set sub_script_url tftp://123.123.123.123/ESM/counting_guts.tcl

    array set msg_to_watch {
        SYS-5-CONFIG_I            5
    }

#  ====================== End User Setup ============================
```

```
# Initialize processes for counting

    if { [info exists init] == 0 } {
       source $sub_script_url
       set position $module_position
    }

# Process the message

process_category

} ;# end namespace counting
```

Message Counting Support Module (counting_guts.tcl)

```
#  ===================================================================
#  Embedded Syslog Manager                              ||        ||
#                                                       ||        ||
#  Message Counting Support Module                     ||||      ||||
#                                                   ..:||||||:..:||||||:..
#  (No User Modification)                          ------------------------
#                                                  C i s c o  S y s t e m s
# ====================================================================================


namespace eval ::counting {

# namespace variables

array set cat_msg_sev {}
array set cat_msg_traceback {}
array set cat_msg_pid {}
array set cat_msg_proc {}
array set cat_msg_ts {}
array set cat_msg_buginfseq {}
array set cat_msg_name {}
array set cat_msg_fac {}
array set cat_msg_format {}
array set cat_msg_args {}
array set cat_msg_count {}
array set cat_msg_dump_ts {}

# Should I count 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 clear_category {index} {
        variable cat_msg_sev
        variable cat_msg_traceback
        variable cat_msg_pid
        variable cat_msg_proc
        variable cat_msg_ts
        variable cat_msg_buginfseq
        variable cat_msg_name
        variable cat_msg_fac
        variable cat_msg_format
```

```
        variable cat_msg_args
        variable cat_msg_count
        variable cat_msg_dump_ts

        unset cat_msg_sev($index) cat_msg_traceback($index) cat_msg_pid($index)\
              cat_msg_proc($index) cat_msg_ts($index) \
              cat_msg_buginfseq($index)  cat_msg_name($index) \
              cat_msg_fac($index) cat_msg_format($index) cat_msg_args($index)\
              cat_msg_count($index) cat_msg_dump_ts($index)
    }

# send out the counted messages

   proc dump_category {category} {
        variable cat_msg_sev
        variable cat_msg_traceback
        variable cat_msg_pid
        variable cat_msg_proc
        variable cat_msg_ts
        variable cat_msg_buginfseq
        variable cat_msg_name
        variable cat_msg_fac
        variable cat_msg_format
        variable cat_msg_args
        variable cat_msg_count
        variable cat_msg_dump_ts
        variable poll_interval

        set dump_timestamp [cisco_service_timestamp]

foreach index [array names cat_msg_count $category] {
      set fsm "$cat_msg_fac($index)-$cat_msg_sev($index)-$cat_msg_name($index)"
      set ::orig_msg \
        [format "%s%s: %%%s: %s %s %s %s - (%d occurence(s) between %s and %s)"\
        $cat_msg_buginfseq($index)\
          $dump_timestamp\
          $fsm \
          [uplevel 1 [linsert $cat_msg_args($index) 0 ::format
$cat_msg_format($index) ]] \
          $cat_msg_pid($index) \
          $cat_msg_proc($index) \
          $cat_msg_traceback($index) \
          $cat_msg_count($index) \
          $cat_msg_ts($index) \
          $dump_timestamp]

# Prepare for remaining ESM filters

                set ::severity $cat_msg_sev($index)
                set ::traceback $cat_msg_traceback($index)
                set ::pid $cat_msg_pid($index)
                set ::process $cat_msg_proc($index)
                set ::timestamp $cat_msg_ts($index)
                set ::buginfseq $cat_msg_buginfseq($index)
                set ::mnemonic $cat_msg_name($index)
                set ::facility $cat_msg_fac($index)
                set ::format_string $cat_msg_format($index)
                set ::msg_args [split $cat_msg_args($index)]

                esm_errmsg $counting::position
                clear_category $index
        }
  }
```

```
# See if this message already has come through since the last dump.
# If so, increment the count, otherwise store it.

   proc process_category {} {
        variable cat_msg_sev
        variable cat_msg_traceback
        variable cat_msg_pid
        variable cat_msg_proc
        variable cat_msg_ts
        variable cat_msg_buginfseq
        variable cat_msg_name
        variable cat_msg_fac
        variable cat_msg_format
        variable cat_msg_args
        variable cat_msg_count
        variable cat_msg_dump_ts

        if { [string length $::orig_msg] == 0} {
           return ""
        }

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

        set correlation_window [expr [ query_category $category ] * 1000]

        if { $correlation_window == 0 } {
            return $::orig_msg
        }

        set message_args [join $::msg_args]
        set index "$category,[lindex $::msg_args 0]"
        if { [info exists cat_msg_count($index)] } {
                  incr cat_msg_count($index)
         } else {
              set cat_msg_sev($index) $::severity
              set cat_msg_traceback($index) $::traceback
              set cat_msg_pid($index) $::pid
              set cat_msg_proc($index) $::process
              set cat_msg_ts($index) $::timestamp
              set cat_msg_buginfseq($index) $::buginfseq
              set cat_msg_name($index) $::mnemonic
              set cat_msg_fac($index) $::facility
              set cat_msg_format($index) $::format_string
              set cat_msg_args($index) $message_args
              set cat_msg_count($index) 1
              set cat_msg_dump_ts($index) [clock seconds]
              catch [after $correlation_window counting::dump_category $index]
         }
         return ""
   }



# Initialized
set init 1

} ;#end namespace counting
```

XML Tagging: Example

This ESM syslog filter module applies user-defined XML tags to syslog messages.

```
#  ================================================================
#  Embedded Syslog Manager                              ||        ||
#                                                       ||        ||
#  XML Tagging Filter                                  ||||      ||||
#                                                   ..:||||||:..:||||||:..
#                                                  ------------------------
#                                                  C i s c o  S y s t e m s
#  ================================================================
#
# Usage: Define desired tags below.
#
# Namespace: xml

# Check for null message

    if { [string length $::orig_msg] == 0} {
       return ""
    }

namespace eval xml {

####  define tags ####
set MSG_OPEN "<ios-log-msg>"
set MSG_CLOSE "</ios-log-msg>"
set FAC_OPEN   "<facility>"
set FAC_CLOSE  "</facility>"
set SEV_OPEN   "<severity>"
set SEV_CLOSE  "</severity>"
set MNE_OPEN   "<msg-id>"
set MNE_CLOSE  "</msg-id>"
set SEQ_OPEN   "<seq>"
set SEQ_CLOSE  "</seq>"
set TIME_OPEN  "<time>"
set TIME_CLOSE "</time>"
set ARGS_OPEN  "<args>"
set ARGS_CLOSE "</args>"
set ARG_ID_OPEN "<arg id="
set ARG_ID_CLOSE "</arg>"
set PROC_OPEN "<proc>"
set PROC_CLOSE "</proc>"
set PID_OPEN "<pid>"
set PID_CLOSE "</pid>"
set TRACE_OPEN "<trace>"
set TRACE_CLOSE "</trace>"

#  ======================= End User Setup ==============================

#### clear result ####

set result ""

#### message opening, facility, severity, and name ####
append result $MSG_OPEN $FAC_OPEN $::facility $FAC_CLOSE $SEV_OPEN $::severity
$SEV_CLOSE $MNE_OPEN $::mnemonic $MNE_CLOSE

#### buginf sequence numbers ####

if { [string length $::buginfseq ] > 0 } {
    append result $SEQ_OPEN $::buginfseq $SEQ_CLOSE
```

```
}

#### timestamps ####

if { [string length $::timestamp ] > 0 } {
    append result $TIME_OPEN $::timestamp $TIME_CLOSE
}

#### message args ####
if { [info exists ::msg_args] } {
    if { [llength ::msg_args] > 0 } {
        set i 0
        append result $ARGS_OPEN
        foreach arg $::msg_args {
            append result $ARG_ID_OPEN $i ">" $arg $ARG_ID_CLOSE
                incr i
        }
        append result $ARGS_CLOSE
    }
}

#### traceback ####

if { [string length $::traceback ] > 0 } {
    append result $TRACE_OPEN $::traceback $TRACE_CLOSE
}

#### process ####

if { [string length $::process ] > 0 } {
    append result $PROC_OPEN $::process $PROC_CLOSE
}

#### pid ####

if { [string length $::pid ] > 0 } {
    append result $PID_OPEN $::pid $PID_CLOSE
}

#### message close ####

append result $MSG_CLOSE


return "$result"

} ;# end namespace xml
```

SMTP-based E-mail Alert: Example

This ESM syslog filter module example watches for configuration messages and sends them to the e-mail address supplied as a CLI argument. This filter is divided into two files. The first file implements the filter, and the second file implements the SMTP client.

```
#  ====================================================================
#  Embedded Syslog Manager                              ||        ||
#                                                       ||        ||
#  Email Filter                                        ||||      ||||
#  (Configuration Change Warning)                  ..:||||||:..:||||||:..
#                                                 ------------------------
#                                                  C i s c o  S y s t e m s
#   ====================================================================
```

```
# Usage:  Provide email address as CLI argument.  Set email server IP in
#         email_guts.tcl
#
# Namespace: email

if { [info exists email::init] == 0 } {
   source tftp://123.123.123.123/ESM/email_guts.tcl
}

# Check for null message

if { [string length $::orig_msg] == 0} {
      return ""
   }

if { [info exists ::msg_args] } {
    if { [string compare -nocase CONFIG_I $::mnemonic ] == 0 } {
                email::sendmessage $::cli_args $::mnemonic \
                [string trim $::orig_msg]
    }
}
return $::orig_msg
```

E-mail Support Module (email_guts.tcl)

```
#  ====================================================================
#  Embedded Syslog Manager                              ||        ||
#                                                       ||        ||
#  Email Support Module                                 ||||      ||||
#                                                    ..:||||||:..:||||||:..
#                                                   ------------------------
#                                                    C i s c o  S y s t e m s
#   ====================================================================
#
# Usage: Set email host IP, from, and friendly strings below.
#

namespace eval email {

    set sendmail(smtphost) 64.102.17.214
    set sendmail(from) $::hostname
    set sendmail(friendly) $::hostname


    proc sendmessage {toList subject body} {

        variable sendmail

        set smtphost $sendmail(smtphost)
        set from $sendmail(from)
        set friendly $sendmail(friendly)

        set sockid [socket $smtphost 25]

## DEBUG
set status [catch {
        puts $sockid "HELO $smtphost"
        flush $sockid
        set result [gets $sockid]

        puts $sockid "MAIL From:<$from>"
        flush $sockid
        set result [gets $sockid]
```

```
        foreach to $toList {
            puts $sockid "RCPT To:<$to>"
            flush $sockid
        }

        set result [gets $sockid]

        puts  $sockid "DATA "
        flush $sockid
        set result [gets  $sockid]

        puts  $sockid "From: $friendly <$from>"
        foreach to $toList {
            puts $sockid "To:<$to>"
        }
        puts  $sockid "Subject: $subject"
        puts  $sockid "\n"

        foreach line [split $body "\n"] {
            puts  $sockid " $line"
        }

        puts  $sockid "."
        puts  $sockid "QUIT"
        flush $sockid
        set result [gets  $sockid]

} result]

        catch {close $sockid }
    if {$status} then {
        return -code error $result
    }
}

} ;# end namespace email

set email::init 1
```

Stream: Example

This ESM syslog filter module example watches for a given facility (first CLI argument) and routes these messages to a given stream (second CLI argument).

```
#  ===================================================================
#  Embedded Syslog Manager                              ||        ||
#                                                       ||        ||
#  Stream Filter (Facility)                            ||||      ||||
#                                                  ..:||||||:..:||||||:..
#                                                 ------------------------
#                                                  C i s c o  S y s t e m s
#   ===================================================================

# Usage:  Provide facility and stream as CLI arguments.
#
# Namespace: global

# Check for null message

#  ======================= End User Setup ==============================
```

```
set args [split $::cli_args]

if { [info exists ::msg_args] } {
    if { $::facility == [lindex $args 0] } {
                set ::stream [lindex $args 1]
    }
}
return $::orig_msg}
```

Source IP Tagging: Example

The **logging source-interface** CLI command can be used to specify a source IP address in all syslog packets sent from the router. The following syslog filter module example demonstrates the use of **show** CLI commands (**show running-config** and **show ip interface** in this case) within a filter module to add the source IP address to syslog messages. The script looks for the local namespace variable "source_ip::init" first. If the variable is not defined in the first syslog message processed, the filter will run the **show** commands and use regular expressions to get the source-interface and then its IP address.

Note that in this script, the **show** commands are only run once. If the source-interface or its IP address were to be changed, the filter would have to be re-initialized to pick up the new information. (You could have the show commands run on every syslog message, but this would not scale very well.)

```
#  ====================================================================
#  Embedded Syslog Manager                              ||        ||
#                                                       ||        ||
#  Source IP Module                                    ||||      ||||
#                                                   ..:||||||:..:||||||:..
#                                                  ------------------------
#                                                   C i s c o  S y s t e m s
#   ====================================================================

# Usage: Adds Logging Source Interface IP address to all messages.
#
# Namespace:source_ip
#
#  ======================= End User Setup ==============================

namespace eval ::source_ip {

    if { [info exists init] == 0 } {
       if { [catch {regexp {^logging source-interface (.*$)} [exec show
run | inc logging source-interface] match source_int}]} {
            set suffix "No source interface specified"
       } elseif { [catch {regexp {Internet address is (.*)/.*$} [exec
show ip int $source_int | inc Internet] match ip_addr}]} {
            set suffix "No IP address configured for source interface"
       } else {
            set suffix $ip_addr
       }
       set init 1
    }

    if { [string length $::orig_msg] == 0} {
      return ""
    }

    return "$::orig_msg - $suffix"

} ;# end namespace source_ip
```

Additional References

The following sections provide references related to the Embedded Syslog Manager feature.

Related Documents

Related Topic	Document Title
System Message Logging	*Troubleshooting and Fault Management* module
XML Formatted System Message Logging	*XML Interface to Syslog Messages* module
Tcl 8.3.4 Support in Cisco IOS Software	*Cisco IOS Scripting with Tcl* module
Network Management commands (including logging commands): complete command syntax, defaults, command mode, command history, usage guidelines, and examples	*Cisco IOS Network Management Command Reference*

Standards

Standard	Title
No new or modified standards are supported, and support for existing standards has not been modified.	—

MIBs

MIB	MIBs Link
No new or modified standards are supported, and support for existing standards has not been modified.	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFCs[1]	Title
RFC-3164	The BSD Syslog Protocol • This RFC is informational only. The Cisco implementation of syslog does not claim full compliance with the protocol guidelines mentioned in this RFC.

1. Not all supported RFCs are listed.

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/techsupport

Command Reference

The following commands are introduced or modified in the feature or features documented in this module. For information about these commands, see the *Cisco IOS Network Management Command Reference* at http://www.cisco.com/en/US/docs/ios/netmgmt/command/reference/nm_book.html. For information about all Cisco IOS commands, use the Command Lookup Tool at http://tools.cisco.com/Support/CLILookup or the *Cisco IOS Master Command List, All Releases*, at http://www.cisco.com/en/US/docs/ios/mcl/allreleasemcl/all_book.html.

- **logging buffered filtered**
- **logging console filtered**
- **logging filter**
- **logging host**
- **logging monitor filtered**
- **logging origin-id**
- **show logging**

Feature Information for Embedded Syslog Manager

Table 2 lists the release history for this feature.

Not all commands may be available in your Cisco IOS software release. For release information about a specific command, see the command reference documentation.

Use Cisco Feature Navigator to find information about platform support and software image support. Cisco Feature Navigator enables you to determine which Cisco IOS and Catalyst OS software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Note Table 2 lists only the Cisco IOS software release that introduced support for a given feature in a given Cisco IOS software release train. Unless noted otherwise, subsequent releases of that Cisco IOS software release train also support that feature.

Table 2 Feature Information for Embedded Syslog Manager

Feature Name	Releases	Feature Information
Embedded Syslog Manager	12.3(2)T 12.3(2)XE 12.2(25)S 12.2(33)SRC 12.2(33)SB Cisco IOS XE Release 2.1 12.2(33)SXI	The Embedded Syslog Manager (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. In Cisco IOS XE Release 2.1, this feature was introduced on Cisco ASR 1000 series routers. The following commands were introduced or modified: **logging buffered filtered**, **logging console filtered**, **logging filter**, **logging host**, **logging monitor filtered**, **logging origin-id**, **show logging**.

Glossary

Note Refer to the *Internetworking Terms and Acronyms* for terms not included in this glossary.

console—In the context of this feature, specifies the connection (CTY or console line) to the console port of the router. Typically, this is a terminal attached directly to the console port, or a PC with a terminal emulation program. Corresponds to the **show terminal** command.

monitor—In the context of this feature, specifies the TTY (TeleTYpe terminal) line connection at a line port. In other words, the "monitor" keyword corresponds to a terminal line connection or a Telnet (terminal emulation) connection. TTY lines (also called ports) communicate with peripheral devices such as terminals, modems, and serial printers. An example of a TTY connection is a PC with a terminal emulation program connected to the device using a dial-up modem.

SEMs—Abbreviation for system error messages. "System error messages" is the term formerly used for messages generated by the system logging (syslog) process. Syslog messages use a standardized format, and come in 8 severity levels, from "emergencies" (level 0) to "debugging" (level 7). The term "system error message" is actually misleading, as these messages can include notifications of router activity beyond "errors" (such as informational notices).

syslog—Abbreviation for the system message logging process in Cisco IOS software. Also used to identify the messages generated, as in "syslog messages." Technically, the term "syslog" refers only to the process of logging messages to a remote host or hosts, but is commonly used to refer to all Cisco IOS system logging processes.

trap—A trigger in the system software for sending error messages. In the context of this feature, "trap logging" means logging messages to a remote host. The remote host is actually a syslog host from the perspective of the device sending the trap messages, but because the receiving device typically provides collected syslog data to other devices, the receiving device is also referred to as a "syslog server."

Logging to Local Nonvolatile Storage (ATA Disk)

First Published: August 26, 2003
Last Updated: March 10, 2009

The Logging to Local Nonvolatile Storage (ATA Disk) feature enables system logging messages to be saved on an advanced technology attachment (ATA) flash disk. Messages saved on an ATA drive persist after a router is rebooted.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the "Feature Information for Logging to Local Nonvolatile Storage (ATA Disk)" section on page 8.

Use Cisco Feature Navigator to find information about platform support and Cisco IOS and Catalyst OS software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Contents

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

Prerequisites for Logging to Local Nonvolatile Storage (ATA Disk)

The logging buffered Command Must Be Enabled

Before the Logging to Local Nonvolatile Storage (ATA Disk) feature can be enabled with the **logging persistent** command, you must enable the logging of messages to an internal buffer with the **logging buffered** command. For additional information, refer to the "Writing Logging Messages to an ATA Disk" section on page 3, and to the "Related Documents" section on page 6.

Restrictions for Logging to Local Nonvolatile Storage (ATA Disk)

Available ATA Disk Space Constrains the Size and Number of Stored Log Files

The amount of ATA disk space allocated to system logging messages constrains the number of logging files that can be stored. When the allocation threshold is passed, the oldest log file in the directory is deleted to make room for new system logging messages. To permanently store system logging messages, you must archive them to an external device. For more information, refer to the "Copying Logging Messages to an External Disk" section on page 4.

Note Logging to Local Nonvolatile Storage can use up to 2 GB of storage space.

Information About Logging to Local Nonvolatile Storage (ATA Disk)

The Logging to Local Nonvolatile Storage (ATA Disk) feature adds a router's ATA flash disk as a storage destination for logging messages. When using this feature, be sure to understand the following concepts:

System Logging Messages

System logging messages include error and debug messages generated by application programming interfaces (APIs) on the router. Typically, logging messages are stored in a router's memory buffer; when the buffer is full, older messages are overwritten by new messages. All logging messages are erased from the memory buffer when the router reboots.

ATA Flash Disks

ATA flash disks are PC cards included with some Cisco routers, which are used to provide nonvolatile data storage. The greater the capacity of the ATA flash disk, the more data, such as logging messages, it can hold. Logging messages written to an ATA flash disk persist when the router reboots.

How to Configure Logging to Local Nonvolatile Storage (ATA Disk)

This section contains the following procedures:

Writing Logging Messages to an ATA Disk

Perform this task to enable the Logging to Local Nonvolatile Storage (ATA Disk) feature and write logging messages to an ATA flash disk:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **logging buffered** [*buffer-size* | *severity-level*]
4. **logging persistent** [**url** {**disk0:***/directory* | **disk1:***/directory*}] [**size** *filesystem-size*] [**filesize** *logging-file-size*] [**batch** *batch-size*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enables global configuration mode.

	Command or Action	Purpose
Step 3	**logging buffered** [*buffer-size* | *severity-level*] **Example:** `Router(config)# logging buffered`	Enables system message logging to a local buffer and limits messages logged to the buffer based on severity. • The optional *buffer-size* argument specifies the size of the buffer from 4096 to 4294967295 bytes. The default size varies by platform. • The optional *severity-level* argument limits the logging of messages to the buffer to those no less severe than the specified level.
Step 4	**logging persistent** [**url** {**disk0:**/*directory* | **disk1:**/*directory*}] [**size** *filesystem-size*] [**filesize** *logging-file-size*] [**batch** *batch-size*] **Example:** `Router(config)# logging persistent url disk0:/syslog size 134217728 filesize 16384 batch 5098`	Writes logging messages from the memory buffer to the specified directory on the router's ATA disk. • Before logging messages are written to a file on the ATA disk, the Cisco IOS software checks to see if there is sufficient disk space. If not, the oldest file of logging messages (by timestamp) is deleted, and the current file is saved. • The filename format of log files is *log_MM:DD:YYYY::hh:mm:ss* (for example, log_06:10:2008::07:42:14). For Release 12.4(20)T and later releases, the filename format is changed to *log_YYYYMMDD-hhmmss* (for example, log_20080610-074214). This feature supports only one log file per second due to its filename format, which contains a timestamp suffix down to the seconds level.

Copying Logging Messages to an External Disk

Perform this task to copy logging messages from the ATA flash disk to an external disk.

SUMMARY STEPS

1. **enable**
2. **copy** *source-url destination-url*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**copy** *source-url destination-url* **Example:** `Router# copy disk0:/syslog ftp://myuser/mypass@192.168.1.129/syslog`	Copies the specified file or directory on the ATA flash disk via FTP to the specified URL.

Configuration Examples for Logging to Local Nonvolatile Storage (ATA Disk)

This section provides the following configuration examples:

- Writing Logging Messages to an ATA Disk: Example, page 5
- Copying Logging Messages to an External Disk: Example, page 5

Writing Logging Messages to an ATA Disk: Example

The following example shows how to write up to 134217728 bytes (128 MB) of logging messages to the syslog directory of disk 0, specifying a file size of 16384 bytes:

```
Router(config)# logging buffered
Router(config)# logging persistent url disk0:/syslog size 134217728 filesize 16384
```

Copying Logging Messages to an External Disk: Example

The following example shows how to copy logging messages from the router's ATA flash disk to an external disk:

```
Router# copy disk0:/syslog ftp://myuser/mypass@192.168.1.129/syslog
```

Additional References

The following sections provide references related to the Logging to Local Nonvolatile Storage (ATA Disk) feature.

Related Documents

Related Topic	Document Title
copy command	*Cisco IOS Configuration Fundamentals Command Reference*
Network Management commands (including **logging** commands): complete command syntax, defaults, command mode, command history, usage guidelines, and examples	*Cisco IOS Network Management Command Reference*

Standards

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

MIBs

MIBs	MIBs Link
No new or modified MIBs are supported by this feature, and support for existing MIBs has not been modified by this feature.	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

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

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/techsupport

Command Reference

The following commands are introduced or modified in the feature or features documented in this module. For information about these commands, see the *Cisco IOS Network Management Command Reference* at http://www.cisco.com/en/US/docs/ios/netmgmt/command/reference/nm_book.html. For information about all Cisco IOS commands, go to the Command Lookup Tool at http://tools.cisco.com/Support/CLILookup or to the *Cisco IOS Master Commands List*.

- **logging buffered**
- **logging persistent**

Feature Information for Logging to Local Nonvolatile Storage (ATA Disk)

Table 1 lists the release history for this feature.

Not all commands may be available in your Cisco IOS software release. For release information about a specific command, see the command reference documentation.

Use Cisco Feature Navigator to find information about platform support and software image support. Cisco Feature Navigator enables you to determine which Cisco IOS and Catalyst OS software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Note Table 1 lists only the Cisco IOS software release that introduced support for a given feature in a given Cisco IOS software release train. Unless noted otherwise, subsequent releases of that Cisco IOS software release train also support that feature.

Table 1 Feature Information for Logging to Local Nonvolatile Storage (ATA Disk)

Feature Name	Releases	Feature Information
Logging to Local Nonvolatile Storage (ATA Disk)	12.0(26)S 12.2(25)S 12.2(28)SB 12.2(33)SRB 12.4(15)T 12.2(33)SB 12.4(20)T	The Logging to Local Nonvolatile Storage (ATA Disk) feature enables system logging messages to be saved on an advanced technology attachment (ATA) flash disk. Messages saved on an ATA drive persist after a router is rebooted. The following commands were introduced or modified: **logging persistent**, **logging buffered**.

Reliable Delivery and Filtering for Syslog

First Published: November 17, 2006
Last Updated: November 14, 2008

The Reliable Delivery and Filtering for Syslog feature allows a device to be customized for receipt of syslog messages. This feature provides reliable and secure delivery for syslog messages using Blocks Extensible Exchange Protocol (BEEP). Additionally, it allows multiple sessions to a single logging host, independent of the underlying transport method, and provides a filtering mechanism called a message discriminator.

This module describes the functions of the Reliable Delivery and Filtering for Syslog feature and how to configure them in a network.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the "Feature Information for Reliable Delivery and Filtering for Syslog" section on page 17.

Use Cisco Feature Navigator to find information about platform support and Cisco IOS and Catalyst OS software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Contents

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

Prerequisites for Reliable Delivery and Filtering for Syslog

- Router level rate limit is set to meet business needs, network traffic requirements, or performance requirements.
- Each BEEP session must have an RFC 3195-compliant syslog-RAW exchange profile.
- A Simple Authentication and Security Layer (SASL) profile specifying “DIGEST-MD5” for provisioning services must be established when a crypto image is used.
- Syslog servers must be compatible with BEEP.
- Syslog server applications must be capable of handling multiple sessions to use the multiple session capability of the Reliable Delivery and Filtering for Syslog feature.

Restrictions for Reliable Delivery and Filtering for Syslog

- Only the syslog-RAW, SASL, and Transport Layer Security (TLS) profiles are supported.
- Both ends of a syslog session must use the same transport method.
- A message discriminator must be defined before it can be associated with a specific syslog session.
- A syslog session can be associated with only one message discriminator.
- Message delivery with User Datagram Protocol (UDP) will be faster than with either TCP or BEEP.

Information About Reliable Delivery and Filtering for Syslog

To use the Reliable Delivery and Filtering for Syslog feature, you should understand the following concepts:

BEEP Transport Support

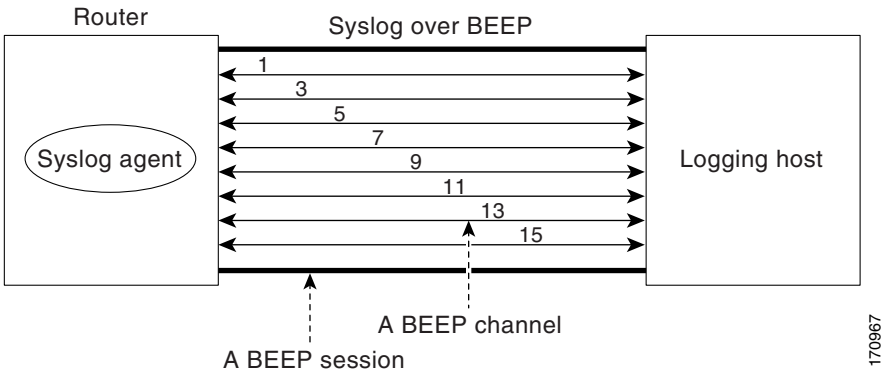
BEEP is a generic application protocol framework for connection-oriented, asynchronous interactions. It is intended to provide the features that traditionally have been duplicated in various protocol implementations. BEEP typically runs on top of TCP and allows the exchange of messages. Unlike HTTP and similar protocols, either end of the connection can send a message at any time. BEEP also includes facilities for encryption and authentication and is highly extensible.

BEEP as a transport protocol for syslog messages provides multiple channels. Each channel can be configured for a separate session to the same host. BEEP provides reliable transport. Syslog messages sent over a BEEP connection are guaranteed to be delivered in sequence.

With command-line interface (CLI) commands introduced in the Reliable Delivery and Filtering for Syslog feature, you can configure a new BEEP session to have a maximum of eight channels.

Figure 1 shows a BEEP session with eight channels, allowing eight separate syslog sessions.

Figure 1 BEEP Session with Eight Channels



Channels are identified as 1, 3, 5, 7, 9, 11, 13, and 15. The number of available channels (eight) was designed to correspond to the number of severity levels of classic RFC-3164 syslog messages (0 to 7). Message discriminators can be used such that severity levels are mapped to BEEP channels. An intelligent BEEP syslog server (depending upon the BEEP stack used) could use this mapping to prioritize messages with higher severity (see RFC 3081, section 3.1.4). Unless associated with a message discriminator, all syslog sessions (channels) receive all syslog messages.

Syslog Message

A syslog message has a sequence number that allows the host to use the number as an identifier for the message as well as to detect whether there were any gaps in the messages that were received. Syslog messages are numbered consecutively. The reliability of BEEP does not replace the need for sequence numbers, which are required for the following reasons:

- A sequence number provides an easy way to identify a syslog message. Independent of reliability considerations, the sequence number serves as a message identifier.
- A BEEP session may not be in place for the entire time that a device sending syslog messages is up. Sequence numbers provide a way for management applications to assess whether messages were missed between BEEP sessions.
- BEEP is only one of several transports. Unreliable transports are also used and the syslog protocol should not rely on a reliable transport always being provided.

The existing numbering scheme for syslog messages is limited with the extension of syslog to accommodate advanced message discrimination features and multiple hosts. Message discrimination leads to gaps in the sequence numbers, meaning that hosts lose the ability to detect whether they have missed a message. If syslog messages are numbered consecutively on each session to avoid the gaps in sequence numbers, it will not be possible to easily correlate which messages are the same and which ones are different because the sequence number would no longer uniquely identify a message.

To separate identification from sequencing and reliability, the following changes to syslog messages were made:

- The sequence number is retained as an identifier for the message. Messages with a lower number precede messages with a higher number, but they are not guaranteed to be consecutive.
- An additional field is added in the body portion of a syslog message to help ensure sequencing. The contents of this field contain a sequence number for a particular session. The same message transmitted over different sessions may have a different sequence number.

Syslog Session

A syslog session is a logical link from the syslog agent on a router to the recipient of a syslog message. For example, a syslog session can be established between a syslog agent and any of the following:

- Router console
- Router logging buffer
- Router monitor
- External syslog server

A syslog session runs over a transport connection between the syslog source and the syslog destination. A transport connection can use any of the following protocols:

- TCP
- UDP (association to one remote address and port)
- BEEP (channel within a BEEP session)

Figure 2 shows a mapping of syslog sessions and transport protocols between a router and a syslog server using an Open Systems Interconnection (OSI) model.

Note Figure 2 is best viewed using Internet Explorer.

Figure 2 Router to Syslog Server Mapping of Syslog Sessions and Transport Protocols

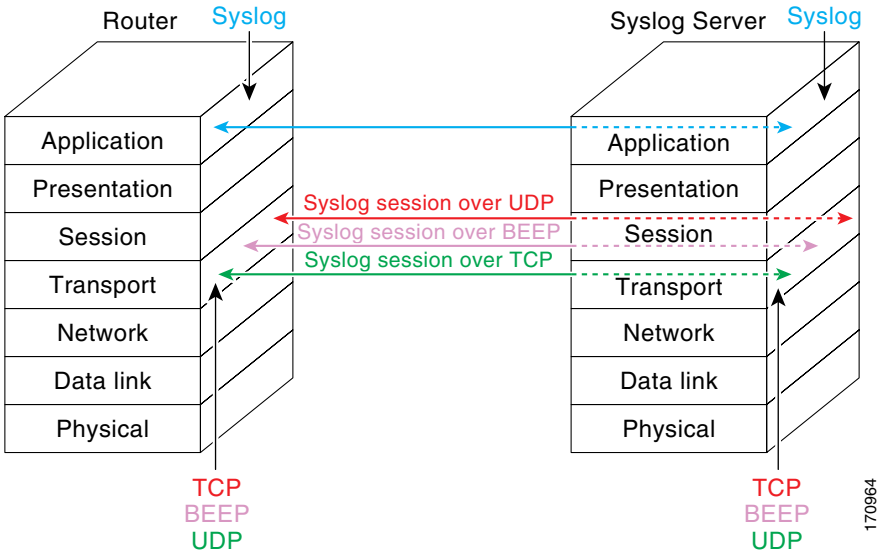
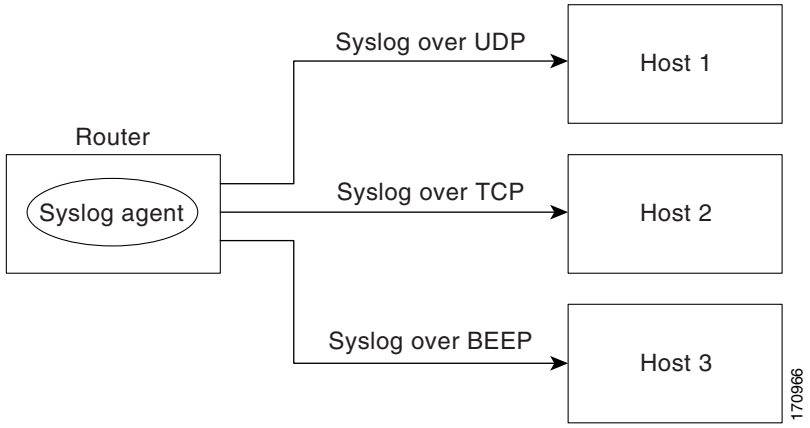


Figure 3 shows multiple syslog sessions from a single syslog agent to different hosts using UDP, TCP and BEEP.

Figure 3 Multiple Syslog Sessions from One Syslog Agent to Multiple Hosts

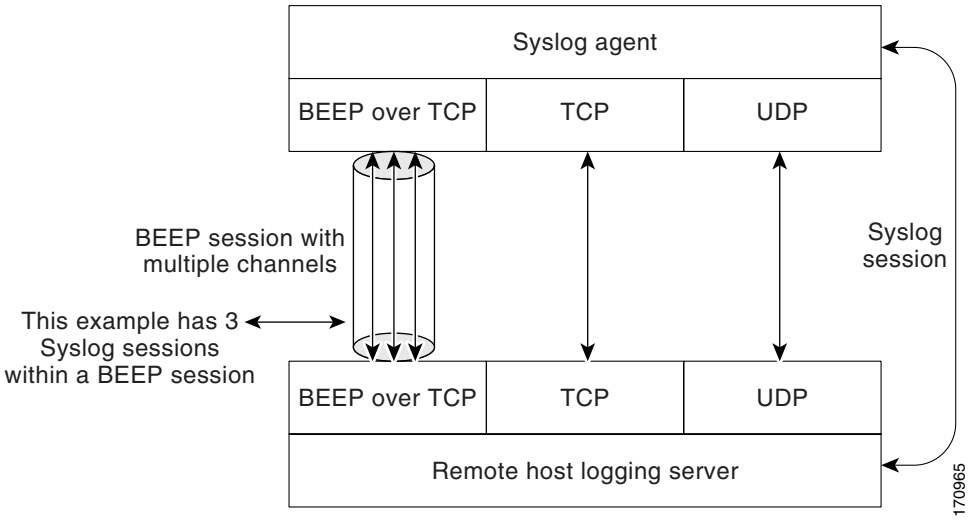


Multiple Syslog Sessions

A syslog session is independent of a transport connection. A Cisco router can support multiple syslog sessions, each running over its own transport connection. Multiple syslog sessions cannot share the same transport connection, but multiple syslog sessions may terminate at the same remote host, each running over its own transport connection. An example is a BEEP session in which multiple channels are used.

Figure 4 shows an end-to-end view of a syslog session. Note the three syslog sessions within a single BEEP session.

Figure 4 End-to-End View of a Syslog Session



The TCP and UDP protocols do not have multiplexed channels but the protocols do allow for using multiple ports to establish multiple syslog sessions to the same syslog host. To enable the UDP and TCP transport methods to have capability similar to BEEP's multiple channel capability, the Reliable Delivery and Filtering for Syslog feature allows multiple syslog sessions to be established via the UDP and TCP transport methods to the same logging host. Multiple syslog sessions going over BEEP sessions is also supported.

Message Discriminator

A message discriminator is a syslog processor. A message discriminator is associated with a syslog session and binds that session to a transport connection.

Prior to message delivery, the message is subject to the message discriminator with a user-specified list of criteria. After the first filtering criterion results in a message being blocked, the filtering check stops.

Note The sequence of criteria in the CLI does not affect the sequence in which criteria is checked.

- Following are filtering criteria. These criteria are checked in the order listed here:
  - Severity level or levels specified
  - Facility within the message body that matches a regular expression
  - Mnemonic that matches a regular expression
  - Part of the body of a message that matches a regular expression

A message discriminator offers the following capabilities:

- Optional rate limiting—Specifying a transmission rate of messages per time interval that is not to be exceeded. If the rate limit is exceeded, messages are either delayed or dropped, at the discretion of the device. The application of a rate limiter means that reliable delivery of syslog messages over that syslog session is no longer guaranteed. The purpose of a rate limiter is to avoid potential "flooding" at recipient syslog servers for applications that do not require guaranteed syslog delivery.

- Correlating—Inspecting candidate event messages and possibly aggregating information across events, creating a new event that contains the aggregated information. Correlating functions include:
  - Elimination of duplicate messages by maintaining a message count and waiting a specific time period between sending the first message of a certain type and sending the next message of that type
  - Elimination of oscillating messages
  - Simple message correlation; for example, if one message is a symptom of a cause reported by another message, one consolidated message is reported

A message discriminator can be associated with a specific destination and transport; that is, the filter can be host dependent. For this reason, a message discriminator is attached to a syslog session, transport, or channel, with possible device support for multiple sessions, transports, or channels, each of which can be attached to a different discriminator.

The establishment of a message discriminator should be separate from the establishment of a syslog session. A message discriminator should refer to the syslog session, transport, or channel to which it should be attached. The reasons for the separation are the following:

- Message discriminators can be managed separately from the connections, and refinements in the capabilities available to set up message discriminators need not affect how syslog sessions are set up and vice versa.
- Multiple connections can be attached to the same message discriminator, allowing for various syslog redundancy topologies.

When an explicit message discriminator is not associated with a syslog session, the generic message discriminator from the router-wide global settings is used. You can create an “empty” message discriminator without specifying attribute values (no rate limit and no filter configured).

Rate Limiting

The router-wide rate limiting capability in Cisco IOS syslog is preserved in the Reliable Delivery and Filtering for Syslog feature and is referred to as “global rate limiting.” If you do not use global rate limiting, all event messages are sent to remote syslog hosts if system resources can support the volume. When global rate limiting is set, it applies to all destinations. The value is set to the rate-limit attribute of the “generic message discriminator” if one has been set. The disadvantage of global rate limiting is that the rate limit of the least performing remote syslog host sets the rate for how fast a router can send out syslog messages.

The Reliable Delivery and Filtering for Syslog feature provides syslog session-based rate limiting to bypass the effects of global rate limiting. This session-based rate limiting is associated with a specific message discriminator and allows you to set the rate acceptance level independently for each syslog session.

Use of global rate limiting is not recommended when session-based rate limiting is in effect. A rate limit in a message discriminator specifies a not-to-exceed rate of syslog messages but does not guarantee that this rate will be reached. A configured global rate limit may cause messages on a session to be dropped even if the rate limit for that session has not been reached. These actions are important to understand if global rate limiting and session-based rate limiting are used concurrently.

Benefits of Reliable Delivery and Filtering for Syslog

- Authentication and encryption capabilities in BEEP provide reliable and secure delivery for syslog messages
- Multiple sessions to a single logging host independent of the underlying transport method
- Session-based message filtering and rate limiting
- Multiple connections can be attached to the same message discriminator, allowing various syslog redundancy topologies
- New CLI command to disable the default syslog count
- New CLI command to help identify relative positions of syslog messages that are dropped due to rate limiting

How to Configure Reliable Delivery and Filtering for Syslog

To configure Reliable Delivery and Filtering for Syslog, perform the following tasks:

- Creating a Message Discriminator, page 8
- Associating a Message Discriminator with a Logging Buffer, page 9
- Associating a Message Discriminator with a Console Terminal, page 10
- Associating a Message Discriminator with Terminal Lines, page 11
- Enabling Message Counters, page 12
- Adding and Removing a BEEP Session, page 13

Creating a Message Discriminator

Perform this task to create a message discriminator for syslog messages.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **logging discriminator** *discr-name* [[**facility**] [**mnemonics**] [**msg-body**] {**drops** *string* | **includes** *string*}] [**severity** {**drops** *sev-num* | **includes** *sev-num*}] [**rate-limit** *msglimit*]
4. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**logging discriminator** *discr-name* [[**facility**] [**mnemonics**] [**msg-body**] {**drops** *string* | **includes** *string*}] [**severity** {**drops** *sev-num* | **includes** *sev-num*}] [**rate-limit** *msglimit*] **Example:** `Router(config)# logging discriminator pacfltr1 facility includes facl357`	Creates a message discriminator with a facility subfilter. In this example, all messages with "facl357" in the facility field will be delivered.
Step 4	**end** **Example:** `Router(config)# end`	Returns the CLI to privileged EXEC mode.

Associating a Message Discriminator with a Logging Buffer

Perform this task to associate a message discriminator with a specific buffer.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **logging discriminator** *discr-name* [[**facility**] [**mnemonics**] [**msg-body**] {**drops** *string* | **includes** *string*}] [**severity** {**drops** *sev-num* | **includes** *sev-num*}] [**rate-limit** *msglimit*]
4. **logging buffered** [**discriminator** *discr-name* | **xml**] [*buffer-size*] [*severity-level*]
5. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**logging discriminator** *discr-name* [[**facility**] [**mnemonics**] [**msg-body**] {**drops** *string* | **includes** *string*}] [**severity** {**drops** *sev-num* | **includes** *sev-num*}] [**rate-limit** *msglimit*] **Example:** `Router(config)# logging discriminator pacfltr2`	Creates a message discriminator.
Step 4	**logging buffered** [**discriminator** *discr-name* | **xml**] [*buffer-size*] [*severity-level*] **Example:** `Router(config)# logging buffered discriminator pacfltr2 5`	Enables logging to a local buffer and specifies a message discriminator.
Step 5	**end** **Example:** `Router(config)# end`	Returns the CLI to privileged EXEC mode.

Associating a Message Discriminator with a Console Terminal

Perform this task to associate a message discriminator with a console terminal.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **logging discriminator** *discr-name* [[**facility**] [**mnemonics**] [**msg-body**] {**drops** *string* | **includes** *string*}] [**severity** {**drops** *sev-num* | **includes** *sev-num*}] [**rate-limit** *msglimit*]
4. **logging console** [**discriminator** *discr-name* | **xml**] [*severity-level*]
5. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**logging discriminator** *discr-name* [[**facility**] [**mnemonics**] [**msg-body**] {**drops** *string* | **includes** *string*}] [**severity** {**drops** *sev-num* | **includes** *sev-num*}] [**rate-limit** *msglimit*] **Example:** `Router(config)# logging discriminator pacfltr3`	Creates a message discriminator.
Step 4	**logging console** [**discriminator** *discr-name* | **xml**] [*severity-level*] Example: `Router(config)# logging console discriminator pacfltr3 1`	Enables logging to the console and specifies a message discriminator filtering messages at a specific severity level.
Step 5	**end** **Example:** `Router(config)# end`	Returns the CLI to privileged EXEC mode.

Associating a Message Discriminator with Terminal Lines

Perform this task to associate a message discriminator with terminal lines and have messages display at a monitor.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **logging discriminator** *discr-name* [[**facility**] [**mnemonics**] [**msg-body**] {**drops** *string* | **includes** *string*}] [**severity** {**drops** *sev-num* | **includes** *sev-num*}] [**rate-limit** *msglimit*]
4. **logging monitor** [**discriminator** *discr-name* | **xml**] [*severity-level*]
5. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**logging discriminator** *discr-name* [[**facility**] [**mnemonics**] [**msg-body**] {**drops** *string* | **includes** *string*}] [**severity** {**drops** *sev-num* | **includes** *sev-num*}] [**rate-limit** *msglimit*] **Example:** `Router(config)# logging discriminator pacfltr4`	Creates a message discriminator.
Step 4	**logging monitor** [**discriminator** *discr-name* | **xml**] [*severity-level*] **Example:** `Router(config)# logging monitor discriminator pacfltr4 2`	Specifies a message discriminator named pacfltr4 and enables logging to the terminal lines of messages at severity level 2 and lower.
Step 5	**end** **Example:** `Router(config)# end`	Returns the CLI to privileged EXEC mode.

Enabling Message Counters

Perform this task to enable logging of debug, log, or syslog messages.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **logging message-counter** {**debug** | **log** | **syslog**}
4. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**logging message-counter** {**debug** | **log** | **syslog**} **Example:** `Router(config)# logging message-counter syslog`	Enables logging of syslog messages.
Step 4	**end** **Example:** `Router(config)# end`	Returns the CLI to privileged EXEC mode.

Adding and Removing a BEEP Session

Perform this task to add and remove a BEEP session.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **logging host** {{*ip-address* | *hostname*} [**vrf** *vrf-name*] | **ipv6** {*ipv6-address* | *hostname*}} [**discriminator** *discr-name* | [[**filtered** [**stream** *stream-id*] | **xml**]] [**transport** {[**beep** [**audit**] [**channel** *chnl-number*] [**sasl** *profile-name*] [**tls cipher** [*cipher-num*] **trustpoint** *trustpt-name*]]] | **tcp** [**audit**] | **udp**} [**port** *port-num*]] [**sequence-num-session**] [**session-id** {**hostname** | **ipv4** | **ipv6** | **string** *custom-string*}]
4. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**logging host** {{*ip-address* | *hostname*} [**vrf** *vrf-name*] | **ipv6** {*ipv6-address* | *hostname*}} [**discriminator** *discr-name* | [[**filtered** [**stream** *stream-id*] | **xml**]] [**transport** {[**beep** [**audit**] [**channel** *chnl-number*] [**sasl** *profile-name*] [**tls cipher** [*cipher-num*] **trustpoint** *trustpt-name*]]] | **tcp** [**audit**] | **udp**} [**port** *port-num*]] [**sequence-num-session**] [**session-id** {**hostname** | **ipv4** | **ipv6** | **string** *custom-string*}] **Example:** `Router(config)# logging host host3 transport beep port 600 channel 3`	Identifies a logging host and specifies the transport protocol, port, and channel for logging messages.
Step 4	**end** **Example:** `Router(config)# end`	Returns the CLI to privileged EXEC mode.

Configuration Examples for Reliable Delivery and Filtering for Syslog

This section provides the following configuration example:

- Configuring Transport and Logging: Example, page 14

Configuring Transport and Logging: Example

```
Router(config)# show running-config | include logging

logging buffered xml
 logging
 logging
logging host 209.165.201.1 transport udp port 601
 logging synchronous

Router(config)# logging host 209.165.201.1 transport beep port 600 channel 3
Router(config)# logging host 209.165.201.1 transport tcp port 602
Router(config)# show running-config | include logging

logging buffered xml
```

```
 logging
 logging
logging host 209.165.201.1 transport udp port 601
logging host 209.165.201.1 transport beep port 600 channel 3
logging host 209.165.201.1 transport tcp port 602
 logging synchronous
Router(config)#
```

Additional References

The following sections provide references related to the Reliable Delivery and Filtering for Syslog feature.

Related Documents

Related Topic	Document Title
Syslog logging	*Troubleshooting and Fault Management* module
Network Management commands (including logging commands): complete command syntax, defaults, command mode, command history, usage guidelines, and examples	*Cisco IOS Network Management Command Reference*

Standards

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

MIBs

MIB	MIBs Link
No new or modified MIBs are supported by this feature, and support for existing MIBs has not been modified by this feature.	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
RFC 3195	*Reliable Delivery for Syslog*
RFC 3081, section 3.1.4	*Mapping the BEEP Core onto TCP,* "Use of Flow Control"
RFC 3164	*The BSD Syslog Protocol*

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/techsupport

Command Reference

The following commands are introduced or modified in the feature or features documented in this module. For information about these commands, see the *Cisco IOS Network Management Command Reference* at http://www.cisco.com/en/US/docs/ios/netmgmt/command/reference/nm_book.html. For information about all Cisco IOS commands, go to the Command Lookup Tool at http://tools.cisco.com/Support/CLILookup or to the *Cisco IOS Master Commands List*.

- **logging buffered**
- **logging console**
- **logging discriminator**
- **logging host**
- **logging message-counter**
- **logging monitor**
- **show logging**

Feature Information for Reliable Delivery and Filtering for Syslog

Table 1 lists the release history for this feature.

Not all commands may be available in your Cisco IOS software release. For release information about a specific command, see the command reference documentation.

Use Cisco Feature Navigator to find information about platform support and software image support. Cisco Feature Navigator enables you to determine which Cisco IOS, Catalyst OS, and Cisco IOS XE software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Note Table 1 lists only the Cisco IOS software release that introduced support for a given feature in a given Cisco IOS software release train. Unless noted otherwise, subsequent releases of that Cisco IOS software release train also support that feature.

Table 1 Feature Information for Reliable Delivery and Filtering for Syslog

Feature Name	Releases	Feature Information
Reliable Delivery and Filtering for Syslog	12.4(11)T 12.2(33)SRB 12.2(33)SB Cisco IOS XE Release 2.1 12.2(33)SXI	The Reliable Delivery and Filtering for Syslog feature allows a device to be customized for receipt of syslog messages. This feature provides for reliable and secure delivery for syslog messages using BEEP. Additionally it allows multiple sessions to a single logging host, independent of the underlying transport method, and provides a filtering mechanism called a message discriminator. In Cisco IOS XE Release 2.1, this feature was introduced on Cisco ASR 1000 Series Routers. The following commands were introduced or modified: **logging buffered**, **logging console**, **logging discriminator**, **logging host**, **logging message-counter**, **logging monitor**, **show logging**.

HTTP Services

HTTP 1.1 Web Server and Client

First Published: November 20, 2006
Last Updated: September 07, 2009

The HTTP 1.1 Web Server and Client feature provides a consistent interface for users and applications by implementing support for HTTP 1.1 in Cisco IOS software-based devices. When combined with the HTTPS feature, the HTTP 1.1 Web Server and Client feature provides a complete, secure solution for HTTP services between Cisco devices.

This module describes the concepts and the tasks related to configuring the HTTP 1.1 Web Server and Client feature.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the "Feature Information for the HTTP 1.1 Web Server and Client" section on page 11.

Use Cisco Feature Navigator to find information about platform support and Cisco IOS and Catalyst OS software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Contents

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

Information About the HTTP 1.1 Web Server and Client

This feature updates the Cisco implementation of the Hypertext Transfer Protocol (HTTP) from 1.0 to 1.1. The HTTP server allows features and applications, such as the Cisco web browser user interface, to be run on your routing device.

The Cisco implementation of HTTP 1.1 is backward-compatible with previous Cisco IOS releases. If you are currently using configurations that enable the HTTP server, no configuration changes are needed, as all defaults remain the same.

The process of enabling and configuring the HTTP server also remains the same as in previous releases. Support for Server Side Includes (SSIs) and HTML forms has not changed. Additional configuration options, in the form of the **ip http timeout-policy** command and the **ip http max-connections** command, have been added. These options allow configurable resource limits for the HTTP server. If you do not use these optional commands, the default policies are used.

Remote applications may require that you enable the HTTP server before using them. Applications that use the HTTP server include:

- Cisco web browser user interface, which uses the Cisco IOS Homepage Server, HTTP-based EXEC Server, and HTTP IOS File System (IFS) Server
- VPN Device Manager (VDM) application, which uses the VDM Server and the XML Session Manager (XSM)
- QoS Device Manager (QDM) application, which uses the QDM Server
- IP Phone and Cisco IOS Telephony Service applications, which use the ITS Local Directory Search and IOS Telephony Server (ITS)

No Cisco applications use the HTTP Client in Cisco IOS Release 12.2(15)T.

About HTTP Server General Access Policies

The **ip http timeout-policy** command allows you to specify general access characteristics for the server by configuring a value for idle time, connection life, and request maximum. By adjusting these values you can configure a general policy; for example, if you want to maximize throughput for HTTP connections, you should configure a policy that minimizes connection overhead. You can configure this type of policy by specifying large values for the **life** and **request** options so that each connection stays open longer and more requests are processed for each connection.

Another example would be to configure a policy that minimizes the response time for new connections. You can configure this type of policy by specifying small values for the **life** and **request** options so that the connections are quickly released to serve new clients.

A throughput policy would be better for HTTP sessions with dedicated management applications, as it would allow the application to send more requests before the connection is closed, while a response time policy would be better for interactive HTTP sessions, as it would allow more people to connect to the server at the same time without having to wait for connections to become available.

In general, you should configure these options as appropriate for your environment. The value for the **idle** option should be balanced so that it is large enough not to cause an unwanted request or response timeout on the connection, but small enough that it does not hold a connection open longer than necessary.

Access security policies for the HTTP server are configured using the **ip http authentication** command, which allows only selective users to access the server, the **ip http access-class** command, which allows only selective IP hosts to access the server, and the **ip http accounting commands** command, which specifies a particular command accounting method for HTTP server users.

How to Configure HTTP 1.1 Web Server and Client

This section contains the following tasks:

Configuring the HTTP 1.1 Web Server

Perform this task to enable the HTTP server and configure optional server characteristics. The HTTP server is disabled by default.

Note If you want to configure authentication (step 4), you must configure the authentication type before you begin configuring the HTTP 1.1 web server.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip http server**
4. **ip http authentication {aaa | enable | local | tacacs} (optional)**
5. **ip http accounting commands** *level* {**default** | *named-accounting-method-list*} (optional)
6. **ip http port** *port-number* (optional)
7. **ip http path** *url* (optional)
8. **ip http access-class** *access-list-number* (optional)
9. **ip http max-connections** *value* (optional)
10. **ip http timeout-policy idle** *seconds* **life** *seconds* **requests** *value* (optional)

DETAILED STEPS

	Command or Action	Purpose

Step 1	`enable`
	
	Example:
	`Router> enable`
		Enables privileged EXEC mode.
		• Enter your password if prompted.

Step 2	`configure terminal`
	
	Example:
	`Router# configure terminal`
		Enters global configuration mode.

	Command or Action	Purpose
Step 3	**ip http server** **Example:** `Router(config)# ip http server`	Enables the HTTP 1.1 server, including the Cisco web browser user interface. **Note** If you are enabling the HTTP over Secure Socket Layer (HTTPS) server using the **ip http secure-server** command, you should disable the standard HTTP server using the **no ip http server** command. This command is required to ensure only secure connections to the server.
Step 4	**ip http authentication** {**aaa** | **enable** | **local** | **tacacs**} **Example:** `Router(config)# ip http authentication local`	(Optional) Specifies the authentication method to be used for login when a client connects to the HTTP server. The methods for authentication are: **aaa**—Indicates that the authentication method used for the AAA login service (specified by the **aaa authentication login default** command) should be used for authentication. **enable**—Indicates that the "enable" password should be used for authentication. (This is the default method.) **local** —Indicates that the login user name, password and privilege level access combination specified in the local system configuration (by the **username** global configuration command) should be used for authentication and authorization. **tacacs**—Indicates that the TACACS (or XTACACS) server should be used for authentication.
Step 5	**ip http accounting commands** *level* {**default** | *named-accounting-method-list*} **Example:** `Router(config)# ip http accounting commands 15 default`	(Optional) Specifies a particular command accounting method for HTTP server users. Command accounting for HTTP and HTTPS is automatically enabled when authentication, authorization, and accounting (AAA) is configured on the device. It is not possible to disable accounting for HTTP and HTTPS. HTTP and HTTPS will default to using the global AAA default method list for accounting. The CLI can be used to configure HTTP and HTTPS to use any predefined AAA method list. *level*—Valid privilege level entries are integers from 0 to 15. **default**—Indicates the **default** accounting method list configured by the aaa accounting commands CLI. *named-accounting-method-list*—Indicates the name of the predefined command accounting method list.
Step 6	**ip http port** *port-number* **Example:** `Router(config)# ip http port 8080`	(Optional) Specifies the server port that should be used for HTTP communication (for example, for the Cisco web browser user interface).

	Command or Action	Purpose
Step 7	**ip http path** *url* **Example:** `Router(config)# ip http path slot1:`	(Optional) Sets the base HTTP path for HTML files. The base path is used to specify the location of the HTTP server files (HTML files) on the local system. Generally, the HTML files are located in system flash memory.
Step 8	**ip http access-class** *access-list-number* **Example:** `Router(config)# ip http access-class 20`	(Optional) Specifies the access list that should be used to allow access to the HTTP server.
Step 9	**ip http max-connections** *value* **Example:** `Router(config)# ip http max-connections 10`	(Optional) Sets the maximum number of concurrent connections to the HTTP sever that will be allowed. The default value is 5.
Step 10	**ip http timeout-policy idle** *seconds* **life** *seconds* **requests** *value* **Example:** `Router(config)# ip http timeout-policy idle 30 life 120 requests 100`	(Optional) Sets the characteristics that determine how long a connection to the HTTP server should remain open. The characteristics are: **idle**—The maximum number of seconds the connection will be kept open if no data is received or response data cannot be sent out on the connection. Note that a new value may not take effect on any already existing connections. If the server is too busy or the limit on the **life** time or the number of **requests** is reached, the connection may be closed sooner. The default value is 180 seconds (3 minutes). **life**—The maximum number of seconds the connection will be kept open, from the time the connection is established. Note that the new value may not take effect on any already existing connections. If the server is too busy or the limit on the idle time or the number of requests is reached, it may close the connection sooner. Also, since the server will not close the connection while actively processing a request, the connection may remain open longer than the specified **life** time if processing is occurring when the life maximum is reached. In this case, the connection will be closed when processing finishes. The default value is 180 seconds (3 minutes). The maximum value is 86400 seconds (24 hours). **requests**—The maximum limit on the number of requests processed on a persistent connection before it is closed. Note that the new value may not take effect on already existing connections. If the server is too busy or the limit on the idle time or the life time is reached, the connection may be closed before the maximum number of requests are processed. The default value is 1. The maximum value is 86400.

Configuring the HTTP Client

Perform this task to enable the HTTP client and configure optional client characteristics.

The standard HTTP 1.1 client and the secure HTTP client are always enabled. No commands exist to disable the HTTP client. For information about configuring optional characteristics for the HTTPS client, see the *HTTPS-HTTP Server and Client with SSL 3.0*, Release 12.2(15)T, feature module.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip http client cache** {**ager interval** *minutes* | **memory** {**file** *file-size-limit* | **pool** *pool-size-limit*}
4. **ip http client connection** {**forceclose** | **idle timeout** *seconds* | **retry** *count* | **timeout** *seconds*}
5. **ip http client password** *password*
6. **ip http client proxy-server** *proxy-name* **proxy-port** *port-number*
7. **ip http client response timeout** *seconds*
8. **ip http client source-interface** *type number*
9. **ip http client username** *username*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**ip http client cache** {**ager interval** *minutes* | **memory** {**file** *file-size-limit* | **pool** *pool-size-limit*} **Example:** `Router(config)# ip http client cache memory file 5`	Configures HTTP client cache.
Step 4	**ip http client connection** {**forceclose** | **idle timeout** *seconds* | **retry** *count* | **timeout** *seconds*} **Example:** `Router(config)# ip http client connection timeout 10`	Configures an HTTP client connection.

	Command or Action	Purpose
Step 5	**ip http client password** *password* **Example:** `Router(config)# ip http client password pswd1`	Configures the default password used for connections to remote HTTP servers.
Step 6	**ip http client proxy-server** *proxy-name* **proxy-port** *port-number* **Example:** `Router(config)# ip http client proxy-server server1 proxy-port 52`	Configures an HTTP proxy server.
Step 7	**ip http client response timeout** *seconds* **Example:** `Router(config)# ip http client response timeout 60`	Specifies the timeout value, in seconds, that the HTTP client waits for a response from the server.
Step 8	**ip http client source-interface** *type number* **Example:** `Router(config)# ip http client source-interface ethernet1/0`	Configures a source interface for the HTTP client.
Step 9	**ip http client username** *username* **Example:** `Router(config)# ip http client user1`	Configures the default username used for connections to remote HTTP servers.

Configuration Examples for HTTP 1.1 Web Server

This section provides the following configuration examples:

- Configuring the HTTP 1.1 Web Server: Example, page 7
- Verifying HTTP Connectivity, page 8

Configuring the HTTP 1.1 Web Server: Example

The following example shows a typical configuration that enables the server and sets some of the characteristics:

```
ip http server
ip http authentication aaa
ip http accounting commands 15 default
ip http path flash:
ip access-list standard 20
 permit 209.165.202.130 0.0.0.255
 permit 209.165.201.1 0.0.255.255
 permit 209.165.200.225 0.255.255.255
! (Note: all other access implicitly denied)
 end
ip http access-class 10
ip http max-connections 10
```

```
ip http accounting commands 1 oneacct
```

In the following example, a Throughput timeout policy is applied. This configuration would allow each connection to be idle a maximum of 30 seconds (approximately). Each connection will remain open (be "alive") until either the HTTP server has been busy processing requests for approximately 2 minutes (120 seconds) or until approximately100 requests have been processed.

```
ip http timeout-policy idle 30 life 120 requests 100
```

In the following example, a Response Time timeout policy is applied. This configuration would allow each connection to be idle a maximum of 30 seconds (approximately). Each connection will be closed as soon as the first request has been processed.

```
ip http timeout-policy idle 30 life 30 requests 1
```

Verifying HTTP Connectivity

To verify remote connectivity to the HTTP server, enter the system IP address in a web browser, followed by a colon and the appropriate port number (80 is the default port number).

For example, if the system IP address is 209.165.202.129 and the port number is 8080, enter **http://209.165.202.129:8080** as the URL in a web browser.

If HTTP authentication is configured, a login dialog box will appear. Enter the appropriate username and password. If the default login authentication method of "enable" is configured, you may leave the username field blank, and use the "enable" password to log in.

The system home page should appear in your browser.

Where to Go Next

For information about secure HTTP connections using Secure Sockets Layer (SSL) 3.0, refer to the *HTTPS - HTTP with SSL 3.0*, Release 12.2(15)T, feature module at:
http://www.cisco.com/en/US/docs/ios/12_2t/12_2t15/feature/guide/ftsslsht.html

Additional References

The following sections provide references related to the HTTP 1.1 Web Server and Client.

Related Documents

Related Topic	Document Title
HTTPS	• *HTTPS—HTTP with SSL 3.0* feature module • *Firewall Support of HTTPS Authentication Proxy* feature module
HTTP commands	*Cisco IOS Network Management Command Reference*

Standards

Standard	Title
No specific standards are supported by this feature. Note that HTTP 1.1, as defined in RFC 2616, is currently classified as a “Standards Track” document by the IETF.	—

MIBs

MIB	MIBs Link
• No specific MIBs are supported for this feature.	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC[1]	Title
RFC 2616	*Hypertext Transfer Protocol -- HTTP/1.1*

1. Not all supported RFCs are listed.

The Cisco implementation of the HTTP version 1.1 supports a subset of elements defined in RFC 2616. Following is a list of supported RFC 2616 headers:

- Allow (Only GET, HEAD, and POST methods are supported)
- Authorization, WWW-Authenticate - Basic authentication only
- Cache-control
- Chunked Transfer Encoding
- Connection close

- Content-Encoding
- Content-Language
- Content-Length
- Content-Type
- Date, Expires
- Location

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/techsupport

Feature Information for the HTTP 1.1 Web Server and Client

Table 1 lists the features in this module and provides links to specific configuration information. Only features that were introduced or modified in Cisco IOS Release 12.4T or Cisco IOS Release 12.2SR or a later release appear in the table.

Not all commands may be available in your Cisco IOS software release. For release information about a specific command, see the command reference documentation.

Use Cisco Feature Navigator to find information about platform support and software image support. Cisco Feature Navigator enables you to determine which Cisco IOS and Catalyst OS software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Note Table 1 lists only the Cisco IOS software release that introduced support for a given feature in a given Cisco IOS software release train. Unless noted otherwise, subsequent releases of that Cisco IOS software release train also support that feature.

Table 1 Feature Information for HTTP 1.1 Web Server and Client

Feature Name	Releases	Feature Information
HTTP 1.1 Web Server and Client	12.2(15)T 12.4(15)T 12.2(33)SRC 12.2(33)SB	The HTTP 1.1 Web Server and Client feature provides a consistent interface for users and applications by implementing support for HTTP 1.1 in Cisco IOS software-based devices. When combined with the HTTPS feature, the HTTP 1.1 Web Server and Client feature provides a complete, secure solution for HTTP services between Cisco devices. This entire module provides information about this feature. The following commands were introduced or modified by this feature: **debug ip http all**, **debug ip http client**, **ip http access-class**, **ip http authentication**, **ip http client cache**, **ip http client connection**, **ip http client password**, **ip http client proxy-server**, **ip http client response timeout**, **ip http client source-interface**, **ip http client username**, **ip http max-connections**, **ip http path**, **ip http port**, **ip http server**, **ip http timeout-policy**, **show ip http client**, **show ip http client connection**, **show ip http client history**, **show ip http client session-module**, **show ip http server**, **show ip http server secure status**.
HTTP TACAC+ Accounting Support	12.4(15)T 12.2(33)SRC 12.2(33)SB	The HTTP TACAC+ Accounting Support feature introduces the **ip http accounting commands** command. This command is used to specify a particular command accounting method for HTTP server users. Command accounting provides information about the commands for a specified privilege level that are being executed on a device. Each command accounting record corresponds to one IOS command executed at its respective privilege level, as well as the date and time the command was executed, and the user who executed it. The following sections provide information about this feature: • Configuring the HTTP 1.1 Web Server, page 3 The following commands were introduced or modified by this feature: **ip http accounting commands**.

HTTPS—HTTP Server and Client with SSL 3.0

First Published: March 31, 2003
Last Updated: September 07, 2009

The HTTPS—HTTP Server and Client with SSL 3.0 feature provides Secure Socket Layer (SSL) version 3.0 support for the HTTP 1.1 server and HTTP 1.1 client within Cisco IOS software. SSL provides server authentication, encryption, and message integrity to allow secure HTTP communications. SSL also provides HTTP client authentication. HTTP over SSL is abbreviated as HTTPS.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the "Feature Information for HTTPS—HTTP Server and Client with SSL 3.0" section on page 16.

Use Cisco Feature Navigator to find information about platform support and Cisco IOS and Catalyst OS software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Contents

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

Prerequisites for HTTPS—HTTP Server and Client with SSL 3.0

To enable secure HTTP connections (encryption) without a configured certificate authority trustpoint, you must first ensure that each device has the key (such as a Rivest, Shamir, and Adleman [RSA] public key or a shared key) of the other device. In most cases, an RSA key pair will be generated automatically. The RSA key pair is used for creating a self-signed certificate (which is also generated automatically).

Restrictions for HTTPS—HTTP Server and Client with SSL 3.0

The HTTPS—HTTP Server and Client with SSL 3.0 feature is available only in Cisco IOS software images that support SSL. SSL is supported in "IPSec 56" (contains "k8" in the image name) and "IPSec 3DES" images (contains "k9" in the image name). "IPSec 56" images provide up to 64-bit encryption, "IPSec 3 DES" images provide greater than 64-bit encryption. The following CipherSuites are supported in IPSec Data Encryption Standard (DES) images:

- SSL_RSA_WITH_RC4_128_MD5—RSA key exchange (RSA Public Key Cryptography) with RC4 128-bit encryption and message digest algorithm 5 (MD5) for message digest
- SSL_RSA_WITH_RC4_128_SHA—RSA key exchange with RC4 128-bit encryption and Secure Hash Algorithm (SHA) for message digest
- SSL_RSA_WITH_3DES_EDE_CBC_SHA—RSA key exchange with 3DES and DES-EDE3-CBC for message encryption and SHA for message digest
- SSL_RSA_WITH_DES_CBC_SHA—RSA key exchange with DES-CBC for message encryption and SHA for message digest

For IPSec 56 images, only the SSL_RSA_WITH_DES_CBC_SHA CipherSuite is supported. For further details on these CipherSuites, see the *SSL Protocol Version 3.0* Internet-Draft document (see the "Additional References" section on page 14).

RSA (in conjunction with the specified encryption and digest algorithm combinations) is used for both key generation and authentication on SSL connections. This usage is independent of whether a certificate authority (CA) trustpoint is configured.

Information About HTTPS—HTTP Server and Client with SSL 3.0

To configure the HTTP with SSL 3.0 (HTTPS) feature, you should understand the following concepts:

Secure HTTP Server and Secure HTTP Client

A secure HTTP connection means that data sent to and received from an HTTP server are encrypted before being sent out over the Internet. HTTP with SSL encryption provides a secure connection to allow such functions as configuring a router from a web browser. Cisco's implementation of the secure HTTP

server and secure HTTP client uses an implementation of the SSL version 3.0. Application layer encryption provides an alternative to older methods such as having to set up a tunnel to the HTTP server for remote management. HTTP over SSL is abbreviated as HTTPS; the URL of a secure connection will begin with https:// instead of http://.

The Cisco IOS HTTP secure server's primary role is to listen for HTTPS requests on a designated port (the default HTTPS port is 443) and to pass the request to the HTTP 1.1 web server. The HTTP 1.1 server processes requests and passes responses (served pages) back to the HTTP secure server, which, in turn, responds to the original request.

The Cisco IOS HTTP secure client's primary role is to respond to Cisco IOS application requests for HTTPS User Agent services, perform HTTPS User Agent services on the application's behalf, and pass the response back to the application.

Certificate Authority Trustpoints

Certificate authorities (CAs) are responsible for managing certificate requests and issuing certificates to participating IPSec network devices. These services provide centralized security key and certificate management for the participating devices. Specific CA servers are referred to as "trustpoints."

The HTTPS server provides a secure connection by providing a certified X.509v3 certificate to the client when a connection attempt is made. The certified X.509v3 certificate is obtained from a specified CA trustpoint. The client (usually a web browser), in turn, has a public key that allows it to authenticate the certificate.

Configuring a CA trustpoint is highly recommended for secure HTTP connections. However, if a CA trustpoint is not configured for the routing device running the HTTPS server, the server will certify itself and generate the needed RSA key pair. Because a self-certified (self-signed) certificate does not provide adequate security, the connecting client will generate a notification that the certificate is self-certified, and the user will have the opportunity to accept or reject the connection. This option is available for internal network topologies (such as testing).

The HTTPS—HTTP Server and Client with SSL 3.0 feature also provides an optional command (**ip http secure-client-auth**) that, when enabled, has the HTTPS server request an X.509v3 certificate from the client. Authenticating the client provides more security than server authentication by itself.

For additional information on certificate authorities, see the "Configuring Certification Authority Interoperability" chapter in the *Cisco IOS Security Configuration Guide.*

CipherSuites

A CipherSuite specifies the encryption algorithm and digest algorithm to use on an SSL connection. Web browsers offer a list of supported CipherSuites when connecting to the HTTPS server, and the client and server will negotiate the best encryption algorithm to use from those that are supported by both. For example, Netscape Communicator 4.76 supports U.S. security with RSA Public Key Cryptography, MD2, MD5, RC2-CBC, RC4, DES-CBC, and DES-EDE3-CBC.

For the best possible encryption, you should use a browser that supports 128-bit encryption, such as Microsoft Internet Explorer version 5.5 (or later), or Netscape Communicator version 4.76 (or later). The SSL_RSA_WITH_DES_CBC_SHA CipherSuite provides less security than the other CipherSuites, because it does not offer 128-bit encryption.

In terms of router processing load (speed), the following list ranks the CipherSuites from fastest to slowest (slightly more processing time is required for the more secure and more complex CipherSuites):

1. SSL_RSA_WITH_DES_CBC_SHA

2. SSL_RSA_WITH_RC4_128_MD5
3. SSL_RSA_WITH_RC4_128_SHA
4. SSL_RSA_WITH_3DES_EDE_CBC_SHA

How to Configure the HTTPS—HTTP Server and Client with SSL 3.0

To configure the HTTPS—HTTP Server and Client with SSL 3.0 feature, complete the procedures in the following sections:

Declaring a Certificate Authority Trustpoint

Configuring a CA trustpoint is highly recommended for secure HTTP connections. The certified X.509v3 certificate for the secure HTTP server (or client) is obtained from the specified CA trustpoint. If you do not declare a CA trustpoint, then a self-signed certificate will be used for secure HTTP connections. The self-signed certificate is generated automatically.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **hostname** *name*
4. **ip domain-name** *name ip domain name*
5. **crypto key generate rsa usage-keys**
6. **crypto ca trustpoint** *name*
7. **enrollment url** *url*
8. **enrollment http-proxy** *host-name port-number*
9. **crl** {**query** *url* | **optional** | **best-effort**}
10. **primary**
11. **exit**
12. **crypto ca authenticate** *name*
13. **crypto ca enrollment** *name*
14. **copy running-config startup-config**
    or
    copy system:running-config nvram:startup-config

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**hostname** *name* **Example:** `Router(config)# hostname Router`	Specifies the hostname of the router. • This step is needed only if you have not previously configured a hostname for your router. The hostname is required because a fully qualified domain name is needed for security keys and certificates.
Step 4	**ip domain-name** *name* **Example:** `Router(config)# ip domain-name example.com`	Specifies the IP domain name of the router. • This step is needed only if you have not previously configured an IP domain name for your router. The domain name is required because a fully qualified domain name is needed for security keys and certificates.
Step 5	**crypto key generate rsa usage-keys** **Example:** `Router(config)# crypto key generate rsa usage-keys`	(Optional) Generates an RSA key pair. • The **usage-keys** keyword specifies that two RSA special-usage key pairs should be generated (that is, one encryption pair and one signature pair) instead of one general-purpose key pair. • RSA key pairs are used to sign and encrypt Internet key exchange (IKE) key management messages and are required before you can obtain a certificate for your router. • RSA key pairs are generated automatically. This command can be used to regenerate the keys, if needed. **Note** There are other keywords and arguments for this command, but they do not pertain to this feature.
Step 6	**crypto ca trustpoint** *name* **Example:** `Router(config)# crypto ca trustpoint TP1`	Specifies a local configuration name for the CA trustpoint and enters CA trustpoint configuration mode. **Note** The **crypto ca identity** command was replaced by the **crypto ca trustpoint** command in Cisco IOS Release 12.2(8)T.
Step 7	**enrollment url** *url* **Example:** `Router(ca-trustpoint)# enrollment url http://example.com`	Specifies a URL of the CA where your router should send certificate requests. • If you are using Simple Certificate Enrollment Protocol (SCEP) for enrollment, the URL argument must be in the form **http://***CA-name*, where *CA-name* is the host Domain Name System (DNS) name or IP address of the CA trustpoint.

	Command or Action	Purpose
Step 8	**enrollment http-proxy** *host-name port-number* **Example:** `Router(ca-trustpoint)# enrollment http-proxy example.com 8080`	(Optional) Configures the router to obtain certificates from the CA through an HTTP proxy server.
Step 9	**crl** {**query** *url* | **optional** | **best-effort**} **Example:** `Router(ca-trustpoint)# crl query ldap://example.com`	Configures the router to request a certificate revocation list (CRL), make CRL checking optional, or perform CRL checking on a "best-effort" basis. • CRLs ensure that the certificate of the peer has not been revoked. • The **crl optional** command configures the router to accept certificates even if the appropriate CRL cannot be downloaded. • Use the **crl query** *url* command to specify the Lightweight Directory Access Protocol (LDAP) URL of the CA server; for example, **ldap://another-server**.
Step 10	**primary** **Example:** `Router(ca-trustpoint)# primary`	(Optional) Specifies that this trustpoint should be used as the primary (default) trustpoint for CA requests. • Use this command if more than one CA trustpoint will be configured on this router.
Step 11	**exit** **Example:** `Router(ca-trustpoint)# exit`	Exits CA trustpoint configuration mode and returns to global configuration mode.
Step 12	**crypto ca authenticate** *name* **Example:** `Router(config)# crypto ca authenticate TP1`	Authenticates the CA by getting the public key of the CA. • Use the same name that you used when declaring the CA in the **crypto ca trustpoint** command.
Step 13	**crypto ca enrollment** *name* **Example:** `Router(config)# crypto ca enrollment TP1`	Obtains the certificate from the specified CA trustpoint. • This command requests a signed certificate from the CA for each RSA key pair.
Step 14	**copy running-config startup-config** or **copy system:running-config nvram:startup-config** **Example:** `Router(config)# copy running-config startup-config`	Saves the configuration to NVRAM. • This command is required to save the certificates into NVRAM. If not used, the certificates would be lost at router reload. **Note** To execute EXEC mode commands in global configuration mode, you can add the **do** keyword before the command. For example, instead of **copy running-config startup-config**, you could enter **do copy running-config startup-config**.

Configuring the HTTPS Server with SSL 3.0

To disable the standard HTTP server and configure the HTTPS server with SSL 3.0, complete the procedure in this section.

Prerequisites

If a certificate authority is to be used for certification, you should declare the CA trustpoint on the routing device before enabling the secure HTTP server.

SUMMARY STEPS

1. **enable**
2. **show ip http server status**
3. **configure terminal**
4. **no ip http server**
5. **ip http secure-server**
6. **ip http secure-port** *port-number*
7. **ip http secure-ciphersuite** [**3des-ede-cbc-sha**] [**rc4-128-sha**] [**rc4-128-md5**] [**des-cbc-sha**]
8. **ip http secure-client-auth**
9. **ip http secure-trustpoint** *name*
10. **end**
11. **show ip http server secure status**

DETAILED STEPS

	Command or Action	Purpose

Step 1	`enable`
	
	Example:
	`Router> enable`
		Enables privileged EXEC mode.
		• Enter your password if prompted.

Step 2	`Router# show ip http server status` **Example:** `Router# show ip http server status`	(Optional) Displays the status of the HTTP server. • If you are unsure whether the secure HTTP server is supported in the software image you are running, enter this command and look for the line "HTTP secure server capability: {Present | Not present}". • This command displays the status of the standard HTTP server (enabled or disabled).

Step 3	`configure terminal`
	
	Example:
	`Router# configure terminal`
		Enters global configuration mode.

	Command or Action	Purpose
Step 4	**no ip http server** **Example:** `Router(config)# no ip http server`	Disables the standard HTTP server. **Note** When enabling the HTTPS server you should always disable the standard HTTP server to prevent insecure connections to the same services. This is a precautionary step (typically, the HTTP server is disabled by default).
Step 5	**ip http secure-server** **Example:** `Router(config)# ip http secure-server`	Enables the HTTPS server.
Step 6	**ip http secure-port** *port-number* **Example:** `Router(config)# ip http secure-port 1025`	(Optional) Specifies the port number that should be used for the HTTPS server. The default port number is 443. Valid options are 443 or any number in the range 1025 to 65535.
Step 7	**ip http secure-ciphersuite** [**3des-ede-cbc-sha**] [**rc4-128-sha**] [**rc4-128-md5**] [**des-cbc-sha**] **Example:** `Router(config)# ip http secure-ciphersuite rc4-128-sha rc4-128-md5`	(Optional) Specifies the CipherSuites (encryption algorithms) that should be used for encryption over the HTTPS connection. • This command allows you to restrict the list of CipherSuites that the server offers the connecting clients. For example, you may want to allow only the most secure CipherSuite to be used. • Unless you have a reason to specify the CipherSuites that should be used, or you are unfamiliar with the details of these CipherSuites, you should leave this command unconfigured and let the server and client negotiate the CipherSuite that they both support (this is the default).
Step 8	**ip http secure-client-auth** **Example:** `Router(config)# ip http secure-client-auth`	(Optional) Configures the HTTP server to request an X.509v3 certificate from the client in order to authenticate the client during the connection process. • In the default connection and authentication process, the client requests a certificate from the HTTP server, but the server does not attempt to authenticate the client. Authenticating the client provides more security than server authentication by itself, but not all clients may be configured for CA authentication.
Step 9	**ip http secure-trustpoint** *name* **Example:** `Router(config)# ip http secure-trustpoint trustpoint-01`	Specifies the CA trustpoint that should be used to obtain an X.509v3 security certificate and to authenticate the connecting client's certificate. • Use of this command assumes you have already declared a CA trustpoint using the **crypto ca trustpoint** command and associated submode commands. • Use the same trustpoint name that you used in the associated **crypto ca trustpoint** command.

	Command or Action	Purpose
Step 10	**end** **Example:** `Router(config)# end`	Ends the current configuration session and returns you to privileged EXEC mode.
Step 11	**show ip http server secure status** **Example:** `Router# show ip http server secure status`	Displays the status of the HTTP secure server configuration.

Verifying the Configuration of the HTTPS Server

To verify the configuration of the HTTPS server, connect to the router running the HTTPS server with a web browser by entering **https://***url*, where *url* is the IP address or hostname of the router. Successful connection using the **https** prefix (instead of the standard **http**) indicates that the HTTPS server is configured properly. If a port other than the default port is configured (using the **ip http secure-port** command), you must also specify the port number after the URL. For example:

```
https://209.165.202.129:1026
```
or
```
https://host.domain.com:1026
```

Generally, you can verify that the HTTPS server is configured and that you have a secure connection by locating an image of a padlock at the bottom of your browser window. Also note that secure HTTP connections have a URL that starts with “https:” instead of “http:”.

Providing Additional Security and Efficiency

The configuration of the standard HTTP server applies to the secure HTTP server as well. To provide additional security and efficiency to both the standard HTTP server and the HTTPS server, complete the procedure in this section.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip http path** *path-name*
4. **ip http access-class** *access-list-number*
5. **ip http max-connections** *value*
6. **ip http timeout-policy idle** *seconds* **life** *seconds* **requests** *value*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**ip http path** *path-name* **Example:** `Router(config)# ip http path slot1:`	(Optional) Sets the base HTTP path for HTML files. • The base path is used to specify the location of the HTTP server files (HTML files) on the local system. Generally, the HTML files are located in system flash memory.
Step 4	**ip http access-class** *access-list-number* **Example:** `Router(config)# ip http access-class 20`	(Optional) Specifies the access list that should be used to allow access to the HTTP server.
Step 5	**ip http max-connections** *value* **Example:** `Router(config)# ip http max-connections 10`	(Optional) Sets the maximum number of concurrent connections to the HTTP server that will be allowed. The default value is 5.

	Command or Action	Purpose
Step 6	**ip http timeout-policy idle** *seconds* **life** *seconds* **requests** *value* **Example:** `Router(config)# ip http timeout-policy idle 30 life 120 requests 100`	(Optional) Sets the characteristics that determine how long a connection to the HTTP server should remain open. The characteristics are: • **idle**—The maximum number of seconds the connection will be kept open if no data is received or response data cannot be sent out on the connection. Note that a new value may not take effect on any already existing connections. If the server is too busy or the limit on the **life** time or the number of **requests** is reached, the connection may be closed sooner. The default value is 180 seconds (3 minutes). • **life**—The maximum number of seconds the connection will be kept open, from the time the connection is established. Note that the new value may not take effect on any already existing connections. If the server is too busy or the limit on the idle time or the number of requests is reached, it may close the connection sooner. Also, because the server will not close the connection while actively processing a request, the connection may remain open longer than the specified **life** time if processing is occurring when the life maximum is reached. In this case, the connection will be closed when processing finishes. The default value is 180 seconds (3 minutes). The maximum value is 86,400 seconds (24 hours). • **requests**—The maximum limit on the number of requests processed on a persistent connection before it is closed. Note that the new value may not take effect on any already existing connections. If the server is too busy or the limit on the idle time or the life time is reached, the connection may be closed before the maximum number of requests are processed. The default value is 1. The maximum value is 86,400.

Configuring the HTTPS Client with SSL 3.0

To configure the HTTPS client with SSL 3.0, complete the procedure in this section.

Prerequisites

The standard HTTP client and the secure HTTP client are always enabled.

A certificate authority is required for secure HTTP client certification; the following steps assume that you have previously declared a CA trustpoint on the routing device. If a CA trustpoint is not configured, and the remote HTTPS server requires client authentication, connections to the secure HTTP client will fail.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip http client secure-trustpoint** *trustpoint-name*

4. **ip http client secure-ciphersuite** [**3des-ede-cbc-sha**] [**rc4-128-sha**] [**rc4-128-md5**] [**des-cbc-sha**]
5. **end**
6. **show ip http client secure status**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**ip http client secure-trustpoint** *trustpoint-name* **Example:** `Router(config)# ip http client secure-trustpoint trustpoint01`	(Optional) Specifies the CA trustpoint that should be used if the remote HTTP server requests client authentication. • Use of this command assumes you have already declared a CA trustpoint using the **crypto ca trustpoint** command and associated submode commands. • Use the same trustpoint name that you used in the associated **crypto ca trustpoint** command. • This command is optional if client authentication is not needed, or if a primary trustpoint has been configured. If the **ip http client secure-trustpoint** command is not used, the router will use the primary trustpoint, as specified by the **primary** CA trustpoint configuration mode command.
Step 4	**ip http client secure-ciphersuite** [**3des-ede-cbc-sha**] [**rc4-128-sha**] [**rc4-128-md5**] [**des-cbc-sha**] **Example:** `Router(config)# ip http client secure-ciphersuite rc4-128-sha rc4-128-md5`	(Optional) Specifies the CipherSuites (encryption algorithms) that should be used for encryption over the HTTPS connection. • This command allows you to restrict the list of CipherSuites that the client offers when connecting to a secure HTTP server. For example, you may want to allow only the most secure CipherSuites to be used. • Unless you have a reason to specify the CipherSuites that should be used, or you are unfamiliar with the details of these CipherSuites, you should leave this command unconfigured and let the server and client negotiate the CipherSuite that they both support (this is the default).

	Command or Action	Purpose
Step 5	**end** **Example:** `Router(config)# end`	Ends the current configuration session and returns to privileged EXEC mode.
Step 6	**show ip http client secure status** **Example:** `Router# show ip http client secure status`	Displays the status of the HTTP secure server configuration.

Configuration Examples for the HTTPS—HTTP Server and Client with SSL 3.0 feature

The following example shows a configuration session in which the secure HTTP server is enabled, the port for the secure HTTP server is configured as 1025, and the remote CA trustpoint server "CA-trust-local" is used for certification.

```
Router# show ip http server status

HTTP server status: Disabled
HTTP server port: 80
HTTP server authentication method: enable
HTTP server access class: 0
HTTP server base path:
Maximum number of concurrent server connections allowed: 5
Server idle time-out: 600 seconds
Server life time-out: 600 seconds
Maximum number of requests allowed on a connection: 1
HTTP secure server capability: Present
HTTP secure server status: Disabled
HTTP secure server port: 443
HTTP secure server ciphersuite: 3des-ede-cbc-sha des-cbc-sha rc4-128-md5 rc4-12a
HTTP secure server client authentication: Disabled
HTTP secure server trustpoint:

Router# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)# ip http secure-server
Router(config)# ip http client secure-trustpoint CA-trust-local
Router(config)# ip http secure-port 1024
Invalid secure port value.

Router(config)# ip http secure-port 1025
Router(config)# ip http secure-ciphersuite rc4-128-sha rc4-128-md5
Router(config)# end
Router# show ip http server secure status
HTTP secure server status: Enabled
HTTP secure server port: 1025
HTTP secure server ciphersuite: rc4-128-md5 rc4-128-sha
HTTP secure server client authentication: Disabled
HTTP secure server trustpoint: CA-trust-local
```

In the following example, the CA trustpoint CA-trust-local is specified, and the HTTPS client is configured to use this trustpoint for client authentication requests:

```
Router# config terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)# crypto ca trustpoint CA-trust-local
Router(ca-trustpoint)# enrollment url http://example.com
Router(ca-trustpoint)# crl query ldap://example.com
Router(ca-trustpoint)# primary
Router(ca-trustpoint)# exit
Router(config)# ip http client secure-trustpoint CA-trust-local
Router(config)# end
Router# copy running-config startup-config
```

Additional References

The following sections provide references related to the HTTPS—HTTP Server and Client with SSL 3.0 feature.

Related Documents

Related Topic	Document Title
SSL 3.0	*The SSL Protocol Version 3.0* This document is available from various sources online.
Standard Cisco Web Client	*HTTP 1.1 Web Server and Client*
Certification Authority Interoperability	*Cisco IOS Security Configuration Guide: Secure Connectivity*

Standards

Standard	Title
No new or modified standards are supported by this feature.	—

Related MIBs

MIBs	MIBs Link
No new or modified MIBs are supported by this feature.	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

Related RFCs

RFCs	Description
RFC 2616	Cisco's implementation of HTTP is based on *RFC 2616: Hypertext Transfer Protocol -- HTTP/1.1*.

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/techsupport

Feature Information for HTTPS—HTTP Server and Client with SSL 3.0

Table 1 lists the release history for this feature.

Not all commands may be available in your Cisco IOS software release. For release information about a specific command, see the command reference documentation.

Use Cisco Feature Navigator to find information about platform support and software image support. Cisco Feature Navigator enables you to determine which Cisco IOS and Catalyst OS software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Note Table 1 lists only the Cisco IOS software release that introduced support for a given feature in a given Cisco IOS software release train. Unless noted otherwise, subsequent releases of that Cisco IOS software release train also support that feature.

Table 1 Feature Information for HTTPS—HTTP Server and Client with SSL 3.0

Feature Name	Releases	Feature Information
HTTPS—HTTP Server and Client with SSL 3.0	12.2(15)T 12.2(33)SRA 12.2(33)SXH 12.2(33)SB	This feature provides Secure Socket Layer (SSL) version 3.0 support for the HTTP 1.1 server and HTTP 1.1 client within Cisco IOS software. SSL provides server authentication, encryption, and message integrity to allow secure HTTP communications. SSL also provides HTTP client authentication. This feature is supported only in Cisco software images that support SSL. Specifically, SSL is supported in "IPSec 56" and "IPSec 3DES" images (contains "k8" or "k9" in the image name).

Glossary

RSA—RSA is a widely used Internet encryption and authentication system that uses public and private keys for encryption and decryption. The RSA algorithm was invented in 1978 by Ron Rivest, Adi Shamir, and Leonard Adleman. The abbreviation RSA comes from the first letter of the last names of the three original developers. The RSA algorithm is included in many applications, such as the web browsers from Microsoft and Netscape. The RSA encryption system is owned by RSA Security.

SHA—The Secure Hash Algorithm. SHA was developed by NIST and is specified in the Secure Hash Standard (SHS, FIPS 180). Often used as an alternative to Digest 5 algorithm.

signatures, digital—In the context of SSL, "signing" means to encrypt with a private key. In digital signing, one-way hash functions are used as input for a signing algorithm. In RSA signing, a 36-byte structure of two hashes (one SHA and one MD5) is signed (encrypted with the private key).

SSL 3.0—Secure Socket Layer version 3.0. SSL is a security protocol that provides communications privacy over the Internet. The protocol allows client and server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery. SSL uses a program layer located between the Internet's HTTP and TCP layers. SSL is included as part of most web server products and as part of most Internet browsers. The SSL 3.0 specification can be found at http://home.netscape.com/eng/ssl3/.

HTTP Client API for Tcl IVR

The HTTP Client API for Tcl IVR feature provides support for Tcl IVR applications to retrieve data from or post data to an HTTP server. Also introduced with this feature is a new command-line interface structure for configuring voice applications and support for additional Tcl 8.3.4 commands.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the "Feature Information for HTTP Client API for Tcl IVR" section on page 4.

Use Cisco Feature Navigator to find information about platform support and Cisco IOS and Catalyst OS software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Contents

Prerequisites for HTTP Client API for Tcl IVR and New Cisco Voice Application Command-Line Interface Structure

- Familiarity with Tcl IVR, VoiceXML, and Cisco IOS commands.
- Required hardware:

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

  - Cisco 3600 series
  - Cisco AS5300
  - Cisco AS5350
  - Cisco AS5400
  - Cisco AS5800
  - Cisco AS58550
- Required software:
  - Cisco IOS Release 12.3(14)T or later
  - Tcl 8.3.4
  - VoiceXML 2.0

Restrictions for HTTP Client API for Tcl IVR and New Cisco Voice Application Command-Line Interface Structure

If Cisco IOS configuration commands are used within the Tcl scripts, submode commands must be entered as quoted arguments on the same line as the configuration command.

Information About HTTP Client API for Tcl IVR and New Cisco Voice Application Command-Line Interface Structure

HTTP API for Tcl IVR 2.0

An HTTP application programming interface to the IOS HTTP client is provided. The HTTP package is accessed using the **package require httpios 1.0** Tcl command. Additional commands are provided to configure HTTP. See the *Tcl IVR API Version 2.0 Programming Guide* for more information.

Newly-Supported Tcl 8.3.4 Commands

The following Tcl 8.3.4 commands are now supported:

- cd
- close
- eof
- fconfigure
- file

- fileevent
- flush
- glob
- namespace
- open
- package
- pwd
- read
- seek

The following command is modified:

- puts

See the *Tcl IVR API Version 2.0 Programming Guide* for more information.

New Cisco Voice Application Command-Line Interface Structure

The **call application voice** command structure for configuring Tcl and IVR applications has been restructured to provide easier configuration of application parameters than the earlier CLI structure.

For more information, see the "Cisco IOS Release 12.3(14)T and Later Voice Application Command-Line Interface Structure Changes" section in Configuring Basic Functionality for Tcl IVR and VoiceXML Applications in the *Cisco IOS Tcl IVR and VoiceXML Application Guide*.

Feature Information for HTTP Client API for Tcl IVR

Table 1 lists the release history for this feature.

Not all commands may be available in your Cisco IOS software release. For release information about a specific command, see the command reference documentation.

Use Cisco Feature Navigator to find information about platform support and software image support. Cisco Feature Navigator enables you to determine which Cisco IOS and Catalyst OS software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Note Table 1 lists only the Cisco IOS software release that introduced support for a given feature in a given Cisco IOS software release train. Unless noted otherwise, subsequent releases of that Cisco IOS software release train also support that feature.

Table 1 Feature Information for HTTP Client API for Tcl IVR

Feature Name	Releases	Feature Information
HTTP Client API for Tcl IVR	12.3(14)T	This feature was introduced.

HTTP Inspection Engine

The HTTP Inspection Engine feature allows users to configure their Cisco IOS Firewall to detect and prohibit HTTP connections—such as tunneling over port 80, unauthorized request methods, and non-HTTP compliant file transfers—that are not authorized within the scope of the security policy configuration. Tunneling unauthorized protocols through port 80 and over HTTP exposes a network to significant security risks.

The Cisco IOS Firewall can now be configured with a security policy that adheres to the following tasks:

- Allowing specific traffic targeted for port 80 to traverse the firewall. The traffic is inspected for protocol conformance and for the types of HTTP commands that are allowed or disallowed.
- Denying specific traffic targeted for port 80 that does not comply to HTTP traffic standards. The firewall is enabled to drop the packet, reset the connection, and send a syslog message, as appropriate.

Feature History for HTTP Inspection Engine

Release	Modification
12.3(14)T	This feature was introduced.

Finding Support Information for Platforms and Cisco IOS Software Images

Use Cisco Feature Navigator to find information about platform support and Cisco IOS software image support. Access Cisco Feature Navigator at http://www.cisco.com/go/fn. You must have an account on Cisco.com. If you do not have an account or have forgotten your username or password, click **Cancel** at the login dialog box and follow the instructions that appear.

Contents

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

Restrictions for HTTP Inspection Engine

The Cisco 831 router with 48M RAM does not have enough memory to support this feature.

Information About HTTP Inspection Engine

Before configuring an application firewall to detect and police specific traffic targeted for port 80, you should understand the following concepts:

What Is a Security Policy?

The application firewall uses a security policy, which consists of a collection of static signatures, to detect security violations. A static signature is a collection of parameters that specify protocol conditions that must be met before an action is taken. (For example, a signature may specify that an HTTP data stream containing the POST method must reset the connection.) These protocol conditions and reactions are defined by the end user via the command-line interface (CLI) to form a security policy.

Cisco IOS HTTP Application Policy Overview

HTTP uses port 80 to transport Internet web services, which are commonly used on the network and rarely challenged with regards to their legitimacy and conformance to standards. Because port 80 traffic is typically allowed through the network without being challenged, many application developers are leveraging HTTP traffic as an alternative transport protocol in which to enable their application to travel through or even bypass the firewall.

Most firewalls provide only packet filtering capabilities that simply permit or deny port 80 traffic without inspecting the data stream; the Cisco IOS application firewall for HTTP performs packet inspection as follows:

- Detects HTTP connections that are not authorized within the scope of the security policy configuration.
- Detects users who are tunneling applications through port 80.

If the packet is not in compliance with the HTTP protocol, it will be dropped, the connection will be reset, and a syslog message will be generated, as appropriate.

How to Define and Apply an HTTP Application Policy to a Firewall for Inspection

This section contains the following procedures:

Defining an HTTP Application Policy

Use this task to create an HTTP application firewall policy.

Restrictions

Although application firewall policies are defined in global configuration mode, only one global policy for a given protocol is allowed per interface.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **appfw policy-name** *policy-name*
4. **application** *protocol*
5. **strict-http action** {**reset** | **allow**} [**alarm**]
6. **content-length** {**min** *bytes* **max** *bytes* | **min** *bytes* | **max** *bytes*} **action** {**reset** | **allow**} [**alarm**]
7. **content-type-verification** [**match-req-resp**] **action** {**reset** | **allow**} [**alarm**]
8. **max-header-length** {**request** *bytes* **response** *bytes*} **action** {**reset** | **allow**} [**alarm**]
9. **max-uri-length** *bytes* **action** {**reset** | **allow**} [**alarm**]
10. **request-method** {**rfc** *rfc-method* | **extension** *extension-method*} **action** {**reset** | **allow**} [**alarm**]
11. **port-misuse** {**p2p** | **tunneling** | **im** | **default**} **action** {**reset** | **allow**} [**alarm**
12. **transfer-encoding type** {**chunked** | **compress** | **deflate** | **gzip** | **identity** | **default**} **action** {**reset** | **allow**} [**alarm**]
13. **timeout** *seconds*
14. **audit-trail** {**on** | **off**}
15. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**appfw policy-name** *policy-name* **Example:** `Router(config)# appfw policy-name mypolicy`	Defines an application firewall policy and puts the router in application firewall policy configuration mode.

	Command or Action	Purpose
Step 4	**application** *protocol* **Example:** `Router(cfg-appfw-policy)# application http`	Allows you to configure inspection parameters for a given protocol. Currently, only HTTP traffic can be inspected. • *protocol* —Specify the **http** keyword. This command puts you in appfw-policy-*protocol* configuration mode, where "*protocol*" is dependent upon the specified protocol. Because only HTTP can be specified, the configuration mode is appfw-policy-http.
Step 5	**strict-http action** {**reset** | **allow**} [**alarm**] **Example:** `Router(cfg-appfw-policy-http)# strict-http action allow alarm`	(Optional) Allows HTTP messages to pass through the firewall or resets the TCP connection when HTTP noncompliant traffic is detected.
Step 6	**content-length** {**min** *bytes* **max** *bytes* | **min** *bytes* | **max** *bytes*} **action** {**reset** | **allow**} [**alarm**] **Example:** `Router(cfg-appfw-policy-http)# content-length max 1 action allow alarm`	(Optional) Permits or denies HTTP traffic through the firewall on the basis of message size. • **min** | **max** *bytes*—Minimum or maximum content length, in bytes, allowed per message. Number of bytes range: 0 to 65535.
Step 7	**content-type-verification** [**match-req-resp**] **action** {**reset** | **allow**} [**alarm**] **Example:** `Router(cfg-appfw-policy-http)# content-type-verification match-req-resp action allow alarm`	(Optional) Permits or denies HTTP traffic through the firewall on the basis of content message type.
Step 8	**max-header-length** {**request** *bytes* **response** *bytes*} **action** {**reset** | **allow**} [**alarm**] **Example:** `Router(cfg-appfw-policy-http)# max-header-length request 1 response 1 action allow alarm`	(Optional) Permits or denies HTTP traffic on the basis of the message header length. • *bytes*—Number of bytes ranging from 0 to 65535.
Step 9	**max-uri-length** *bytes* **action** {**reset** | **allow**} [**alarm**] **Example:** `Router(cfg-appfw-policy-http)# max-uri-length 1 action allow alarm`	(Optional) Permits or denies HTTP traffic on the basis of the URI length in the request message.

	Command or Action	Purpose
Step 10	**request method** {**rfc** *rfc-method* | **extension** *extension-method*} **action** {**reset** | **allow**} [**alarm**] **Example:** `Router(cfg-appfw-policy-http)# request-method rfc default action allow alarm`	(Optional) Permits or denies HTTP traffic according to either the request methods or the extension methods. • **rfc**—Specifies that the supported methods of RFC 2616, *Hypertext Transfer Protocol—HTTP/1.1*, are to be used for traffic inspection. • *rfc-method*—Any one of the following RFC 2616 methods can be specified: **connect**, **default, delete**, **get**, **head**, **options**, **post**, **put**, **trace**. • **extension**—Specifies that the extension methods are to be used for traffic inspection. • *extension-method*—Any one of the following extension methods can be specified: **copy**, **default, edit**, **getattribute**, **getproperties**, **index**, **lock**, **mkdir**, **move**, **revadd**, **revlabel**, **revlog**, **save**, **setattribute**, **startrev**, **stoprev**, **unedit**, **unlock**.
Step 11	**port-misuse** {**p2p** | **tunneling** | **im** | **default**} **action** {**reset** | **allow**} [**alarm**] **Example:** `Router(cfg-appfw-policy-http)# port-misuse default action allow alarm`	(Optional) Permits or denies HTTP traffic through the firewall on the basis of specified applications in the HTTP message. • **p2p**—Peer-to-peer protocol applications subject to inspection: Kazaa and Gnutella. • **tunneling**—Tunneling applications subject to inspection: HTTPPort/HTTPHost, GNU Httptunnel, GotoMyPC, Firethru, Http-tunnel.com Client • **im**—Instant messaging protocol applications subject to inspection: Yahoo Messenger. • **default**—All applications are subject to inspection.
Step 12	**transfer-encoding type** {**chunked** | **compress** | **deflate** | **gzip** | **identity** | **default**} **action** {**reset** | **allow**} [**alarm**] **Example:** `Router(cfg-appfw-policy-http)# transfer-encoding type default action allow alarm`	(Optional) Permits or denies HTTP traffic according to the specified transfer-encoding of the message. • **chunked**—Encoding format (specified in RFC 2616, *Hypertext Transfer Protocol—HTTP/1*) in which the body of the message is transferred in a series of chunks; each chunk contains its own size indicator. • **compress**—Encoding format produced by the UNIX "compress" utility. • **deflate**—"ZLIB" format defined in RFC 1950, *ZLIB Compressed Data Format Specification version 3.3*, combined with the "deflate" compression mechanism described in RFC 1951, *DEFLATE Compressed Data Format Specification version 1.3*. • **gzip**—Encoding format produced by the "gzip" (GNU zip) program. • **identity**—Default encoding, which indicates that no encoding has been performed. • **default**—All of the transfer encoding types.

	Command or Action	Purpose
Step 13	**timeout** *seconds* **Example:** `Router(cfg-appfw-policy-http)# timeout 60`	(Optional) Overrides the global TCP idle timeout value for HTTP traffic. **Note** If this command is not issued, the default value specified via the **ip inspect tcp idle-time** command will be used.
Step 14	**audit-trail** {**on** | **off**} **Example:** `Router(cfg-appfw-policy-http)# audit-trail on`	(Optional) Turns audit trail messages on or off. **Note** If this command is not issued, the default value specified via the **ip inspect audit-trail** command will be used.
Step 15	**end** **Example:** `Router(cfg-appfw-policy-http)# end`	Exits cfg-appfw-policy-http configuration mode.

What to Do Next

After you have successfully defined an application policy for HTTP traffic inspection, you must apply the policy to an inspection rule. Thereafter, the inspection rule must be applied to an interface. For information on completing this task, see the section "Applying an HTTP Application Policy to a Firewall for Inspection."

Applying an HTTP Application Policy to a Firewall for Inspection

Use this task to apply an HTTP application policy to an inspection rule, followed by applying the inspection rule to an interface.

Note An application policy can coexist with other inspection protocols (for example, an HTTP policy and an FTP policy can coexist).

Prerequisites

You must have already defined an application policy (as shown in the section "Defining an HTTP Application Policy").

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip inspect name** *inspection-name* **appfw** *policy-name*
4. **ip inspect name** *inspection-name* **http** [**alert** {**on** | **off**}] [**audit-trail** {**on** | **off**}] [**timeout** *seconds*]
5. **interface** *type number*
6. **ip inspect** *inspection-name* {**in** | **out**}
7. **exit**

8. **exit**
9. **show appfw configuration** [*name*]

   or

   show ip inspect {**name** *inspection-name* | **config** | **interfaces** | **session** [**detail**] | **statistics** | **all**}

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**ip inspect name** *inspection-name* **appfw** *policy-name* **Example:** `Router(config)# ip inspect name firewall appfw mypolicy`	Defines a set of inspection rules for the application policy. • *policy-name*—Must match the policy name specified via the **appfw policy-name** command.
Step 4	**ip inspect name** *inspection-name* **http** [**alert** {**on** | **off**}] [**audit-trail** {**on** | **off**}] [**timeout** *seconds*] **Example:** `Router(config)# ip inspect name firewall http`	Defines a set of inspection rules that is to be applied to all HTTP traffic. • The *inspection-name* argument must match the *inspection-name* argument specified in Step 3.
Step 5	**interface** *type number* **Example:** `Router#(config)# interface FastEthernet0/0`	Configures an interface type and enters interface configuration mode.
Step 6	**ip inspect** *inspection-name* {**in** | **out**} **Example:** `Router#(config-if)# ip inspect firewall in`	Applies the inspection rules (defined in Step 3 and Step 4) to all traffic entering the specified interface. • The *inspection-name* argument must match the inspection name defined via the **ip inspect name** command.
Step 7	**exit** **Example:** `Router#(config-if)# exit`	`Exits interface configuration mode.`

	Command or Action	Purpose
Step 8	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode.

Step 9	**show appfw configuration** [*name*]
	
	Example:
	`Router# show appfw configuration`
	
	or
	
	show ip inspect {**name** *inspection-name* | **config** | **interfaces** | **session** [**detail**] | **statistics** | **all**}
	
	Example:
	`Router# show ip inspect config`
		(Optional) Displays application firewall policy configuration information.
		
		(Optional) Displays firewall-related configuration information.

Troubleshooting Tips

To help troubleshoot the application firewall configuration, issue the following application-firewall specific debug command: **debug appfw** {**application** *protocol* **| function-trace | object-creation | object-deletion | events | timers | detailed**}.

The following sample configuration shows how to configure an HTTP policy with application firewall debugging enabled:

```
Router(config)# appfw policy-name myPolicyAPPFW  FUNC:appfw_policy_find
APPFW  FUNC:appfw_policy_find -- Policy myPolicy is not found
APPFW  FUNC:appfw_policy_alloc
APPFW  FUNC:appfw_policy_alloc -- policy_alloc 0x65727278
APPFW  FUNC:appfw_policy_alloc -- Policy 0x65727278 is set to valid
APPFW  FUNC:appfw_policy_alloc -- Policy myPolicy has been created
APPFW  FUNC:appfw_policy_command -- memlock policy 0x65727278

! Debugging sample for application (HTTP) creation

Router(cfg-appfw-policy)# application httpAPPFW  FUNC:appfw_http_command
APPFW  FUNC:appfw_http_appl_find
APPFW  FUNC:appfw_http_appl_find -- Application not found
APPFW  FUNC:appfw_http_appl_alloc
APPFW  FUNC:appfw_http_appl_alloc -- appl_http 0x64D7A25C
APPFW  FUNC:appfw_http_appl_alloc -- Application HTTP parser structure 64D7A25C created

! Debugging sample for HTTP-specific application inspection
Router(cfg-appfw-policy-http)#
Router(cfg-appfw-policy-http)# strict-http action reset alarm
APPFW  FUNC:appfw_http_subcommand
APPFW  FUNC:appfw_http_subcommand -- strict-http cmd turned on

Router# debug appfw detailed

APPFW Detailed Debug debugging is on
fw7-7206a#debug appfw object-creation
APPFW Object Creations debugging is on
fw7-7206a#debug appfw object-deletion
APPFW Object Deletions debugging is on
```

Configuration Examples for Setting Up an HTTP Inspection Engine

This section contains the following configuration example:

Setting Up and Verifying an HTTP Inspection Engine: Example

The following example show how to define the HTTP application firewall policy "mypolicy." This policy includes all supported HTTP policy rules. This example also includes sample output from the **show appfw configuration** and **show ip inspect config** commands, which allow you to verify the configured setting for the application policy.

```
! Define the HTTP policy.
appfw policy-name mypolicy
 application http
  strict-http action allow alarm
  content-length maximum 1 action allow alarm
  content-type-verification match-req-rsp action allow alarm
  max-header-length request 1 response 1 action allow alarm
  max-uri-length 1 action allow alarm
  port-misuse default action allow alarm
  request-method rfc put action allow alarm
  transfer-encoding type default action allow alarm
!
!
! Apply the policy to an inspection rule.
ip inspect name firewall appfw mypolicy
ip inspect name firewall http
!
!
! Apply the inspection rule to all HTTP traffic entering the FastEthernet0/0 interface.
interface FastEthernet0/0
 ip inspect firewall in
!
!
! Issue the show appfw configuration command and the show ip inspect config command after
the inspection rule "mypolicy" is applied to all incoming HTTP traffic on the
FastEthernet0/0 interface.
!
Router# show appfw configuration

Application Firewall Rule configuration
  Application Policy name mypolicy
    Application http
      strict-http action allow alarm
      content-length minimum 0 maximum 1 action allow alarm
      content-type-verification match-req-rsp action allow alarm
      max-header-length request length 1 response length 1 action allow alarm
      max-uri-length 1 action allow alarm
      port-misuse default action allow alarm
      request-method rfc put action allow alarm
      transfer-encoding default action allow alarm

Router# show ip inspect config

Session audit trail is disabled
Session alert is enabled
```

```
one-minute (sampling period) thresholds are [400:500] connections
max-incomplete sessions thresholds are [400:500]
max-incomplete tcp connections per host is 50. Block-time 0 minute.
tcp synwait-time is 30 sec -- tcp finwait-time is 5 sec
tcp idle-time is 3600 sec -- udp idle-time is 30 sec
dns-timeout is 5 sec
Inspection Rule Configuration
Inspection name firewall
http alert is on audit-trail is off timeout 3600
```

Additional References

The following sections provide references related to the HTTP Inspection Engine feature.

Related Documents

Related Topic	Document Title
Firewall commands: complete command syntax, command mode, defaults, usage guidelines, and examples	*Cisco IOS Security Command Reference*

Standards

Standards	Title
No new or modified standards are supported by this feature.	—

MIBs

MIBs	MIBs Link
No new or modified MIBs are supported by this feature.	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFCs	Title
RFC 2616	*Hypertext Transfer Protocol -- HTTP/1.1*

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/public/support/tac/home.shtml

RMON Support

Configuring RMON Support

First Published: July 27, 1999
Last Updated: November 20, 2009

This module describes the Remote Monitoring (RMON) MIB agent specification and its usage in conjunction with Simple Network Management Protocol (SNMP) to monitor traffic using alarms and events.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the "Feature Information for Configuring RMON Support" section on page 18.

Use Cisco Feature Navigator to find information about platform support and Cisco IOS and Catalyst OS software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Contents

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

Prerequisites for Configuring RMON Support

- RMON requires SNMP to be configured (you must be running a version of SNMP on the server that contains the RMON MIB).
- RMON can be very data and processor intensive. You must measure usage effects to ensure that router performance is not degraded by RMON and to minimize excessive management traffic overhead. Native mode in RMON is less intensive than promiscuous mode.

Restrictions for Configuring RMON Support

- Full RMON packet analysis (as described in RFC 1757) is supported only on an Ethernet interface of Cisco 2500 series routers and Cisco AS5200 series universal access servers.
- A generic RMON console application is recommended in order to take advantage of the RMON network management capabilities.

Information About Configuring RMON Support

To configure RMON, you need to understand the following concepts:

- RMON Overview, page 2
- RMON Groups, page 3
- RMON Event and Alarm Notifications, page 4
- RMON MIB, page 5
- HC Alarm MIB, page 6

RMON Overview

RMON is a standard monitoring specification that enables various network monitors and console systems to exchange network-monitoring data. RMON provides network administrators with more flexibility in selecting network-monitoring probes and consoles with features that meet their particular networking needs.

The RMON specification defines a set of statistics and functions that can be exchanged between RMON-compliant console managers and network probes. RMON provides network administrators with comprehensive network-fault diagnosis, planning, and performance-tuning information.

The RMON feature identifies activity on individual nodes and allows you to monitor all nodes and their interaction on a LAN segment. Used in conjunction with the SNMP agent in a router, RMON allows you to view both traffic that flows through the router and segment traffic that is not necessarily destined for the router. Combining RMON alarms and events (classes of messages that indicate traffic violations and various unusual occurrences over a network) with existing MIBs allows you to choose where proactive monitoring will occur.

RMON Groups

RMON delivers information in RMON groups of monitoring elements, each providing specific sets of data to meet common network-monitoring requirements. Each group is optional so that you do not need to support all the groups within the Management Information Base (MIB). Some RMON groups require support of other RMON groups to function properly.

Table 1 summarizes the nine monitoring groups specified in the RFC 1757 Ethernet RMON MIB. For more information on gathering RMON statistics for these data types, refer to "Configuring RMON Groups" section on page 9.

Note All Cisco IOS software images ordered without the explicit RMON option include limited RMON support (RMON alarms and event groups only). Images ordered with the RMON option include support for all nine management groups (statistics, history, alarms, hosts, hostTopN, matrix, filter, capture, and event). As a security precaution, support for the capture group allows capture of packet header information only; data payloads are not captured.

Table 1 RMON Monitoring Groups

RMON Group	Function	Elements
Statistics	Contains statistics measured by the probe for each monitored interface on this device.	Packets dropped, packets sent, bytes sent (octets), broadcast packets, multicast packets, CRC errors, runts, giants, fragments, jabbers, collisions, and counters for packets ranging from 64 to 128, 128 to 256, 256 to 512, 512 to 1024, and 1024 to 1518 bytes.
History	Records periodic statistical samples from a network and stores them for later retrieval.	Sample period, number of samples, items sampled.
Alarm	Periodically takes statistical samples from variables in the probe and compares them with previously configured thresholds. If the monitored variable crosses a threshold, an event is generated.	Includes the alarm table and requires the implementation of the event group. Alarm type, interval, starting threshold, stop threshold.
Host	Contains statistics associated with each host discovered on the network.	Host address, packets, and bytes received and transmitted, as well as broadcast, multicast, and error packets.
HostTopN	Prepares tables that describe the hosts that top a list ordered by one of their base statistics over an interval specified by the management station. Thus, these statistics are rate-based.	Statistics, host(s), sample start and stop periods, rate base, duration.

Table 1 RMON Monitoring Groups

RMON Group	Function	Elements
Matrix	Stores statistics for conversations between sets of two addresses. As the device detects a new conversation, it creates a new entry in its table.	Source and destination address pairs and packets, bytes, and errors for each pair.
Filters	Enables packets to be matched by a filter equation. These matched packets form a data stream that might be captured or that might generate events.	Bit-filter type (mask or not mask), filter expression (bit level), conditional expression (and, or not) to other filters.
Packet Capture	Enables packets to be captured after they flow through a channel.	Size of buffer for captured packets, full status (alarm), number of captured packets.
Events	Controls the generation and notification of events from this device.	Event type, description, last time event sent.

RMON Event and Alarm Notifications

Thresholds allow you to minimize the number of notifications sent on the network. The RMON MIB defines two traps, the risingAlarm trap which is the rising-threshold value and fallingAlarm trap which is the falling-threshold value. Alarms are triggered when a problem exceeds a set rising-threshold value. No alarm notifications are sent until the agent recovers, as defined by the falling-threshold value. This means that notifications are not sent each time a minor failure or recovery occurs.

You can set an RMON alarm on any MIB object in the access server. You cannot disable all the alarms you configure at once. The delta value tests the change between MIB variables, which affects the alarmSampleType in the alarmTable of the RMON MIB. The absolute value tests each MIB variable directly, which affects the alarmSampleType in the alarmTable of the RMON MIB.

Refer to RFC 1757 to learn more about alarms and events and how they interact with each other.

RMON MIB

RMON MIB supports for polling of 64 bit counters and includes the following features:

- usrHistory group. This MIB group is similar to the RMON etherHistory group except that the group enables you to specify the MIB objects that are collected at each interval.
- partial probeConfig group. This MIB group is a subset of the probeConfig group implemented in read-only mode. These objects implement the simple scalars from this group. Table 2 details new partial probeConfig group objects.

Table 2 partial probeConfig Group Objects

Object	Description
probeCapabilities	The RMON software groups implemented.
probeSoftwareRev	The current version of Cisco IOS software running on the device.
probeHardwareRev	The current version of the Cisco device.
probeDateTime	The current date and time.
probeResetControl	Initiates a reset.
probeDownloadFile	The source of the image running on the device.
probeDownloadTFTPServer	The address of the server that contains the Trivial File Transfer Protocol (TFTP) file that is used by the device to download new versions of Cisco IOS software.
probeDownloadAction	Specifies the action of the commands that cause the device to reboot.
probeDownloadStatus	The state of a reboot.
netDefaultGateway	The router mapped to the device as the default gateway.
hcRMONCapabilities	Specifies the features mapped to this version of RMON.

In Cisco IOS Release 12.1, the RMON agent was rewritten to improve performance and add some new features. Table 3 highlights some of the improvements implemented.

Table 3 RMON MIB Updates

Prior to the RMON MIB Update in Cisco IOS Release 12.1	New Functionality in Cisco IOS Release 12.1
RMON configurations do not persist across reboots. Information is lost after a new session on the RMON server.	RMON configurations persist across reboots. Information is preserved after a new session on the RMON server.
Packet analysis applies only on the MAC header of the packet.	Complete packet capture is performed with analysis applied to all frames in packet.
Only RMON I MIB objects are used for network monitoring.	RMON I and selected RMON II objects are used for network monitoring.

HC Alarm MIB

The high-capacity (HC) Alarm MIB, which is an extension of RMON Alarm group table objects, supports polling of RMON variables up to 64 bit values. The HC-ALARM-MIB defines two traps, the hcRisingAlarm which provides the rising-threshold value and hcFallingAlarm which provides the falling-threshold value.

Refer to RFC 3434 to learn more about HC alarms.

How to Configure RMON Support

The tasks in the following sections explain how to configure RMON support:

- Configuring RMON, page 6 (required)
- Configuring RMON Event and Alarm Notifications, page 7 (required)
- Configuring RMON Groups, page 9 (optional)

Configuring RMON

This task explains how to configure RMON and RMON queue size. In native mode, RMON monitors only those packets that are received by the interface. In promiscuous mode, RMON monitors all packets on the LAN segment.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **rmon** {**native** | **promiscuous**}
5. **exit**
6. **rmon queuesize** *size*
7. **exit**
8. **show rmon**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**interface** *type number* **Example:** `Router(config)# interface FastEthernet 1/0`	Specifies an interface type and number, and places the router in interface configuration mode.
Step 4	**rmon** {**native** | **promiscuous**} **Example:** `Router(config-if)# rmon native`	Configures RMON on Ethernet interfaces in native or promiscuous mode. • In the example, RMON is configured in the native mode.
Step 5	**exit** **Example:** `Router(config-if)# exit`	Exits the interface configuration mode and places the router in global configuration mode.
Step 6	**rmon queuesize** *size* **Example:** `Router(config)# rmon queuesize 128`	(Optional) Configures the size of the queue that holds packets for analysis by the RMON process.
Step 7	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode and enters privileged EXEC mode.
Step 8	`Router#` **show rmon** **Example:** `Router# show rmon`	Displays general RMON statistics.

Configuring RMON Event and Alarm Notifications

The following tasks describe how to configure RMON event and alarm notifications.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **rmon event number** [**log**] [**trap** *community*] [**description** *string*] [**owner** *string*]
4. **rmon alarm** *number variable interval* {**delta** | **absolute**} **rising-threshold** *value* [*event-number*] **falling-threshold** *value* [*event-number*] [**owner** *string*]
5. **rmon hc-alarms** *number variable interval* {**delta** | **absolute**} **rising-threshold** *value* [*event-number*] **falling-threshold** *value* [*event-number*] [**owner** *string*]
6. **exit**
7. **show rmon alarms**
8. **show rmon hc-alarms**
9. **show rmon events**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**rmon event number** [**log**] [**trap** *community*] [**description** *string*] [**owner** *string*]	Adds or removes an event (in the RMON event table) that is associated with an RMON event number.
Step 4	**rmon alarm** *number variable interval* {**delta** | **absolute**} **rising-threshold** *value* [*event-number*] **falling-threshold** *value* [*event-number*] [**owner** *string*] **Example:** `Router(config)# rmon alarm 10 ifEntry.20.1 20 delta rising-threshold 15 1 falling-threshold 0 owner owner1`	Configures an alarm on any MIB object.
Step 5	**rmon hc-alarms** *number variable interval* {**delta** | **absolute**} **rising-threshold** *value* [*event-number*] **falling-threshold** *value* [*event-number*] [**owner** *string*] **Example:** `Router(config)# rmon hc-alarms 2 ifInOctets.2 20 delta rising-threshold 2000 2 falling-threshold 1000 1 owner own`	(Optional) Configures an HC alarm on any MIB object.
Step 6	**exit** **Example:** `Router(config)# exit`	Exits the global configuration mode and enters the privileged EXEC mode.
Step 7	**show rmon alarms** **Example:** `Router# show rmon alarm`	Displays the RMON alarm table.
Step 8	**show rmon hc-alarms** **Example:** `Router# show rmon hc-alarms`	Displays the RMON HC alarm table.
Step 9	**show rmon events** **Example:** `Router# show rmon events`	Displays the RMON event table.

Configuring RMON Groups

The following tasks explain how to configure RMON groups by gathering RMON statistics for data types.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **rmon collection history controlEntry** *integer* [**owner** *ownername*] [**buckets** *bucket-number*] [**interval** *seconds*]
5. **rmon collection host controlEntry** *integer* [**owner** *ownername*]
6. **rmon collection matrix controlEntry** *integer* [**owner** *ownername*]
7. **rmon collection rmon1 controlEntry** *integer* [**owner** *ownername*]
8. **exit**
9. **rmon capture-userdata**
10. **exit**
11. **show rmon history**
12. **show rmon hosts**
13. **show rmon matrix**
14. **show rmon statistics**
15. **show rmon capture**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Router(config)# interface FastEthernet 1/0`	Specifies an interface type and number, and places the router in interface configuration mode.

	Command or Action	Purpose
Step 4	**rmon collection history controlEntry** *integer* [**owner** *ownername*] [**buckets** *bucket-number*] [**interval** *seconds*] **Example:** `Router(config-if)# rmon collection history controlEntry 20 owner john`	(Optional) Enables RMON history gathering on an interface.
Step 5	**rmon collection host controlEntry** *integer* [**owner** *ownername*] **Example:** `Router(config-if)# rmon collection host controlEntry 40 owner own1`	(Optional) Enables RMON MIB host collection group of statistics on an interface.
Step 6	**rmon collection matrix controlEntry** *integer* [**owner** *ownername*] **Example:** `Router(config-if)# rmon collection matrix controlEntry 25 owner john`	(Optional) Enables RMON MIB matrix group of statistics on an interface.
Step 7	**rmon collection rmon1 controlEntry** *integer* [**owner** *ownername*] **Example:** `Router(config-if)# rmon collection rmon1 controlEntry 30 owner john`	(Optional) Enables all possible autoconfigurable RMON MIB statistic collections on an interface.
Step 8	**exit** **Example:** `Router(config-if)# exit`	Exits the interface configuration mode and places the router in global configuration mode.
Step 9	**rmon capture-userdata** **Example:** `Router(config)# rmon capture-userdata`	Disables the packet zeroing feature that initializes the user payload portion of each RMON MIB packet.
Step 10	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode and enters privileged EXEC mode.
Step 11	**show rmon history** **Example:** `Router# show rmon history`	Displays the RMON history table.
Step 12	**show rmon hosts** **Example:** `Router# show rmon hosts`	Displays the RMON hosts table.

	Command or Action	Purpose
Step 13	**show rmon matrix** **Example:** `Router# show rmon matrix`	Displays the RMON matrix table and values associated with RMON variables.
Step 14	**show rmon statistics** **Example:** `Router# show rmon statistics`	Displays the RMON statistics table.
Step 15	**show rmon capture** **Example:** `Router# show rmon capture`	Displays the contents of the router's RMON capture table.

Configuration Examples for RMON Support

This section provides the following examples:

- Configuring RMON: Example, page 13
- Configuring RMON Event and Alarm Notifications: Example, page 13
- Configuring RMON Tables: Example, page 15

Configuring RMON: Example

The following example shows how to configure RMON with a queuesize of 100 packets in promiscuous mode:

```
Router> enable
Router# configure terminal
Router(config)# interface fastethernet 0/0
Router(config-if)# rmon promiscuous
Router(config-if)# exit
Router(config)# rmon queuesize 100
```

The following is a sample output from the **show rmon** command. All counters are from the time the router was initialized:

```
Router# show rmon

145678 packets input (34562 promiscuous), 0 drops
145678 packets processed, 0 on queue, queue utilization 15/100
```

Configuring RMON Event and Alarm Notifications: Example

The following example shows how to enable the **rmon event** global configuration command:

```
Router> enable
Router# configure terminal
Router(config)# rmon event 1 log trap eventtrap description "High ifOutErrors" owner
owner_a
```

This example creates RMON event number 1, which is defined as High ifOutErrors, and generates a log entry when the event is triggered by an alarm. The user owner_a owns the row that is created in the event table by this command. This example also generates an SNMP trap when the event is triggered.

The following is a sample output from the **show rmon events** command:

```
Router# show rmon events

Event 1 is active, owned by owner_a
 Description is High ifOutErrors
 Event firing causes log and trap to community rmonTrap, last fired 00:00:00
```

The following example shows how to configure an RMON alarm using the **rmon alarm** global configuration command:

```
Router> enable
Router# configure terminal
Router(config)# rmon alarm 10 ifEntry.20.1 20 delta rising-threshold 15 1
falling-threshold 0 owner owner_a
```

This example configures RMON alarm number 10. The alarm monitors the MIB variable ifEntry.20.1 once every 20 seconds until the alarm is disabled, and checks the change in the rise or fall of the variable. If the ifEntry.20.1 value shows a MIB counter increase of 15 or more, such as from 100000 to 100015, the alarm is triggered. The alarm in turn triggers event number 1, which is configured with the **rmon event** command. Possible events include a log entry or an SNMP trap. If the ifEntry.20.1 value changes by 0, the alarm is reset and can be triggered again.

The following is sample output from the **show rmon alarms** command

```
Router# show rmon alarms
Alarm 2 is active, owned by owner_a
 Monitors ifEntry.20.1.20 every 20 seconds
 Taking delta samples, last value was 0
 Rising threshold is 15, assigned to event 12
 Falling threshold is 0, assigned to event 0
 On startup enable rising or falling alarm
```

The following example shows how to configure an RMON HC alarm using the **rmon hc-alarms** global configuration command:

```
Router> enable
Router# configure terminal
Router(config)# rmon hc-alarms 2 ifInOctets.2 20 delta rising-threshold 2000 2
falling-threshold 1000 1 owner own
```

This example configures RMON HC alarm number 2. The alarm monitors the MIB variable ifInOctets.2 once every 20 seconds until the alarm is disabled, and checks the change in the rise or fall of the variable. If the ifInOctets.2 value shows a MIB counter increase of 2000 or more, such as from 100000 to 100015, the alarm is triggered. The alarm in turn triggers event number 2, which is configured with the **rmon event** command. Possible events include a log entry or a Simple Network Management Protocol (SNMP) trap. If the ifInOctets.2 value changes by 1000 (falling threshold is 1000), the alarm is reset and can be triggered again.

To display the contents of the RMON HC alarm table of the router, use the **show rmon hc-alarms** command in privileged EXEC mode. The following is sample output:

```
Router# show rmon hc-alarms

Router#show rmon hc-alarms
 Monitors ifInOctets.1 every 20 second(s)
 Taking absolute samples, last value was 0
 Rising threshold Low is 4096, Rising threshold Hi is 0,
               assigned to event 0
 Falling threshold Low is 1280, Falling threshold Hi is 0,
               assigned to event 0
 On startup enable rising or falling alarm
```

Configuring RMON Tables: Example

The following example shows how to enable the RMON collection matrix group of statistics with an ID number of 25 and specifies john as the owner:

```
Router> enable
Router# configure terminal
Router(config)# interface fastethernet 0/0
Router(config-if)# rmon collection matrix controlEntry 25 owner john
```

To view values associated with RMON variables, enter the **show rmon matrix** privileged EXEC command (Cisco 2500 series routers and Cisco AS5200 access servers only). The following is a sample output:

```
Router# show rmon matrix

Matrix 1 is active and owned by john
Monitors controlEntry
Table size is 25, last time an entry was deleted was at 11:18:09
Source addr is 0000.0c47.007b, dest addr is ffff.ffff.ffff
Transmitted 2 pkts, 128 octets, 0 errors
Source addr is 0000.92a8.319e, dest addr is 0060.5c86.5b82
Transmitted 2 pkts, 384 octets, 1 error
```

Additional References

The following sections provide references related to the Configuring RMON Support feature.

Related Documents

Related Topic	Document Title
Cisco IOS commands	*Cisco IOS Master Commands List, All Releases*
CNS commands: complete command syntax, command mode, command history, defaults, usage guidelines, and examples	*Cisco IOS Network Management Command Reference 3.0*

Standards

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

MIBs

MIB	MIBs Link
• RMON MIB • HC-Alarm MIB	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
RFC 1757	*Remote Network Monitoring Management Information Base*
RFC 2021	*Remote Network Monitoring Management Information Base Version 2 using SMIv2*
RFC 3434	*Remote Monitoring MIB Extensions for High Capacity Alarms*

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for Configuring RMON Support

Table 4 lists the release history for this feature and provides links to specific configuration information.

For information on a feature in this technology that is not documented here, see the Configuring RMON Features Roadmap.

Not all commands may be available in your Cisco IOS software release. For release information about a specific command, see the command reference documentation.

Use Cisco Feature Navigator to find information about platform support and software image support. Cisco Feature Navigator enables you to determine which Cisco IOS and Catalyst OS images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Note Table 4 lists only the Cisco IOS software release that introduced support for a given feature in a given Cisco IOS software release train. Unless noted otherwise, subsequent releases of that Cisco IOS software release train also support that feature.

Table 4 Feature Information for Configuring RMON Support

Feature Name	Releases	Feature Information
RMON Full	11.2	The RMON Full feature identifies activity on individual nodes and helps monitor all nodes and their interaction on a LAN segment. Used in conjunction with the SNMP agent in a router, RMON can be used to view both traffic that flows through the router and segment traffic not necessarily destined for the router. The following sections provide information about this feature: • Information About Configuring RMON Support, page 2 • Configuration Examples for RMON Support, page 13
RMON Events and Alarms	11.2 Cisco IOS XE Release 2.1	The RMON Events and Alarms feature introduces the ability to combine RMON alarms and events (classes of messages that indicate traffic violations and various unusual occurrences over a network) with existing MIBs allows you to choose where proactive monitoring will occur. In Cisco IOS XE Release 2.1, this feature was introduced on Cisco ASR 1000 series routers. The following sections provide information about this feature: • RMON Event and Alarm Notifications, page 4 • Configuration Examples for RMON Support, page 13 The following commands were introduced: **rmon alarm**, **rmon event**, **rmon queuesize**.

Table 4 Feature Information for Configuring RMON Support (continued)

Feature Name	Releases	Feature Information
Remote Monitoring MIB Update	12.0(5)T	The RMON Rewrite feature updated the Remote Monitoring MIB to improve performance and available features. The following sections provide information about this feature: • RMON MIB, page 5 • Configuring RMON Groups, page 9 • Configuration Examples for RMON Support, page 13 The following commands were introduced: **rmon capture-userdata**, **rmon collection history**, **rmon collection host**, **rmon collection matrix**, **rmon collection rmon1**, **show rmon capture**, **show rmon filter**, **show rmon hosts**, **show rmon matrix**.
HC Alarm MIB	12.2(33)SXI 12.2(33)SRE	The HC Alarm MIB feature provides an extension to the RMON-1 Alarm group table objects which was used to support counter 32 objects for threshold capabilities. The HC Alarm MIB adds support to threshold capabilities for counter 64 objects. The following sections provide information about this feature: • HC Alarm MIB, page 6 • Configuration Examples for RMON Support, page 13 The following commands were introduced: **rmon hc-alarms**, **show rmon hc-alarms**.
RMON MIB enhancement to support 64 bit counters	12.2(33)SXI 12.2(33)SRE	RMON MIB enhancement to support 64 bit counters features provides support for the ability to poll 64 bit counters. The following section provides information about this feature: • RMON MIB, page 5

SNMP Support

Configuring SNMP Support

First Published: December 20, 2006
Last Updated: November 20, 2009

Simple Network Management Protocol (SNMP) is an application-layer protocol that provides a message format for communication between SNMP managers and agents. SNMP provides a standardized framework and a common language that is used for monitoring and managing devices in a network.

This document discusses how to enable an SNMP agent on a Cisco device and how to control the sending of SNMP notifications from the agent. For information about using SNMP management systems, see the appropriate documentation for your network management system (NMS) application.

For a complete description of the router monitoring commands mentioned in this document, see the *Cisco IOS Network Management Command Reference*. To locate documentation of other commands that appear in this document, use the *Cisco IOS Command Reference Master Index* or search online.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the "Feature Information for Configuring SNMP Support" section on page 75.

Use Cisco Feature Navigator to find information about platform support and Cisco IOS and Catalyst OS software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

Contents

Restrictions for Configuring SNMP Support

Not all Cisco platforms are supported on the features described in this module. Use Cisco Feature Navigator to find information about platform support and Cisco IOS software image support.

Information About Configuring SNMP Support

To configure SNMP support on your network, you should understand the following concepts:

Components of SNMP

SNMP is an application-layer protocol that provides a message format for communication between SNMP managers and agents. SNMP provides a standardized framework and a common language used for monitoring and managing devices in a network.

The SNMP framework is made up of three parts:

- SNMP manager
- SNMP agent
- MIB

SNMP Manager

The SNMP manager is a system that controls and monitors the activities of network hosts using SNMP. The most common managing system is an NMS. The term NMS can be applied either to a dedicated device used for network management or to the applications used on such a device. Several network management applications are available for use with SNMP and range from simple command-line applications to applications that use GUIs, such as the CiscoWorks2000 products.

SNMP Agent

The SNMP agent is the software component within a managed device that maintains the data for the device and reports this data, as needed, to managing systems. The agent resides on the routing device (router, access server, or switch). To enable an SNMP agent on a Cisco routing device, you must define the relationship between the manager and the agent.

Note Although it is possible to configure a Cisco router to be an SNMP agent, this practice is not recommended. Commands that an agent needs to control the SNMP process are available through the Cisco IOS command-line interface (CLI) without additional configuration.

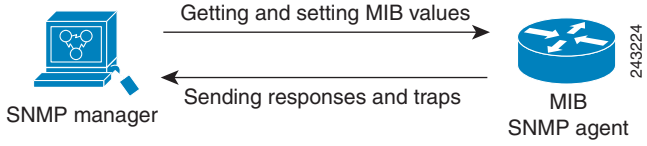
MIB

A MIB is a virtual information storage area for network management information and consists of collections of managed objects. Within a MIB are collections of related objects defined in MIB modules. MIB modules are written in the SNMP MIB module language, as defined in STD 58, RFC 2578, RFC 2579, and RFC 2580 (see the "MIBs and RFCs" section for an explanation of RFC and STD documents). Individual MIB modules are also referred to as MIBs; for example, the Interfaces Group MIB (IF-MIB) is a MIB module within the MIB on your system.

An SNMP agent contains MIB variables whose values the SNMP manager can request or change through Get or Set operations. A manager can get a value from an agent or store a value in that agent. The agent gathers data from the MIB, the repository for information about device parameters and network data. The agent can also respond to manager requests to get or set data.

Figure 1 illustrates the communications between the SNMP manager and agent. A manager sends an agent requests to get and set MIB values. The agent responds to these requests. Independent of this interaction, the agent can send the manager unsolicited notifications (traps or informs) to notify the manager about network conditions.

Figure 1 Communication Between an SNMP Agent and Manager



SNMP Operations

SNMP applications perform the following operations to retrieve data, modify SNMP object variables, and send notifications:

- Get
- Set
- Send notifications

SNMP Get

The SNMP get operation is performed by an NMS to retrieve SNMP object variables. There are three types of get operations:

- get—Retrieves the exact object instance from the SNMP agent.
- getNext—Retrieves the next object variable, which is a lexicographical successor to the specified variable.
- getBulk—Retrieves a large amount of object variable data, without the need for repeated getNext operations.

SNMP Set

The SNMP set operation is performed by an NMS to modify the value of an object variable.

SNMP Notifications

A key feature of SNMP is its capability to generate unsolicited notifications from an SNMP agent.

Traps and Informs

Unsolicited (asynchronous) notifications can be generated as traps or inform requests (informs). Traps are messages alerting the SNMP manager to a condition on the network. Informs are traps that include a request for confirmation of receipt from the SNMP manager. Notifications can indicate improper user authentication, restarts, the closing of a connection, loss of connection to a neighbor router, or other significant events.

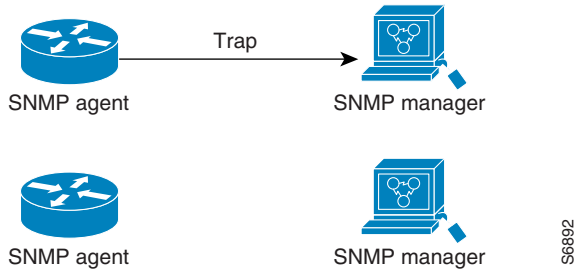
Traps are less reliable than informs because the receiver does not send an acknowledgment when it receives a trap. The sender does not know if the trap was received. An SNMP manager that receives an inform acknowledges the message with an SNMP response protocol data unit (PDU). If the sender never receives a response, the inform can be sent again. Thus, informs are more likely to reach their intended destination.

Traps are often preferred even though they are less reliable because informs consume more resources in the router and in the network. Unlike a trap, which is discarded as soon as it is sent, an inform must be held in memory until a response is received or the request times out. Also, traps are sent only once, whereas an inform may be resent several times. The retries increase traffic and contribute to higher overhead on the network. Use of traps and informs requires a trade-off between reliability and resources. If it is important that the SNMP manager receives every notification, use informs, but if traffic volume or memory usage are concerns and receipt of every notification is not required, use traps.

Figure 2 through Figure 5 illustrate the differences between traps and informs.

Figure 2 shows that an agent successfully sends a trap to an SNMP manager. Although the manager receives the trap, it does not send an acknowledgment. The agent has no way of knowing that the trap reached its destination.

Figure 2 Trap Successfully Sent to SNMP Manager



In Figure 3, the agent successfully sends an inform to the manager. When the manager receives the inform, a response is sent to the agent and the agent knows that the inform reached its destination. Notice that in this example the traffic generated is twice as much as in the interaction shown in Figure 2.

Figure 3 Inform Request Successfully Sent to SNMP Manager

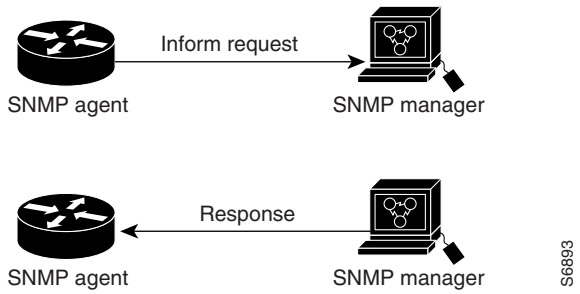


Figure 4 shows an agent sending a trap to a manager that the manager does not receive. The agent has no way of knowing that the trap did not reach its destination. The manager never receives the trap because traps are not resent.

Figure 4 Trap Unsuccessfully Sent to SNMP Manager

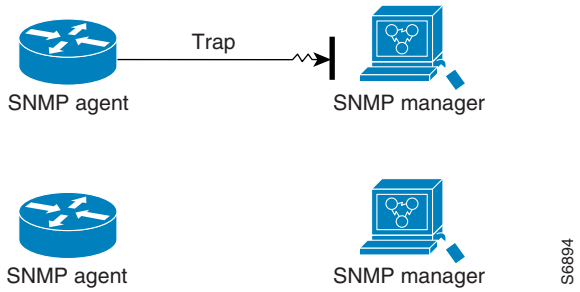
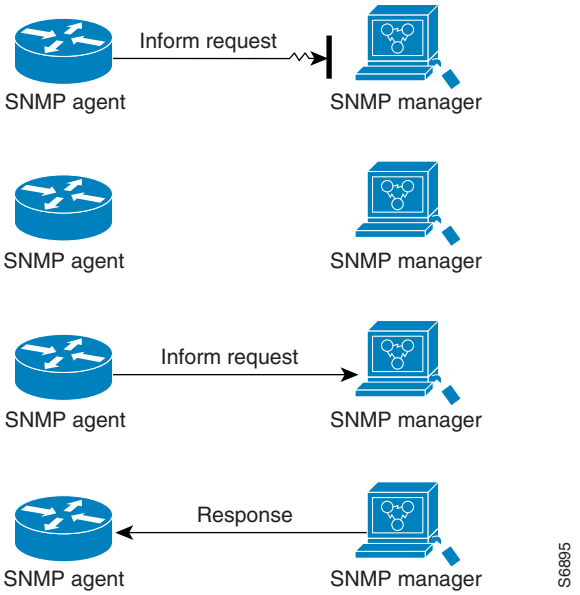


Figure 5 shows an agent sending an inform to a manager that does not reach the manager. Because the manager did not receive the inform, it does not send a response. After a period of time, the agent resends the inform. The manager receives the inform from the second transmission and replies. In this example, more traffic is generated than in the scenario shown in Figure 4 but the notification reaches the SNMP manager.

Figure 5 Inform Unsuccessfully Sent to SNMP Manager



MIBs and RFCs

MIB modules typically are defined in RFC documents submitted to the Internet Engineering Task Force (IETF), an international standards body. RFCs are written by individuals or groups for consideration by the Internet Society and the Internet community as a whole, usually with the intention of establishing a recommended Internet standard. Before being given RFC status, recommendations are published as Internet Draft (I-D) documents. RFCs that have become recommended standards are also labeled as standards (STD) documents. You can learn about the standards process and the activities of the IETF at the Internet Society website at http://www.isoc.org. You can read the full text of all RFCs, I-Ds, and STDs referenced in Cisco documentation at the IETF website at http://www.ietf.org.

The Cisco implementation of SNMP uses the definitions of MIB II variables described in RFC 1213 and definitions of SNMP traps described in RFC 1215.

Cisco provides its own private MIB extensions with every system. Cisco enterprise MIBs comply with the guidelines described in the relevant RFCs unless otherwise noted in the documentation. You can find the MIB module definition files and list of MIBs supported on each Cisco platform on the Cisco MIB website on Cisco.com.

Versions of SNMP

Cisco IOS software supports the following versions of SNMP:

- SNMPv1—Simple Network Management Protocol: a full Internet standard, defined in RFC 1157. (RFC 1157 replaces the earlier versions that were published as RFC 1067 and RFC 1098.) Security is based on community strings.
- SNMPv2c—The community string-based Administrative Framework for SNMPv2. SNMPv2c (the "c" is for "community") is an experimental Internet protocol defined in RFC 1901, RFC 1905, and RFC 1906. SNMPv2c is an update of the protocol operations and data types of SNMPv2p (SNMPv2 Classic) and uses the community-based security model of SNMPv1.

- SNMPv3—Version 3 of SNMP. SNMPv3 is an interoperable standards-based protocol defined in RFCs 3413 to 3415. SNMPv3 provides secure access to devices by authenticating and encrypting packets over the network.

  The security features provided in SNMPv3 are as follows:

  - Message integrity—Ensuring that a packet has not been tampered with in transit.
  - Authentication—Determining that the message is from a valid source.
  - Encryption—Scrambling the contents of a packet to prevent it from being learned by an unauthorized source.

Both SNMPv1 and SNMPv2c use a community-based form of security. The community of SNMP managers able to access the agent MIB is defined by an IP address access control list (ACL) and password.

SNMPv2c support includes a bulk retrieval mechanism and detailed error message reporting to management stations. The bulk retrieval mechanism supports the retrieval of tables and large quantities of information, minimizing the number of round trips required. The SNMPv2c improved error handling support includes expanded error codes that distinguish different types of errors; these conditions are reported through a single error code in SNMPv1. The following three types of exceptions are also reported: no such object, no such instance, and end of MIB view.

SNMPv3 is a security model in which an authentication strategy is set up for a user and the group in which the user resides. A security level is the permitted level of security within a security model. A combination of a security model and a security level determines which security mechanism is employed when handling an SNMP packet.

Three security models are available: SNMPv1, SNMPv2c, and SNMPv3. Table 1 lists the combinations of security models and levels and their meanings.

Table 1 SNMP Security Models and Levels

Model	Level	Authentication	Encryption	What Happens
v1	noAuthNoPriv	Community String	No	Uses a community string match for authentication.
v2c	noAuthNoPriv	Community String	No	Uses a community string match for authentication.
v3	noAuthNoPriv	Username	No	Uses a username match for authentication.
v3	authNoPriv	Message Digest 5 (MD5) or Secure Hash Algorithm (SHA)	No	Provides authentication based on the HMAC-MD5 or HMAC-SHA algorithms.
v3	authPriv	MD5 or SHA	Data Encryption Standard (DES)	Provides authentication based on the HMAC-MD5 or HMAC-SHA algorithms. Provides DES 56-bit encryption in addition to authentication based on the CBC-DES (DES-56) standard.

Note SNMPv2p (SNMPv2 Classic) is not supported in Cisco IOS Release 11.2 and later releases. SNMPv2c replaces the Party-based Administrative and Security Framework of SNMPv2p with a Community-based Administrative Framework. SNMPv2c retained the bulk retrieval and error handling capabilities of SNMPv2p.

You must configure an SNMP agent to use the version of SNMP supported by the management station. An agent can communicate with multiple managers, however, and you can configure Cisco IOS software to support communications with one management station using the SNMPv1 protocol, one using the SNMPv2c protocol, and another using SMNPv3.

SNMPv3 supports RFCs 1901 to 1908, 2104, 2206, 2213, 2214, and 2271 to 2275. For additional information about SNMPv3, see RFC 2570, *Introduction to Version 3 of the Internet-standard Network Management Framework* (this is not a standards document).

Detailed Interface Registration Information

The Interface Index Display for SNMP feature introduces new commands and command modifications that allow advanced users of SNMP to view information about the interface registrations directly on the managed agent. You can display MIB information from the agent without using an external NMS.

Note For the purposes of this document, the agent is a routing device running Cisco IOS software.

This feature addresses three objects in the Interfaces MIB: ifIndex, ifAlias, and ifName. For a complete definition of these objects, see the IF-MIB.my file available from the Cisco SNMPv2 MIB website at ftp://ftp.cisco.com/pub/mibs/v2/.

Interface Index

The ifIndex object (ifEntry 1) is called the Interface Index. The Interface Index is a unique value greater than zero that identifies each interface or subinterface on the managed device. This value becomes the interface index identification number.

The CLI command **show snmp mib ifmib ifindex** allows you to view the SNMP Interface Index Identification numbers assigned to interfaces and subinterfaces. An NMS is not required.

Interface Alias

The ifAlias object (ifXEntry 18) is called the Interface Alias. The Interface Alias is a user-specified description of an interface used for SNMP network management. The ifAlias is an object in the Interfaces Group MIB (IF-MIB) that can be set by a network manager to “name” an interface. The ifAlias value for an interface or subinterface can be set using the **description** command in interface configuration mode or subinterface configuration mode or by using a Set operation from an NMS. Previously, ifAlias descriptions for subinterfaces were limited to 64 characters. (The OLD-CISCO-INTERFACES-MIB allows up to 255 characters for the locIfDescr MIB variable, but this MIB does not support subinterfaces.) A new CLI command, **snmp ifmib ifalias long**, configures the system to handle IfAlias descriptions of up to 256 characters. IfAlias descriptions appear in the output of the CLI **show interfaces** command.

Interface Name

The ifName object (ifXEntry 1) is the textual name of the interface. The purpose of the ifName object is to cross reference the CLI representation of a given interface. The value of this object is the name of the interface as assigned by the local device and is suitable for use in CLI commands. If there is no local name or this object is otherwise not applicable, this object contains a zero-length string. No commands introduced by this feature affect the ifName object, but it is discussed here to show its relation to the ifIndex and ifAlias objects.

The **show snmp mib** command shows all objects in the MIB on a Cisco device (similar to a mibwalk). The objects in the MIB tree are sorted using lexical ordering, meaning that object identifiers are sorted in sequential, numerical order. Lexical ordering is important when using the GetNext operation from an NMS because these operations take an object identifier (OID) or a partial OID as input and return the next object from the MIB tree based on the lexical ordering of the tree.

SNMP Support for VPNs

The SNMP Support for VPNs feature allows SNMP traps and informs to be sent and received using virtual private network (VPN) routing/forwarding (VRF) tables. In particular, this feature adds support to Cisco IOS software for the sending and receiving of SNMP traps and informs specific to individual VPNs.

A VPN is a network that provides high connectivity transfers on a shared system with the same usage guidelines as a private network. A VPN can be built on the Internet over IP, Frame Relay, or ATM networks.

A VRF stores per-VPN routing data. It defines the VPN membership of a customer site attached to the network access server (NAS). A VRF consists of an IP routing table, a derived Cisco Express Forwarding table, and guidelines and routing protocol parameters that control the information that is included in the routing table.

The SNMP Support for VPNs feature provides configuration commands that allow users to associate SNMP agents and managers with specific VRFs. The specified VRF is used for sending SNMP traps and informs and responses between agents and managers. If a VRF is not specified, the default routing table for the VPN is used.

Support for VPNs allows you to configure an SNMP agent to accept only SNMP requests from a certain set of VPNs. With this configuration, service providers can provide network management services to their customers, so customers can manage all user VPN devices.

Interface IfIndex Persistence

Interface Index (IfIndex) is one of the most commonly used identifiers SNMP-based network management applications. IfIndex is a unique identifying number associated with a physical or logical interface; as far as most software is concerned, the ifIndex is the name of the interface.

Although there is no requirement in the relevant RFCs that the correspondence between particular ifIndex values and their interfaces be maintained across reboots, applications such as device inventory, billing, and fault detection increasingly depend on the maintenance of this correspondence.

This feature adds support for an ifIndex value that can persist across reboots, allowing users to avoid the workarounds previously required for consistent interface identification.

It is currently possible to poll the router at regular intervals to correlate the interfaces to the ifIndex, but it is not practical to poll this interface constantly. If this data is not correlated constantly, however, the data may be made invalid because of a reboot or the insertion of a new card into the router in between polls. Therefore, ifIndex persistence is the only way to guarantee data integrity.

IfIndex persistence means that the mapping between the ifDescr object values and the ifIndex object values (generated from the IF-MIB) will be retained across reboots.

Benefits of Interface Index Persistence

Association of Interfaces with Traffic Targets for Network Management

The Interface Index Persistence feature allows for greater accuracy when collecting and processing network management data by uniquely identifying input and output interfaces for traffic flows and SNMP statistics. Relating each interface to a known entity (such as an ISP customer) allows network management data to be more effectively utilized.

Accuracy for Mediation, Fault Detection, and Billing

Network data is increasingly being used worldwide for usage-based billing, network planning, policy enforcement, and trend analysis. The ifIndex information is used to identify input and output interfaces for traffic flows and SNMP statistics. Inability to reliably relate each interface to a known entity, such as a customer, invalidates the data.

MIB Persistence

The MIB Persistence features allow the SNMP data of a MIB to be persistent across reloads; that is, MIB information retains the same set object values each time a networking device reboots. MIB Persistence is enabled by issuing the **snmp mib persist** command, and the MIB data of all MIBs that have had persistence enabled using this command is then written to NVRAM by issuing the **write mib-data** command. All modified MIB data must be written to NVRAM using the **write mib-data** command.

Both Event and Expression MIBs allow you to configure a value for an object and to set up object definitions. Both also allow rows of data to be modified while the row is in an active state.

Scalar objects are stored every time they are changed, and table entries are stored only if the row is in an active state. The Event MIB has two scalar objects and nine tables to be persisted into NVRAM. Following are the tables:

- mteEventNotificationTable
- mteEventSetTable
- mteEventTable
- mteObjectsTable
- mteTriggerBooleanTable
- mteTriggerDeltaTable
- mteTriggerExistenceTable
- mteTriggerTable
- mteTriggerThresholdTable

The Expression MIB has two scalar objects and three tables to be stored in NVRAM. The scalar objects are expResourceDeltaMinimum and expResourceDeltaWildcardInstanceMaximum. Following are the tables:

- expExpressionTable
- expNameTable
- expObjectTable

Writing MIB data to NVRAM may take several seconds. The length of time depends on the amount of MIB data.

Event MIB Persistence and Expression MIB Persistence both allow MIB objects to be saved from reboot to reboot, allowing long-term monitoring of specific devices and interfaces and configurations of object values that are preserved across reboots.

Circuit Interface Identification Persistence

The Circuit Interface MIB (CISCO-CIRCUIT-INTERFACE-MIB) provides a MIB object (cciDescr) that can be used to identify individual circuit-based interfaces for SNMP monitoring. The Circuit Interface Identification Persistence for SNMP feature maintains this user-defined name of the circuit across reboots, allowing the consistent identification of circuit interfaces. Circuit Interface Identification Persistence is enabled using the **snmp mib persist circuit** global configuration command.

Cisco IOS Release 12.2(2)T introduces the Circuit Interface Identification Persistence for SNMP feature. The Circuit Interface MIB (CISCO-CIRCUIT-INTERFACE-MIB) provides a MIB object (cciDescr) that can be used to identify individual circuit-based interfaces for SNMP monitoring. The Cisco Circuit Interface MIB was introduced in Cisco IOS Release 12.1(3)T.

The Circuit Interface Identification Persistence for SNMP feature maintains the user-defined name of the circuit (defined in the cciDescr object) across reboots, allowing for the consistent identification of circuits.

The Circuit Interface Identification Persistence for SNMP feature is a supplement to the Interface Index Persistence feature introduced in Cisco IOS Release 12.1(3)T and in Cisco IOS Release 12.0(11)S. Circuit Interface Identification Persistence is enabled with the **snmp mib persist circuit** global configuration command. Use this command if you need to consistently identify circuits using SNMP across reboots. This command is disabled by default because this feature uses NVRAM.

In addition, the **show snmp mib ifmib ifindex** EXEC mode command allows you to display the Interfaces MIB ifIndex values directly on your system without an NMS; the **show snmp mib** EXEC mode command allows you to display a list of the MIB module identifiers registered directly on your system with an NMS. And the **snmp ifmib ifalias long** command allows you to specify a description for interfaces or subinterfaces of up to 256 characters in length. Prior to the introduction of this command, ifAlias descriptions for SNMP management were limited to 64 characters.

Event MIB

The Event MIB provides the ability to monitor MIB objects on a local or remote system using SNMP and initiate simple actions whenever a trigger condition is met; for example, an SNMP trap can be generated when an object is modified. When the notifications are triggered through events, the NMS does not need to constantly poll managed devices to track changes.

By allowing the SNMP notifications to take place only when a specified condition is met, Event MIB reduces the load on affected devices and improves the scalability of network management solutions.

The Event MIB operates based on event, object lists configured for the event, event action, trigger, and trigger test.

Events

The event table defines the activities to be performed when an event is triggered. These activities include sending a notification and setting a MIB object. The event table has supplementary tables for additional objects that are configured according to event action. If the event action is set to notification, notifications are sent out whenever the object configured for that event is modified.

Object List

The object table lists objects that can be added to notifications based on trigger, trigger test type, or the event that sends a notification. The Event MIB allows wildcarding, which enables you to monitor multiple instances of an object. To specify a group of object identifiers, you can use the wildcard option.

Trigger

The trigger table defines conditions to trigger events. The trigger table lists the objects to be monitored and associates each trigger with an event. An event occurs when a trigger is activated. To create a trigger, you should configure a trigger entry in the mteTriggerTable of the Event MIB. This trigger entry specifies the object identifier of the object to be monitored. Each trigger is configured to monitor a single object or a group of objects specified by a wildcard (*). The Event MIB process checks the state of the monitored object at specified intervals.

Trigger Test

The trigger table has supplementary tables for additional objects that are configured based on the type of test performed for a trigger. For each trigger entry type such as existence, theshold, or Boolean, the corresponding tables (existence, threshold, and Boolean tables) are populated with the information required to perform the test. Event MIB allows you to set event triggers based on existence, threshold, and Boolean trigger types. When the specified test on an object returns a value of *true*, the trigger is activated. You can configure Event MIB to send out notifications to the interested host when a trigger is activated.

Expression MIB

The Expression MIB allows you to create expressions based on a combination of objects. The expressions are evaluated according to the sampling method. The Expression MIB supports the following types of object sampling:

- Absolute
- Delta
- Changed

If there are no delta or change values in an expression, the expression is evaluated when a requester attempts to read the value of expression. In this case, all requesters get a newly calculated value.

For expressions with delta or change values, evaluation is performed for every sampling. In this case, requesters get the value as of the last sample period.

Absolute Sampling

Absolute sampling uses the value of the MIB object during sampling.

Delta Sampling

Delta sampling is used for expressions with counters that are identified based on delta (difference) from one sample to the next. Delta sampling requires the application to do continuous sampling, because it uses the value of the last sample.

Changed Sampling

Changed sampling uses the changed value of the object since the last sample.

SNMP Notification Logging

Systems that support SNMP often need a mechanism for recording notification information. This mechanism protects against notifications being lost because they exceeded retransmission limits. The Notification Log MIB provides a common infrastructure for other MIBs in the form of a local logging function. The SNMP Notification Logging feature adds Cisco IOS CLI commands to change the size of the notification log, to set the global ageout value for the log, and to display logging summaries at the command line. The Notification Log MIB improves notification tracking and provides a central location for tracking all MIBs.

Note The Notification Log MIB supports notification logging on the default log only.

How to Configure SNMP Support

There is no specific command that you use to enable SNMP. The first **snmp-server** command that you enter enables the supported versions of SNMP. All other configurations are optional.

This section contains the following procedures:

- Configuring System Information, page 14 (optional)
- Configuring SNMP Versions 1 and 2, page 15 (optional)
- Configuring SNMP Version 3, page 20 (optional)
- Configuring a Router as an SNMP Manager, page 24 (optional)
- Enabling the SNMP Agent Shutdown Mechanism, page 27 (optional)
- Defining the Maximum SNMP Agent Packet Size, page 28 (optional)
- Limiting the Number of TFTP Servers Used via SNMP, page 29 (optional)
- Disabling the SNMP Agent, page 30 (optional)
- Configuring SNMP Notifications, page 31 (optional)
- Configuring Interface Index Display and Interface Indexes and Long Name Support, page 38 (optional)
- Configuring SNMP Support for VPNs, page 41 (optional)

- Configuring Interface IfIndex Persistence, page 43 (optional)
- Configuring MIB Persistence, page 45 (optional)
- Configuring Event MIB Using SNMP, page 48 (optional)
- Configuring Event MIB Using CLI, page 50 (optional)
- Configuring Expression MIB Using SNMP, page 61 (optional)
- Configuring Expression MIB using CLI, page 63 (optional)

Configuring System Information

You can set the system contact, location, and serial number of the SNMP agent so that these descriptions can be accessed through the configuration file. Although the configuration steps described in this section are optional, configuring the basic information is recommended because it may be useful when troubleshooting your configuration. In addition, the first **snmp-server** command that you issue enables SNMP on the device.

Perform this task as needed.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp-server contact** *text*
4. **snmp-server location** *text*
5. **snmp-server chassis-id** *number*
6. **exit**
7. **show snmp contact**
8. **show snmp location**
9. **show snmp chassis**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**snmp-server contact** *text* **Example:** `Router(config)# snmp-server contact NameOne`	Sets the system contact string.

	Command or Action	Purpose
Step 4	**snmp-server location** *text* **Example:** `Router(config)# snmp-server location LocationOne`	Sets the system location string.
Step 5	**snmp-server chassis-id** *number* **Example:** `Router(config)# snmp-server chassis-id 015A619T`	Sets the system serial number.
Step 6	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode.
Step 7	**show snmp contact** **Example:** `Router# show snmp contact`	(Optional) Displays the contact strings configured for the system.
Step 8	**show snmp location** **Example:** `Router# show snmp location`	(Optional) Displays the location string configured for the system.
Step 9	**show snmp chassis** **Example:** `Router# show snmp chassis`	(Optional) Displays the system serial number.

Configuring SNMP Versions 1 and 2

When you configure SNMP versions 1 and 2, you can optionally create or modify views for community strings to limit which MIB objects an SNMP manager can access.

Perform the following tasks when configuring SNMP version 1 or version 2.

- Creating or Modifying an SNMP View Record, page 16 (optional)
- Creating or Modifying Access Control for an SNMP Community, page 17 (required)

Prerequisites

- An established SNMP community string that defines the relationship between the SNMP manager and the agent
- A host defined to be the recipient of SNMP notifications

Creating or Modifying an SNMP View Record

You can assign views to community strings to limit which MIB objects an SNMP manager can access. You can use a predefined view or create your own view. If you are using a predefined view or no view at all, skip this task.

Perform this task to create or modify an SNMP view record.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp-server view** *view-name oid-tree* {**included** | **excluded**}
4. **no snmp-server view** *view-name oid-tree* {**included** | **excluded**}
5. **exit**
6. **show snmp view**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**snmp-server view** *view-name oid-tree* {**included** | **excluded**} **Example:** `Router(config)# snmp-server view mib2 mib-2 included`	Creates a view record. • In this example, the mib2 view that includes all objects in the MIB-II subtree is created. **Note** You can use this command multiple times to create the same view record. If a view record for the same OID value is created multiple times, the latest entry of the object identifier takes precedence.
Step 4	**no snmp-server view** *view-name oid-tree* {**included** | **excluded**} **Example:** `Router(config)# no snmp-server view mib2 mib-2 included`	Removes a server view.

	Command or Action	Purpose
Step 5	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode.
Step 6	**show snmp view** **Example:** `Router# show snmp view`	(Optional) Displays a view of the MIBs associated with SNMP.

Examples

The following example shows the SNMP view for the system.1.0 OID tree:

```
Router# show snmp view

test system.1.0 - included nonvolatile active
*ilmi system - included permanent active
*ilmi atmForumUni - included permanent active
v1default iso - included permanent active
v1default internet - included permanent active
v1default snmpUsmMIB - excluded permanent active
v1default snmpVacmMIB - excluded permanent active
v1default snmpCommunityMIB - excluded permanent active
v1default ciscoIpTapMIB - excluded permanent active
v1default ciscoMgmt.395 - excluded permanent active
v1default ciscoTap2MIB - excluded permanent active
```

Creating or Modifying Access Control for an SNMP Community

Use an SNMP community string to define the relationship between the SNMP manager and the agent. The community string acts like a password to regulate access to the agent on the router. Optionally, you can specify one or more of the following characteristics associated with the string:

- An access list of IP addresses of the SNMP managers that are permitted to use the community string to gain access to the agent.
- A MIB view, which defines the subset of all MIB objects accessible to the given community.
- Read and write or read-only permission for the MIB objects accessible to the community.

Perform this task to create or modify a community string.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp-server community** *string* [**view** *view-name*] [**ro** | **rw**] [**ipv6** *nacl*] [*access-list-number*]
4. **no snmp-server community** *string*
5. **exit**
6. **show snmp community**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**snmp-server community** *string* [**view** *view-name*] [**ro** | **rw**] [**ipv6** *nacl*] [*access-list-number*] **Example:** `Router(config)# snmp-server community comaccess ro 4`	Defines the community access string. • You can configure one or more community strings.
Step 4	**no snmp-server community** *string* **Example:** `Router(config)# no snmp-server community comaccess`	Removes the community string from the configuration.
Step 5	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode.
Step 6	**show snmp community** **Example:** `Router# show snmp community`	(Optional) Displays the community access strings configured for the system.

Examples

The following example shows the community access strings configured to enable access to the SNMP manager:

```
Router# show snmp community

Community name: private
Community Index: private
Community SecurityName: private
storage-type: nonvolatile        active

Community name: private@1
Community Index: private@1
Community SecurityName: private
storage-type: read-only  active

Community name: public
Community Index: public
Community SecurityName: public
storage-type: nonvolatile        active
```

Configuring a Recipient of an SNMP Trap Operation

SNMP traps are unreliable because the receiver does not send acknowledgments when it receives traps. The sender does not know if the traps were received. However, a SNMP entity that receives an inform acknowledges the message with a SNMP response protocol data unit (PDU). If the sender never receives the response, the inform can be sent again. Thus, informs are more likely to reach their intended destination.

Compared to traps, informs consume more resources in the agent and in the network. Unlike a trap, which is discarded as soon as it is sent, an inform must be held in memory until a response is received or the request times out. Also, traps are sent only once; an inform may be sent several times. The retries increase traffic and overhead on the network.

If you do not enter a **snmp-server host** command, no notifications are sent. To configure the router to send SNMP notifications, you must enter at least one **snmp-server host** command. If you enter the command without keywords, all trap types are enabled for the host.

To enable multiple hosts, you must issue a separate **snmp-server host** command for each host. You can specify multiple notification types in the command for each host.

When multiple **snmp-server host** commands are given for the same host and type of notification, each succeeding command overwrites the previous command. Only the last **snmp-server host** command will be in effect. For example, if you enter an **snmp-server host inform** command for a host and then enter another **snmp-server host inform** command for the same host, the second command replaces the first.

The **snmp-server host** command is used in conjunction with the **snmp-server enable** command. Use the **snmp-server enable** command to specify which SNMP notifications are sent globally. For a host to receive most notifications, at least one **snmp-server enable** command and the **snmp-server host** command for that host must be enabled.

Some notification types cannot be controlled with the **snmp-server enable** command. For example, some notification types are always enabled and others are enabled by a different command. For example, the linkUpDown notifications are controlled by the **snmp trap link-status** command. These notification types do not require an **snmp-server enable** command.

A *notification-type* option's availability depends on the router type and Cisco IOS software features supported on the router. For example, the envmon notification type is available only if the environmental monitor is part of the system. To see what notification types are available on your system, use the command help (?) at the end of the **snmp-server host** command.

Perform this task to configure the recipient of an SNMP trap operation.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp-server host** *host-id* [**traps | informs**] [**version** {**1 | 2c | 3** [**auth | noauth | priv**]}] *community-string* [**udp-port** *port-number*] [*notification-type*]
4. **exit**
5. **show snmp host**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**snmp-server host** *host-id* [**traps** | **informs**][**version** {**1** | **2c** | **3** [**auth** | **noauth** | **priv**]}] *community-string* [**udp-port** *port-number*] [*notification-type*] **Example:** `Router(config)# snmp-server host 172.16.1.27 version 2c public`	Specifies whether you want the SNMP notifications sent as traps or informs, the version of SNMP to use, the security level of the notifications (for SNMPv3), and the recipient (host) of the notifications.
Step 4	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode.
Step 5	**show snmp host** **Example:** `Router# show snmp host`	(Optional) Displays the SNMP notifications sent as traps, the version of SNMP, and the host IP address of the notifications.

Examples

The following example shows the host information configured for SNMP notifications:

```
Router# show snmp host

Notification host: 10.2.28.1 udp-port: 162   type: inform
user: public    security model: v2c
traps: 00001000.00000000.00000000
```

Configuring SNMP Version 3

When you configure SNMP version 3 and you want to use the SNMPv3 security mechanism for handling SNMP packets, you must establish SNMP groups and users with passwords.

Perform the following tasks to configure SNMP version 3.

- Specifying SNMP-Server Group Names, page 21 (required)
- Configuring SNMP Server Users, page 22 (required)

Specifying SNMP-Server Group Names

SNMPv3 is a security model. A security model is an authentication strategy that is set up for a user and the group in which the user resides.

No default values exist for authentication or privacy algorithms when you configure the **snmp-server group** command. Also, no default passwords exist. For information about specifying a MD5 password, see the documentation for the **snmp-server user** command.

Perform this task to specify a new SNMP group or a table that maps SNMP users to SNMP views.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp-server group** [*groupname* {**v1 | v2c | v3** [**auth | noauth | priv**]}] [**read** *readview*] [**write** *writeview*] [**notify** *notifyview*] [**access** *access-list*]
4. **exit**
5. **show snmp group**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**snmp-server group** [*groupname* {**v1** | **v2c** | **v3** [**auth** | **noauth** | **priv**]}] [**read** *readview*] [**write** *writeview*] [**notify** *notifyview*] [**access** *access-list*] **Example:** `Router(config)# snmp-server group group1 v3 auth access lmnop`	Configures the SNMP server group to enable authentication for members of a specified named access list. • In this example, the SNMP server group *group1* is configured to enable user authentication for members of the named access list *lmnop*.
Step 4	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode.
Step 5	**show snmp group** **Example:** `Router# show snmp group`	Displays information about each SNMP group on the network.

Examples

The following example shows information about each SNMP group on the network:

```
Router# show snmp group

groupname: V1                                 security model:v1
readview : v1default                          writeview: <no writeview specified>
notifyview: <no notifyview specified>
row status: active

groupname: ILMI                               security model:v1
readview : *ilmi                              writeview: *ilmi
notifyview: <no notifyview specified>
row status: active

groupname: ILMI                               security model:v2c
readview : *ilmi                              writeview: *ilmi
notifyview: <no notifyview specified>
row status: active

groupname: group1                             security model:v1
readview : v1default                          writeview: <no writeview specified>
notifyview: <no notifyview specified>
row status: active
```

Configuring SNMP Server Users

To configure a remote user, specify the IP address or port number for the remote SNMP agent of the device where the user resides. Also, before you configure remote users for a particular agent, configure the SNMP engine ID, using the **snmp-server engineID** command with the remote option. The remote agent's SNMP engine ID is required when computing the authentication and privacy digests from the password. If the remote engine ID is not configured first, the configuration command will fail.

For the *privpassword* and *auth-password* arguments, the minimum length is one character; the recommended length is at least eight characters, and should include both letters and numbers.

SNMP passwords are localized using the SNMP engine ID of the authoritative SNMP engine. For informs, the authoritative SNMP agent is the remote agent. You must configure the remote agent's SNMP engine ID in the SNMP database before you can send proxy requests or informs to it.

Perform this task to add a new user to an SNMP group.

Passwords and Digests

No default values exist for authentication or privacy algorithms when you configure the command. Also, no default passwords exist. The minimum length for a password is one character, although we recommend using at least eight characters for security. If you forget a password, you cannot recover it and will need to reconfigure the user. You can specify either a plain text password or a localized MD5 digest.

If you have the localized MD5 or SHA digest, you can specify that string instead of the plain text password. The digest should be formatted as aa:bb:cc:dd where aa, bb, and cc are hexadecimal values. Also, the digest should be exactly 16 octets in length.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp-server engineID** {**local** *engine-id* | **remote** *ip-address* [**udp-port** *udp-port-number*] [**vrf** *vrf-name*] *engine-id-string*}
4. **snmp-server user** *username groupname* [**remote** *ip-address* [**udp-port** *port*]] {**v1** | **v2c** | **v3** [**encrypted**] [**auth** {**md5** | **sha**} *auth-password*]} [**access** *access-list*]
5. **exit**
6. **show snmp user** [*username*]
7. **show snmp engineID**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**snmp-server engineID** {**local** *engine-id* | **remote** *ip-address* [**udp-port** *udp-port-number*] [**vrf** *vrf-name*] *engine-id-string*} **Example:** `Router(config)# snmp-server engineID remote 172.12.15.4 udp-port 120 1a2833c0129a`	Configures the SNMP engine ID. • In this example, the SNMP engine ID is configured for a remote user.
Step 4	**snmp-server user** *username groupname* [**remote** *ip-address* [**udp-port** *port*]] {**v1** | **v2c** | **v3** [**encrypted**] [**auth** {**md5** | **sha**} *auth-password*]} [**access** *access-list*] **Example:** `Router(config)# snmp-server user user1 group1 v3 auth md5 password123`	Configures a new user to an SNMP group with the plain text password “password123” for the user “user1” in the SNMPv3 group “group1”.
Step 5	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode and returns to privileged EXEC mode.

	Command or Action	Purpose
Step 6	**show snmp user** [*username*] **Example:** `Router# show snmp user user123`	Displays the information about the configured characteristics of an SNMP user.
Step 7	**show snmp engineID** **Example:** `Router# show snmp engineID`	(Optional) Displays information about the SNMP engine ID configured for an SNMP user.

Examples

The following example shows the SNMP engine ID configured for the remote user:

```
Router# show snmp engineID

Local SNMP engineID: 1A2836C0129A
Remote Engine ID          IP-addr    Port
1A2833C0129A            remote  10.2.28.1 120
```

The following example shows the information about the configured characteristics of the SNMP user1:

```
Router# show snmp user user1

User name: user1
Engine ID: 000000090200000000C025808
storage-type: nonvolatile       active access-list: 10
Rowstatus: active
Authentication Protocol: MD5
Privacy protocol: DES
Group name: group1
```

Configuring a Router as an SNMP Manager

The SNMP manager feature allows a router to act as a network management station—an SNMP client. As an SNMP manager, the router can send SNMP requests to agents and receive SNMP responses and notifications from agents. When the SNMP manager process is enabled, the router can query other SNMP agents and process incoming SNMP traps.

Perform this task to enable the SNMP manager process and to set the session timeout value.

Security Considerations

Most network security policies assume that routers will accept SNMP requests, send SNMP responses, and send SNMP notifications.

With the SNMP manager functionality enabled, the router may also send SNMP requests, receive SNMP responses, and receive SNMP notifications. Your security policy implementation may need to be updated prior to enabling this feature.

SNMP requests typically are sent to User Datagram Protocol (UDP) port 161. SNMP responses are typically sent from UDP port 161. SNMP notifications are typically sent to UDP port 162.

SNMP Sessions

Sessions are created when the SNMP manager in the router sends SNMP requests, such as informs, to a host or receives SNMP notifications from a host. One session is created for each destination host. If there is no further communication between the router and host within the session timeout period, the session will be deleted.

The router tracks statistics, such as the average round-trip time required to reach the host, for each session. Using the statistics for a session, the SNMP manager in the router can set reasonable timeout periods for future requests, such as informs, for that host. If the session is deleted, all statistics are lost. If another session with the same host is later created, the request timeout value for replies will return to the default value.

Sessions consume memory. A reasonable session timeout value should be large enough that regularly used sessions are not prematurely deleted, yet small enough such that irregularly used or one-time sessions are purged expeditiously.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp-server manager**
4. **snmp-server manager session-timeout** *seconds*
5. **exit**
6. **show snmp**
7. **show snmp sessions** [**brief**]
8. **show snmp pending**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**snmp-server manager** **Example:** `Router(config)# snmp-server manager`	Enables the SNMP manager.
Step 4	**snmp-server manager session-timeout** *seconds* **Example:** `Router(config)# snmp-server manager session-timeout 30`	(Optional) Changes the session timeout value.

Step 5	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode.
Step 6	**show snmp** **Example:** `Router# show snmp`	(Optional) Displays the status of SNMP communications.
Step 7	**show snmp sessions** [**brief**] **Example:** `Router# show snmp sessions`	(Optional) Displays displays the status of SNMP sessions.
Step 8	**show snmp pending** **Example:** `Router# show snmp pending`	(Optional) Displays the current set of pending SNMP requests.

Examples

The following example shows the status of SNMP communications:

```
Router# show snmp

Chassis: 01506199

37 SNMP packets input
    0 Bad SNMP version errors
    4 Unknown community name
    0 Illegal operation for community name supplied
    0 Encoding errors
    24 Number of requested variables
    0 Number of altered variables
    0 Get-request PDUs
    28 Get-next PDUs
    0 Set-request PDUs

78 SNMP packets output
    0 Too big errors (Maximum packet size 1500)
    0 No such name errors
    0 Bad values errors
    0 General errors
    24 Response PDUs
    13 Trap PDUs


SNMP logging: enabled
    Logging to 172.17.58.33.162, 0/10, 13 sent, 0 dropped.

SNMP Manager-role output packets
    4 Get-request PDUs
    4 Get-next PDUs
    6 Get-bulk PDUs
    4 Set-request PDUs
    23 Inform-request PDUs
    30 Timeouts
    0 Drops
```

```
SNMP Manager-role input packets
    0 Inform response PDUs
    2 Trap PDUs
    7 Response PDUs
    1 Responses with errors

SNMP informs: enabled
    Informs in flight 0/25 (current/max)
    Logging to 172.17.217.141.162
        4 sent, 0 in-flight, 1 retries, 0 failed, 0 dropped
    Logging to 172.17.58.33.162
        0 sent, 0 in-flight, 0 retries, 0 failed, 0 dropped
```

The following example displays the status of SNMP sessions:

```
Router# show snmp sessions

Destination: 172.17.58.33.162, V2C community: public
  Round-trip-times: 0/0/0 (min/max/last)
  packets output
    0 Gets, 0 GetNexts, 0 GetBulks, 0 Sets, 4 Informs
    0 Timeouts, 0 Drops
  packets input
    0 Traps, 0 Informs, 0 Responses (0 errors)

Destination: 172.17.217.141.162, V2C community: public, Expires in 575 secs
  Round-trip-times: 1/1/1 (min/max/last)
  packets output
    0 Gets, 0 GetNexts, 0 GetBulks, 0 Sets, 4 Informs
    0 Timeouts, 0 Drops
 packets input
    0 Traps, 0 Informs, 4 Responses (0 errors)
```

The following example shows the current set of pending SNMP requests:

```
Router# show snmp pending

req id: 47, dest: 172.17.58.33.161, V2C community: public, Expires in 5 secs

req id: 49, dest: 172.17.58.33.161, V2C community: public, Expires in 6 secs

req id: 51, dest: 172.17.58.33.161, V2C community: public, Expires in 6 secs

req id: 53, dest: 172.17.58.33.161, V2C community: public, Expires in 8 secs
```

Enabling the SNMP Agent Shutdown Mechanism

Using SNMP packets, a network management tool can send messages to users on virtual terminals and on the console. This facility operates in a similar fashion to the **send** EXEC command; however, the SNMP request that causes the message to be issued to the users also specifies the action to be taken after the message is delivered. One possible action is a shutdown request. After a system is shut down, typically it is reloaded. Because the ability to cause a reload from the network is a powerful feature, it is protected by the **snmp-server system-shutdown** global configuration command. If you do not issue this command, the shutdown mechanism is not enabled.

Perform this task to enable the SNMP agent shutdown mechanism.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp-server system-shutdown**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**snmp-server system-shutdown** **Example:** `Router(config)# snmp-server system-shutdown`	Enables system shutdown using the SNMP message reload feature.

Defining the Maximum SNMP Agent Packet Size

You can define the maximum packet size permitted when the SNMP agent is receiving a request or generating a reply.

Perform this task to set the maximum permitted packet size.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp-server packetsize** *byte-count*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**snmp-server packetsize** *byte-count* **Example:** `Router(config)# snmp-server packetsize 512`	Establishes the maximum packet size.

Limiting the Number of TFTP Servers Used via SNMP

You can limit the number of TFTP servers used for saving and loading configuration files via SNMP by using an access list. Limiting the use of TFTP servers in this way conserves system resources and centralizes the operation for manageability.

Perform this task to limit the number of TFTP servers.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp-server tftp-server-list** *number*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**snmp-server tftp-server-list** *number* **Example:** `Router(config)# snmp-server tftp-server-list 12`	Limits the number of TFTP servers used for configuration file copies via SNMP to the servers in an access list.

Troubleshooting Tips

To monitor SNMP trap activity in real time for the purposes of troubleshooting, use the SNMP **debug** commands, including the **debug snmp packet** EXEC command. For documentation of SNMP **debug** commands, see the *Cisco IOS Debug Command Reference.*

Disabling the SNMP Agent

Perform this task to disable any version of an SNMP agent.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **no snmp-server**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**no snmp-server** **Example:** `Router(config)# no snmp-server`	Disables SNMP agent operation.

Configuring SNMP Notifications

To configure a router to send SNMP traps or informs, perform the tasks described in the following sections:

- Configuring the Router to Send SNMP Notifications, page 31 (required)
- Changing Notification Operation Values, page 33 (optional)
- Controlling Individual RFC 1157 SNMP Traps, page 34 (optional)
- Configuring SNMP Notification Log Options, page 36 (optional)

Note Many snmp-server commands use the word **traps** in their command syntax. Unless there is an option within the command to specify either traps or informs, the keyword **traps** should be taken to mean traps, informs, or both. Use the **snmp-server host** command to specify whether you want SNMP notifications to be sent as traps or informs.

To use informs, the SNMP manager (also known as the SNMP proxy manager) must be available and enabled on a device. Earlier, the SNMP manager was available only with Cisco IOS PLUS images. However, the SNMP manager is now available with all Cisco IOS releases that support SNMP.

Use Cisco Feature Navigator for information about SNMP manager support for Cisco IOS releases. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn.

Configuring the Router to Send SNMP Notifications

Perform this task to configure the router to send traps or informs to a host.

SUMMARY STEPS

1. **enable**
2. **configure terminal**

3. **snmp-server engineID remote** *remote-ip-address remote-engineID*
4. **snmp-server user** *username groupname* [**remote** *host* [**udp-port** *port*] {**v1 | v2c | v3** [**encrypted**] [**auth** {**md5 | sha**} *auth-password*]} [**access** *access-list*]
5. **snmp-server group** *groupname* {**v1 | v2c | v3** {**auth | noauth | priv**}} [**read** *readview*] [**write** *writeview*] [**notify** *notifyview*] [**access** *access-list*]
6. **snmp-server host** *host* [**traps | informs**] [**version** {**1 | 2c | 3** [**auth | noauth | priv**]}] *community-string* [*notification-type*]
7. **snmp-server enable traps** [*notification-type* [*notification-options*]]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**snmp-server engineID remote** *remote-ip-address remote-engineID* **Example:** `Router(config)# snmp-server engineID remote 172.16.20.3 80000009030000B064EFE100`	Specifies the SNMP engine ID and configures the VRF name traps-vrf for SNMP communications with the remote device at 172.16.20.3.
Step 4	**snmp-server user** *username groupname* [**remote** *host* [**udp-port** *port*] {**v1** | **v2c** | **v3** [**encrypted**] [**auth** {**md5** | **sha**} *auth-password*]} [**access** *access-list*] **Example:** `Router(config)# snmp-server user abcd public remote 172.16.20.3 v3 encrypted auth md5 publichost remotehostusers`	Configures an SNMP user to be associated with the host created in Step 3. **Note** You cannot configure a remote user for an address without first configuring the engine ID for that remote host. This restriction is imposed in the design of these commands; if you try to configure the user before the host, you will receive a warning message and the command will not be executed.
Step 5	**snmp-server group** *groupname* {**v1** | **v2c** | **v3** {**auth** | **noauth** | **priv**}} [**read** *readview*] [**write** *writeview*] [**notify** *notifyview*] [**access** *access-list*] **Example:** `Router(config)# snmp-server group GROUP1 v2c auth read viewA write viewA notify viewB`	Configures an SNMP group.

	Command or Action	Purpose
Step 6	**snmp-server host** *host* [**traps** | **informs**] [**version** {**1** | **2c** | **3** [**auth** | **noauth** | **priv**]}] *community-string* [*notification-type*] **Example:** `Router(config)# snmp-server host example.com informs version 3 public`	Specifies whether you want the SNMP notifications sent as traps or informs, the version of SNMP to use, the security level of the notifications (for SNMPv3), and the recipient (host) of the notifications. • The **snmp-server host** command specifies which hosts will receive SNMP notifications, and whether you want the notifications sent as traps or informs.
Step 7	**snmp-server enable traps** [*notification-type* [*notification-options*]] **Example:** `Router(config)# snmp-server enable traps bgp`	Enables sending of traps or informs and specifies the type of notifications to be sent. • If a *notification-type* is not specified, all supported notification will be enabled on the router. • To discover which notifications are available on your router, enter the **snmp-server enable traps ?** command. • The **snmp-server enable traps** command globally enables the production mechanism for the specified notification types (such as Border Gateway Protocol [BGP] traps, config traps, entity traps, Hot Standby Router Protocol [HSRP] traps, and so on).

Changing Notification Operation Values

You can specify a value other than the default for the source interface, message (packet) queue length for each host, or retransmission interval.

Perform this task to change notification operation values as needed.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp-server trap-source** *interface*
4. **snmp-server queue-length** *length*
5. **snmp-server trap-timeout** *seconds*
6. **snmp-server informs** [**retries** *retries*] [**timeout** *seconds*] [**pending** *pending*]

DETAILED STEPS

Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.

Step 3	**snmp-server trap-source** *interface* **Example:** `Router(config)# snmp-server trap-source ethernet 2/1`	Sets the IP address for the Ethernet interface in slot2, port 1 as the source for all SNMP notifications.
Step 4	**snmp-server queue-length** *length* **Example:** `Router(config)# snmp-server queue-length 50`	Establishes the message queue length for each notification. • This example shows the queue length set to 50 entries.
Step 5	**snmp-server trap-timeout** *seconds* **Example:** `Router(config)# snmp-server trap-timeout 30`	Defines how often to resend notifications on the retransmission queue.
Step 6	**snmp-server informs** [**retries** *retries*] [**timeout** *seconds*] [**pending** *pending*] **Example:** `Router(config)# snmp-server informs retries 10 timeout 30 pending 100`	Configures inform-specific operation values. • This example sets the maximum number of times to resend an inform, the number of seconds to wait for an acknowledgment before resending, and the maximum number of informs waiting for acknowledgments at any one time.

Controlling Individual RFC 1157 SNMP Traps

Starting with Cisco IOS Release 12.1(3)T, you can globally enable or disable authenticationFailure, linkUp, linkDown, warmStart, and coldStart traps or informs individually. (These traps constitute the "generic traps" defined in RFC 1157.) Note that linkUp and linkDown notifications are enabled by default on specific interfaces but will not be sent unless they are enabled globally.

Perform this task to enable the authenticationFailure, linkUp, linkDown, warmStart, and coldStart notification types.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp-server enable traps snmp** [**authentication**] [**linkup**] [**linkdown**] [**warmstart**] [**coldstart**]
4. **interface** *type slot/port*
5. **no snmp-server link status**
6. **exit**
7. **exit**
8. **show snmp mib ifmib traps**

DETAILED STEPS

Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**snmp-server enable traps snmp** [**authentication**] [**linkup**] [**linkdown**] [**warmstart**] [**coldstart**] **Example:** `Router(config)# snmp-server enable traps snmp`	Enables RFC 1157 generic traps. • When used without any of the optional keywords, enables authenticationFailure, linkUp, linkDown, warmStart, and coldStart traps. • When used with keywords, enables only the trap types specified. For example, to globally enable only linkUp and linkDown SNMP traps or informs for all interfaces, use the **snmp-server enable traps snmp linkup linkdown** form of this command.
Step 4	**interface** *type slot/port* **Example:** `Router(config)# interface ethernet 0/0`	Enters interface configuration mode for a specific interface. **Note** To enable SNMP traps for individual interfaces such as Dialer, use the **snmp trap link-status permit duplicates** command in interface configuration mode. For example, to enter dialer interface configuration mode, enter the interface type as dialer.
Step 5	**no snmp-server link status** **Example:** `Router(config-if)# no snmp-server link status`	Disables the sending of linkUp and linkDown notifications for all generic interfaces. **Note** To disable SNMP traps for individual interfaces such as Dialer, use the **no snmp trap link-status permit duplicates** command in interface configuration mode.
Step 6	**exit** **Example:** `Router(config-if)# exit`	Exits interface configuration mode.
Step 7	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode and returns to privileged EXEC mode.
Step 8	**show snmp mib ifmib traps** **Example:** `Router# show snmp mib ifmib traps`	(Optional) Displays the status of linkup and linkdown traps for each of interfaces configured for the system.

Examples

The following example shows the status of linkup and linkdown traps for all interfaces configured for the system:

```
Router# show snmp mib ifmib traps

ifDescr                   ifindex    TrapStatus
-------------------------------------------------

FastEthernet3/6                   14     enabled
FastEthernet3/19                  27     enabled
GigabitEthernet5/1                57     enabled
unrouted VLAN 1005                73     disabled
FastEthernet3/4                   12     enabled
FastEthernet3/39                  47     enabled
FastEthernet3/28                  36     enabled
FastEthernet3/48                  56     enabled
unrouted VLAN 1003                74     disabled
FastEthernet3/2                   10     enabled
Tunnel0                           66     enabled
SPAN RP Interface                 64     disabled
Tunnel10                          67     enabled
FastEthernet3/44                  52     enabled
GigabitEthernet1/3                3      enabled
FastEthernet3/11                  19     enabled
FastEthernet3/46                  54     enabled
GigabitEthernet1/1                1      enabled
FastEthernet3/13                  21     enabled
unrouted VLAN 1                   70     disabled
GigabitEthernet1/4                4      enabled
FastEthernet3/9                   17     enabled
FastEthernet3/16                  24     enabled
FastEthernet3/43                  51     enabled
```

Configuring SNMP Notification Log Options

Perform this task to configure SNMP notification log options. These options allow you to control the log size and timing values. The SNMP log can become very large and long if left unmodified.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp mib notification-log default**
4. **snmp mib notification-log globalageout** *seconds*
5. **snmp mib notification-log globalsize** *size*
6. **exit**
7. **show snmp mib notification-log**

DETAILED STEPS

Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**snmp mib notification-log default** **Example:** `Router(config)# snmp mib notification-log default`	Creates an unnamed SNMP notification log.
Step 4	**snmp mib notification-log globalageout** *seconds* **Example:** `Router(config)# snmp mib notification-log globalageout 20`	Sets the maximum amount of time SNMP notification log entries remain in the system memory. • In this example, the system is configured to delete entries in the SNMP notification log that were logged more than 20 minutes ago.
Step 5	**snmp mib notification-log globalsize** *size* **Example:** `Router(config)# snmp mib notification-log globalsize 600`	Sets the maximum number of entries that can be stored in all SNMP notification logs.
Step 6	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode.
Step 7	**show snmp mib notification-log** **Example:** `Router# show snmp mib notification-log`	Displays information about the state of the local SNMP notification logging.

Examples

This example shows information about the state of local SNMP notification logging:

```
Router# show snmp mib notification-log

GlobalAgeout 20, GlobalEntryLimit 600
Total Notifications logged in all logs 0
Log Name"", Log entry Limit 600, Notifications logged 0
Logging status enabled
Created by cli
```

Configuring Interface Index Display and Interface Indexes and Long Name Support

The display of Interface Indexes lets advanced users of SNMP view information about the interface registrations directly on a managed agent. An external NMS is not required.

Configuration of Long Alias Names for the interfaces lets users configure the ifAlias (the object defined in the MIB whose length is restricted to 64) up to 255 bytes.

Prerequisites

SNMP must be enabled on your system.

Restrictions

The Interface Index Display and Interface Alias Long Name Support feature is not supported on all Cisco platforms. Use Cisco Feature Navigator to find information about platform support and Cisco IOS software image support.

Perform this task to configure the IF-MIB to retain ifAlias values of longer than 64 characters and to configure the ifAlias values for an interface.

Note To verify if the ifAlias description is longer than 64 characters, perform an SNMP MIB walk for the ifMIB ifAlias variable from an NMS and verify that the entire description is displayed in the values for ifXEntry.18.

The description for interfaces also appears in the output from the **more system:running config** privileged EXEC mode command.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp ifmib ifalias long**
4. **interface** *type number*
5. **description** *text-string*
6. **exit**
7. **show snmp mib**
8. **show snmp mib ifmib ifindex** [*type number*] [**detail**] [**free-list**]

DETAILED STEPS

Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**snmp ifmib ifalias long** **Example:** `Router(config)# snmp ifmib ifalias long`	Configures the Interfaces MIB (IF-MIB) on the system to return ifAlias values of longer than 64 characters to a Network Management System. If the ifAlias values are not configured using the **snmp ifmib ifalias long** command, ifAlias description will be restricted to 64 characters.
Step 4	**interface** *type number* **Example:** `Router(config)# interface ethernet 2/4`	Enters interface configuration mode. • The form of this command varies depending on the interface being configured.
Step 5	**description** *text-string* **Example:** `Router(config)# description This text string description can be up to 256 characters long`	Configures a free-text description of the specified interface. • This description can be up to 240 characters in length and is stored as the ifAlias object value in the IF-MIB. If the ifAlias values are not configured using **snmp ifmib ifalias long** command, ifAlias description for SNMP set and get operations is restricted to 64 characters, although the interface description is configured for more than 64 characters by using the **description** command.
Step 6	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode.
Step 7	**show snmp mib** **Example:** `Router# show snmp mib`	Displays a list of the MIB module instance identifiers registered on your system. • The resulting display could be lengthy.
Step 8	**show snmp mib ifmib ifindex** [*type number*] [**detail**] [**free-list**] **Example:** `Router# show snmp mib ifmib ifindex Ethernet 2/0`	Displays the Interfaces MIB ifIndex values registered on your system for all interfaces or the specified interface.

Examples

The following example lists the MIB module instance identifiers registered on your system. The resulting display could be lengthy. Only a small portion is shown here.

```
Router# show snmp mib

system.1
system.2
sysUpTime
system.4
system.5
system.6
system.7
system.8
sysOREntry.2
sysOREntry.3
sysOREntry.4
interfaces.1
ifEntry.1
ifEntry.2
ifEntry.3
ifEntry.4
ifEntry.5
ifEntry.6
ifEntry.7
ifEntry.8
ifEntry.9
ifEntry.10
ifEntry.11

 --More--

captureBufferEntry.2
captureBufferEntry.3
captureBufferEntry.4
captureBufferEntry.5
captureBufferEntry.6
captureBufferEntry.7
capture.3.1.1
eventEntry.1
eventEntry.2
eventEntry.3
eventEntry.4
eventEntry.5
eventEntry.6

eventEntry.7
logEntry.1
logEntry.2
logEntry.3
logEntry.4
rmon.10.1.1.2
rmon.10.1.1.3
rmon.10.1.1.4
rmon.10.1.1.5
rmon.10.1.1.6
rmon.10.1.1.7
rmon.10.2.1.2
rmon.10.2.1.3
rmon.10.3.1.2
```

The following example shows output for the Interfaces MIB ifIndex values registered on a system for a specific interface:

```
Router# show snmp mib ifmib ifindex Ethernet 2/0

Ethernet2/0: Ifindex = 2
```

The following example shows output for the Interfaces MIB ifIndex values registered on a system for all interfaces:

```
Router# show snmp mib ifmib ifindex

ATM1/0: Ifindex = 1
ATM1/0-aal5 layer: Ifindex = 12
ATM1/0-atm layer: Ifindex = 10
ATM1/0.0-aal5 layer: Ifindex = 13
ATM1/0.0-atm subif: Ifindex = 11
ATM1/0.9-aal5 layer: Ifindex = 32
ATM1/0.9-atm subif: Ifindex = 31
ATM1/0.99-aal5 layer: Ifindex = 36
ATM1/0.99-atm subif: Ifindex = 35
Ethernet2/0: Ifindex = 2
Ethernet2/1: Ifindex = 3
Ethernet2/2: Ifindex = 4
Ethernet2/3: Ifindex = 5
Null0: Ifindex = 14
Serial3/0: Ifindex = 6
Serial3/1: Ifindex = 7
Serial3/2: Ifindex = 8
Serial3/3: Ifindex = 9
```

Troubleshooting Tips

An alternative to using the ifAlias value for the identification of interfaces across reboots is to use the cciDescr object in the Cisco Circuit Interface MIB (CISCO-CIRCUIT-INTERFACE-MIB.my). This MIB object can be used only for circuit-based interfaces such as ATM or Frame Relay interfaces. Cisco IOS Release 12.2(2)T introduced the Circuit Interface Identification Persistence for SNMP feature, which maintains the user-defined name of the circuit (defined in the cciDescr object) across reboots, allowing for the consistent identification of circuit-based interfaces.

Configuring SNMP Support for VPNs

This section describes how to configure SNMP support for VPNs. The SNMP Support for VPNs feature provides configuration commands that allow users to associate SNMP agents and managers with specific VRFs. The specified VRF is used to send SNMP traps and informs and responses between agents and managers. If a VRF is not specified, the default routing table for the VPN is used.

Support for VPNs allows users to configure an SNMP agent to only accept SNMP requests from a certain set of VPNs. With this configuration, providers can provide network management services to their customers who then can manage all user VPN devices.

Restrictions

- This feature is not supported on all Cisco platforms. Use Cisco Feature Navigator to find information about platform support and Cisco IOS software image support.
- Not all MIBs are VPN aware. To list the VPN-aware MIBs, use the **show snmp mib context** command. For more information about VPN-aware MIBs, see the SNMP Support over VPNs—Context-based Access Control configuration module.

Perform this task to configure SNMP support for a specific VPN.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp-server host** *host-address* [**vrf** *vrf-name*] [**traps | informs**] [**version** {**1 | 2c | 3** [**auth | noauth | priv**]}] *community-string* [**udp-port** *port*] [*notification-type*]
4. **snmp-server engineID remote** *ip-address* [**udp-port** *udp-port-number*] [**vrf** *vrf-name*] *engineid-string*
5. **exit**
6. **show snmp host**

DETAILED STEPS

Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**snmp-server host** *host-address* [**vrf** *vrf-name*] [**traps** | **informs**] [**version** {**1** | **2c** | **3** [**auth** | **noauth** |**priv**]}] *community-string* [**udp-port** *port*] [*notification-type*] **Example:** `Router(config)# snmp-server host example.com public vrf trap-vrf`	Specifies the recipient of an SNMP notification operation and specifies the VRF table to be used for the sending of SNMP notifications.
Step 4	**snmp-server engineID remote** *ip-address* [**udp-port** *udp-port-number*] [**vrf** *vrf-name*] *engineid-string* **Example:** `Router(config)# snmp-server engineID remote 172.16.20.3 vrf traps-vrf 800000090300000B064EFE100`	Configures a name for the remote SNMP engine on a router when configuring SNMP over a specific VPN for a remote SNMP user.
Step 5	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode.
Step 6	**show snmp host** **Example:** `Router# show snmp host`	(Optional) Displays the SNMP configuration and verifies that the SNMP Support for VPNs feature is configured properly.

Configuring Interface IfIndex Persistence

The following sections contain the tasks to configure Interface Index Persistence:

Enabling and Disabling IfIndex Persistence Globally

Perform this task to enable IfIndex persistence globally.

Prerequisites

The configuration tasks described in this section assume that you have configured SNMP on your routing device and are using SNMP to monitor network activity using the Cisco IOS command line interface and/or a network management system (NMS) application.

Restrictions

The interface-specific ifIndex persistence command (**snmp ifindex persistence**) cannot be used on subinterfaces. A command applied to an interface is automatically applied to all the subinterfaces associated with that interface.

Testing indicates that approximately 25 bytes of NVRAM storage are used by this feature per interface. There may be some boot delay exhibited on platforms with lower CPU speeds.

Note After ifIndex persistence commands have been entered, the configuration must be saved using the **copy running-config startup-config** EXEC mode command to ensure consistent ifIndex values.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp-server ifindex persist**
4. **no snmp-server ifindex persist**
5. **exit**

DETAILED STEPS

Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**snmp-server ifindex persist** **Example:** `Router(config)# snmp-server ifindex persist`	Globally enables ifIndex values that will remain constant across reboots.
Step 4	**no snmp-server ifindex persist** **Example:** `Router(config)# no snmp-server ifindex persist`	Disables global ifIndex persistence.
Step 5	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode.

Enabling and Disabling IfIndex Persistence on Specific Interfaces

Perform this task to configure ifIndex persistence only on a specific interface.

Tips Use the **snmp ifindex clear** command on a specific interface when you want that interface to use the global configuration setting for ifIndex persistence. This command clears any ifIndex configuration commands previously entered for that specific interface.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type slot/port*
4. **snmp ifindex persist**
5. **no snmp ifindex persist**
6. **end**

DETAILED STEPS

Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type slot/port* **Example:** `Router(config)# interface ethernet 0/1`	Enters interface configuration mode for the specified interface. **Note** The syntax of the **interface** command will vary depending on the platform you are using.
Step 4	**snmp ifindex persist** **Example:** `Router(config-if)# snmp ifindex persist`	Enables an ifIndex value that is constant across reboots on the specified interface.
Step 5	**no snmp ifindex persist** **Example:** `Router(config-if)# no snmp ifindex persist`	Disables an ifIndex value that is constant across reboots on the specified interface.
Step 6	**end** **Example:** `Router(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Configuring MIB Persistence

Note Beginning with Cisco IOS Release 12.4(20)T, MIB persistence is automatic; manual configuration is not required.

The MIB Persistence features allow the SNMP data of a MIB to be persistent across reloads; that is, MIB information retains the same set of object values each time a networking device reboots. The following sections contain tasks for using Distributed Management Event and Expression MIB persistence.

- Enabling and Disabling Event MIB Persistence, page 46 (optional)
- Enabling and Disabling Expression MIB Persistence, page 47 (optional)

Prerequisites

- SNMP is configured on your networking device
- Values for Event MIB and Expression MIB have been configured

Restrictions

- If the number of MIB objects to persist increases, NVRAM storage capacity may be strained. Occasionally, the time taken to write MIB data to NVRAM may be longer than expected.
- The Distributed Management Event MIB Persistence feature is not supported on all Cisco platforms. Use Cisco Feature Navigator to find information about platform support and Cisco IOS software image support.

Enabling and Disabling Event MIB Persistence

Perform this task to configure Event MIB Persistence.

Note Event MIB Persistence is disabled by default.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp mib persist event**
4. **no snmp mib persist event**
5. **exit**
6. **write mib-data**
7. **copy running-config startup-config**

DETAILED STEPS

Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**snmp mib persist event** **Example:** `Router(config)# snmp mib persist event`	Enables MIB Persistence for Event MIB.
Step 4	**no snmp mib persist event** **Example:** `Router(config)# no snmp mib persist event`	(Optional) Disables MIB Persistence for Event MIB.

Step 5	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode.
Step 6	**write mib-data** **Example:** `Router# write mib-data`	Saves Event MIB Persistence configuration data to NVRAM.
Step 7	**copy running-config startup-config** **Example:** `Router# copy running-config startup-config`	Copies the running configuration to the startup configuration.

Enabling and Disabling Expression MIB Persistence

Perform this task to configure Expression MIB Persistence.

Note Expression MIB Persistence is disabled by default.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp mib persist expression**
4. **no snmp mib persist expression**
5. **exit**
6. **write mib-data**
7. **copy running-config startup-config**
8. **more system:running-config**

DETAILED STEPS

Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**snmp mib persist expression** **Example:** `Router(config)# snmp mib persist expression`	Enables MIB Persistence for Expression MIB.

Step 4	**no snmp mib persist expression** **Example:** `Router(config)# no snmp mib persist expression`	(Optional) Disables MIB Persistence for Expression MIB.
Step 5	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode.
Step 6	**write mib-data** **Example:** `Router# write mib-data`	Saves Expression MIB Persistence configuration data to NVRAM.
Step 7	**copy running-config startup-config** **Example:** `Router# copy running-config startup-config`	Copies the running configuration to the startup configuration.
Step 8	**more system:running-config** **Example:** `Router# more system:running-config`	Displays the currently running configuration. • Use this command to verify MIB persistence configuration.

Configuring Event MIB Using SNMP

Event MIB can be configured using SNMP directly. In this procedure, the Event MIB is configured to monitor the delta values of ifInOctets for all interfaces once per minute. If any of the samples exceed the specified threshold, a trap notification will be sent.

There are no Cisco IOS software configuration tasks associated with the Event MIB. All configuration of Event MIB functionality must be performed though applications using SNMP. This section provides a sample configuration session using a network management application on an external device. See the "Related Documents" section for information about configuring SNMP on your Cisco routing device.

All configuration of Event MIB functionality must be performed though applications using SNMP. The following section provides a step-by-step Event MIB configuration using SNMP research tools available for Sun workstations. The **setany** commands given below are executed using the SNMP application. Note that these commands are not Cisco IOS CLI commands. It is assumed that SNMP has been configured on your routing device.

In this configuration, the objective is to monitor ifInOctets for all interfaces. The Event MIB is configured to monitor the delta values of ifInOctets for all interfaces once per minute. If any of the samples exceed the specified threshold of 30, a Trap notification will be sent.

There are four parts to the following example:

- Setting the Trigger in the Trigger Table
- Creating an Event in the Event Table
- Setting the Trigger Threshold in the Trigger Table
- Activating the Trigger

Setting the Trigger in the Trigger Table

Perform this task to set the trigger in the trigger table:

	Command	Purpose
Step 1	`setany -v2c $ADDRESS private mteTriggerEntryStatus.4.106.111.104.110.1 -i 5`	Creates a trigger row in the table with john as the mteOwner and 1 as the trigger name. The index is given in decimal representation of the ASCII value of john.1.
Step 2	`setany -v2c $ADDRESS private mteTriggerValueID.4.106.111.104.110.1 -d 1.3.6.1.2.1.2.2.1.10`	Sets the mteTriggerValueID to the OID to be watched. In this example, the OID to be monitored is ifInOctets.
Step 3	`setany -v2c $ADDRESS private mteTriggerValueIDWildcard.4.106.111.104.110.1 -i 1`	Sets the mteTriggerValueIDWildcard to TRUE to denote a object referenced through wildcarding.
Step 4	`setany -v2c $ADDRESS private mteTriggerTest.4.106.111.104.110.1 -o '20'`	Sets the mteTriggerTest to Threshold.
Step 5	`setany -v2c $ADDRESS private mteTriggerFrequency.4.106.111.104.110.1 -g 60`	Sets the mteTriggerFrequency to 60. This means that ifInOctets are monitored once every sixty seconds.
Step 6	`setany -v2c $ADDRESS private mteTriggerSampleType.4.106.111.104.110.1 -i 2`	Sets the sample type to Delta.
Step 7	`setany -v2c $ADDRESS private mteTriggerEnabled.4.106.111.104.110.1 -i 1`	Enables the trigger.

Creating an Event in the Event Table

Perform this task to create an event in the event table:

	Command	Purpose
Step 1	`setany -v2c $ADDRESS private mteEventEntryStatus.4.106.111.104.110.101.118.101.110. 116 -i 5`	Create a row in the Event Table. The mteOwner here is again john and mteEventName is event. The default action is to send out a notification.
Step 2	`setany -v2c $ADDRESS private mteEventEnabled.4.106.111.104.110.101.118.101.110.116 -i 1`	Enables the Event.
Step 3	`setany -v2c $ADDRESS private mteEventEntryStatus.4.106.111.104.110.101.118.101.110. 116 -i 1`	Makes the EventRow active.

Setting the Trigger Threshold in the Trigger Table

Perform this task to set the trigger threshold in the trigger table:

	Command	Purpose
Step 1	`setany -v2c $ADDRESS private mteTriggerThresholdRising.4.106.111.104.110.1 -i 30`	Sets the Rising Threshold value to 30. Note that a row would already exist for john.1 in the Trigger Threshold Table.

Step 2	`setany -v2c $ADDRESS private mteTriggerThresholdRisingEventOwner.4.106.111.104.110.1 -D "john"`
	
	`setany -v2c $ADDRESS private mteTriggerThresholdRisingEvent.4.106.111.104.110.1 -D "event"`
		Points to the entry in the Event Table that specifies the action that is to be performed.

Activating the Trigger

Perform this task to activate the trigger:

	Command	Purpose
Step 1	`setany -v2c $ADDRESS private mteTriggerEntryStatus.4.106.111.104.110.1 -i 1`	Makes the trigger active.

To confirm the above configuration is working, ensure that at least one of the interfaces gets more than 30 packets in a minute. This should cause a trap to be sent out after one minute.

Monitoring and Maintaining Event MIB

Use the following commands to monitor Event MIB activity from the Cisco IOS command-line interface:

Command	Purpose
`debug management event mib`	Prints messages to the screen whenever the Event MIB evaluates a specified trigger. These messages are given in real-time, and are intended to be used by technical support engineers for troubleshooting purposes.
`show management event`	Displays the SNMP Event values that have been configured on your routing device through the use of the Event MIB.

Configuring Event MIB Using CLI

Event MIB can be configured using SNMP directly. In this procedure, the Event MIB is configured to monitor the delta values of ifInOctets for all interfaces once per minute. If any of the samples exceed the specified threshold, a trap notification will be sent.

However, in the Cisco IOS Release 12.4(20)T, the Event MIB feature is enhanced to add CLIs to configure events, event action, and trigger.

This section contains the following tasks to configure Event MIB:

Configuring Scalar Variables

Perform this task to configure scalar variables for Event MIB.

Prerequisites

To configure the scalar variables for Event MIB, you should be familiar with the Event MIB scalar variables.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp mib event sample minimum** *value*
4. **snmp mib event sample instance maximum** *value*
5. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**snmp mib event sample minimum** *value* **Example:** `Router(config)# snmp mib event sample minimum 10`	Sets the minimum value for object sampling.
Step 4	**snmp mib event sample instance maximum** *value* **Example:** `Router(config)# snmp mib event sample instance maximum 50`	Sets the maximum value for object instance sampling.
Step 5	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode.

Configuring Event MIB Object List

To configure Event MIB, you need to set up a list of objects that can be added to notifications according to trigger, trigger test, or the event.

Prerequisites

To configure the Event MIB object list, you should be familiar with the Event MIB objects and object identifiers, which can be added to notifications according to event, trigger, or the trigger test.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp mib event object list owner** *object-list-owner* **name** *object-list-name object-number*
4. **object id** *object-identifier*
5. **wildcard**
6. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**snmp mib event object list owner** *object-list-owner* **name** *object-list-name object-number* **Example:** `Router(config)# snmp mib event object list owner owner1 name objectA number 10`	Configures the Event MIB object list.
Step 4	**object id** *object-identifier* **Example:** `Router(config-event-objlist)# object id ifInOctets`	Specifies the object identifier for the object configured for the event.
Step 5	**wildcard** **Example:** `Router(config-event-objlist)# wildcard`	(Optional) Starts a wildcarded search for object identifiers. By specifying a partial object identifier, you can obtain a list of object identifiers.
Step 6	**exit** **Example:** `Router(config-event-objlist)# exit`	Exits object list configuration mode.

Configuring Event

Perform this task to configure a management event.

Prerequisites

To configure a management event, you should be familiar with the SNMP MIB events and object identifiers.

SUMMARY STEPS

1. **enable**
2. **config terminal**
3. **snmp mib event owner** *event-owner* **name** *event-name*
4. **description** *event-description*

5. **enable**
6. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**snmp mib event owner** *event-owner* **name** *event-name* **Example:** `Router(config)# snmp mib event owner owner1 event EventA`	Enters the event configuration mode.
Step 4	**description** *event-description* **Example:** `Router(config-event)# description "EventA is an RMON event"`	Describes the function and use of the event.
Step 5	**enable** **Example:** `Router(config-event)# enable`	Enables the event. **Note** The event can be executed during an event trigger only if it is enabled.
Step 6	**exit** **Example:** `Router(config-event)# exit`	Exits event configuration mode.

Configuring Event Action

By configuring an event action, you can define the actions that an application can perform during an event trigger. The actions for an event include sending a notification, setting a MIB object and so on. You can set the event action information to either **set** or **notification**. The actions for the event can be configured only in the event configuration mode.

The following sections contain the tasks to configure event action:

- Configuring Action Notification, page 54
- Configuring Action Set, page 55

Configuring Action Notification

Perform this task to set the notification action for the event.

SUMMARY STEPS

1. **action notification**
2. **object id** *object-id*
3. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**action notification** **Example:** `Router(config-event)# action notification`	Sets the notification action for an event. **Note** If the event action is set to notification, a notification is generated whenever an object associated with an event is modified.
Step 2	**object id** *object-id* **Example:** `Router(config-event-action-notification)# object id ifInOctets`	Configures object for action notification. When the object specified is modified, a notification will be sent to the host system.
Step 3	**exit** **Example:** `Router(config-event-action-notification)# exit`	Exits action notification configuration mode.

Configuring Action Set

Perform this task to set actions for an event.

SUMMARY STEPS

1. **action set**
2. **object id** *object-id*
3. **value** *integer-value*
4. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**action set** **Example:** `Router(config-event)# action set`	Enters action set configuration mode.
Step 2	**object id** *object-id* **Example:** `Router(config-event-action-set)# object id ifInOctets`	Configures object for action set. When the object specified is modified, a specified action will be performed.

	Command or Action	Purpose
Step 3	**value** *integer-value* **Example:** `Router(config-event-action-set)# value 10`	Sets a value for the object.
Step 4	**exit** **Example:** `Router(config-event-action-set)# exit`	Exits action set configuration mode.

Configuring Event Trigger

By configuring an event trigger, you can list the objects to monitor, and associate each trigger to an event.

Perform this task to configure an event trigger.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp mib event trigger owner** *trigger-owner* **name** *trigger-name*
4. **description** *trigger-description*
5. **frequency** *seconds*
6. **object list owner** *object-list-owner* **name** *object-list-name*
7. **object id** *object-identifier*
8. **enable**
9. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**snmp mib event trigger owner** *trigger-owner* **name** *trigger-name* **Example:** `Router(config)# snmp mib event trigger owner owner1 name EventTriggerA`	Enables event trigger configuration mode for the specified event trigger.

	Command or Action	Purpose
Step 4	**description** *trigger-description* **Example:** `Router(config-event-trigger)# description EventTriggerA is an RMON alarm.`	Describes the function and use of the event trigger.
Step 5	**frequency** *seconds* **Example:** `Router(config-event-trigger)# frequency 120`	Configures the waiting time (number of seconds) between trigger samples.
Step 6	**object list owner** *object-list-owner* **name** *object-list-name* **Example:** `Router(config-event-trigger)# object list owner owner1 name ObjectListA`	Specifies the list of objects that can be added to notifications.
Step 7	**object id** *object-identifier* **Example:** `Router(config-event-trigger)# object id ifInOctets`	Configures object identifiers for an event trigger.
Step 8	**enable** **Example:** `Router(config-event-trigger)# enable`	Enables the event trigger.
Step 9	**exit** **Example:** `Router(config-event-trigger)# exit`	Exits event trigger configuration mode.

Configuring Existence Trigger Test

Perform this task to configure trigger parameters for the test existence trigger type.

You should configure this trigger type in the event trigger configuration mode.

SUMMARY STEPS

1. **test existence**
2. **object list owner** *object-list-owner* **name** *object-list-name*
3. **event owner** *event-owner* **name** *event-name*
4. **type** {**present** | **absent** | **changed**}
5. **startup** {**present** | **absent**|}
6. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**test existence** **Example:** `Router(config-event-trigger)# test existence`	Enables test existence configuration mode.
Step 2	**event owner** *event-owner* **name** *event-name* **Example:** `Router(config-event-trigger-existence)# event owner owner1 name EventA`	Configures event for existence trigger test.
Step 3	**object list owner** *object-list-owner* **name** *object-list-name* **Example:** `Router(config-event-trigger-existence)# object list owner owner1 name ObjectListA`	Configures the list of objects for Existence trigger test.
Step 4	**type** {**present** | **absent** | **changed**} **Example:** `Router(config-event-trigger-existence)# type present`	Performs the specified type of existence test. This example uses the present test type. There are three types of existence tests; present, absent and changed. • Present—Setting type to present tests if the objects that appear during the event trigger exist. • Absent—Setting type to absent tests if the objects that disappear during the event trigger exist. • Changed—Setting type to changed tests if the objects that changed during the event trigger exist.
Step 5	**startup** {**present** | **absent**} **Example:** `Router(config-event-trigger-existence)# startup present`	Triggers an event if the test is performed successfully.
Step 6	**exit** **Example:** `Router(config-event-trigger-existence)# exit`	Exits existence trigger test configuration mode.

Configuring Boolean Trigger Test

Perform this task to configure trigger parameters for Boolean trigger type. You should configure this trigger test in the event trigger configuration mode.

SUMMARY STEPS

1. **test boolean**
2. **comparison** {**unequal** | **equal** | **less** | **lessOrEqual** | **greater** | **greaterOrEqual**}

3. **object list owner** *object-list-owner* **name** *object-list-name*
4. **event owner** *event-owner* **name** *event-name*
5. **value** *integer-value*
6. **startup**
7. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**test boolean** **Example:** `Router(config-event-trigger)# test boolean`	Enables Boolean trigger test configuration mode.
Step 2	**comparison** {**unequal** | **equal** | **less** | **lessOrEqual** | **greater** | **greaterOrEqual**} **Example:** `Router(config-event-trigger-boolean)# comparison unequal`	Performs the specified Boolean comparison test. The value for the Boolean comparison test can be set to unequal, equal, less, lessOrEqual, greater, or greaterOrEqual.
Step 3	**value** *integer-value* **Example:** `Router(config-event-trigger-boolean)# value 10`	Sets a value for the Boolean trigger test.
Step 4	**object list owner** *object-list-owner* **name** *object-list-name* **Example:** `Router(config-event-trigger-boolean)# object list owner owner1 name ObjectListA`	Configures the list of objects for Boolean trigger test.
Step 5	**event owner** *event-owner* **name** *event-name* **Example:** `Router(config-event-trigger-boolean)# event owner owner1 name EventA`	Configures event for the Boolean trigger type.
Step 6	**startup** **Example:** `Router(config-event-trigger-boolean)# startup`	Triggers an event if the test is performed successfully.
Step 7	**exit** **Example:** `Router(config-event-trigger-boolean)# exit`	Exits Boolean trigger test configuration mode.

Configuring Threshold Trigger Test

Perform this task to configure trigger parameters for the threshold trigger test. You should configure this trigger test in the event trigger configuration mode.

SUMMARY STEPS

1. **test threshold**
2. **object list owner** *object-list-owner* **name** *object-list-name*
3. **rising** *integer-value*
4. **rising event owner** *event-owner* **name** *event-name*
5. **falling** *integer-value*
6. **falling event owner** *event-owner* **name** *event-name*
7. **delta rising** *integer-value*
8. **delta rising event owner** *event-owner* **name** *event-name*
9. **delta falling** *integer-value*
10. **delta falling event owner** *event-owner* **name** *event-name*
11. **startup** {**rising | falling | rising-or-falling**}
12. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**test threshold** **Example:** `Router(config-event-trigger)# test threshold`	Enables threshold trigger test configuration mode.
Step 2	**object list owner** *object-list-owner* **name** *object-list-name* **Example:** `Router(config-event-trigger-threshold)# object list owner owner1 name ObjectListA`	Configures the list of objects for threshold trigger test.
Step 3	**rising** *integer-value* **Example:** `Router(config-event-trigger-threshold)# rising 100`	Sets the rising threshold to the specified value.
Step 4	**rising event owner** *event-owner* **name** *event-name* **Example:** `Router(config-event-trigger-threshold)# rising event owner owner1 name EventA`	Configures event for Threshold trigger test for rising threshold.
Step 5	**falling** *integer-value* **Example:** `Router(config-event-trigger-threshold)# falling 50`	Sets the falling threshold to the specified value.

	Command or Action	Purpose
Step 6	**falling event owner** *event-owner* **name** *event-name* **Example:** `Router(config-event-trigger-threshold)# falling event owner owner1 name EventB`	Configures event for Threshold trigger test for falling threshold.
Step 7	**delta rising** *integer-value* **Example:** `Router(config-event-trigger-threshold)# delta rising 30`	Sets the delta rising threshold to the specified value when the sampling method specified for the event trigger is delta.
Step 8	**delta rising event owner** *event-owner* **name** *event-name* **Example:** `Router(config-event-trigger-threshold)# delta rising event owner owner1 name EventC`	Configures event for Threshold trigger test for delta rising threshold.
Step 9	**delta falling** *integer-value* **Example:** `Router(config-event-trigger-threshold)# delta falling 10`	Sets the delta falling threshold to the specified value when the sampling method specified for the event trigger is delta.
Step 10	**delta falling event owner** *event-owner* **name** *event-name* **Example:** `Router(config-event-trigger-threshold)# delta falling event owner owner1 name EventAA`	Configures event for Threshold target test for delta falling threshold.
Step 11	**startup** {**rising** | **falling** | **rising-or-falling**} **Example:** `Router(config-event-trigger-threshold)# startup rising`	Triggers an event when the threshold trigger test conditions are met.
Step 12	**exit** **Example:** `Router(config-event-trigger-threshold)# exit`	Exits threshold trigger test configuration mode.

Configuring Expression MIB Using SNMP

Expression MIB can be configured using SNMP directly.

There are no Cisco IOS software configuration tasks associated with the Expression MIB. All configuration of Expression MIB functionality must be performed though applications using SNMP. This section provides a sample configuration session using a network management application on an external device. See the "Related Documents" section for information about configuring SNMP on your Cisco routing device.

All configuration of Expression MIB functionality must be performed though applications using SNMP. The following section provides a step-by-step Expression MIB configuration using SNMP research tools available for Sun workstations. The **setany** commands given below are executed using the SNMP application. Note that these commands are not Cisco IOS CLI commands. It is assumed that SNMP has been configured on your routing device.

In the following configuration, a wildcarded expression involving the addition of the counters ifInOctects and ifOutOctects are evaluated.

	Command	Purpose
Step 1	`setany -v2c $SNMP_HOST private expResourceDeltaMinimum.0 -i 60`	Sets the minimum delta interval that the system will accept.
Step 2	`setany -v2c $SNMP_HOST private expExpressionIndex.116.101.115.116 -g 9`	Sets the identification number used for identifying the expression. expName for example can be 'test' which is ascii 116.101.115.116.
Step 3	`setany -v2c $SNMP_HOST private expNameStatus.116.101.115.116 -i 5`	Creates an entry in the expNameStatusTable. **Note** When an entry is created in the expNameTable, this automatically creates an entry in the expExpressionTable.
Step 4	`setany -v2c $SNMP_HOST private expExpressionComment.9 -D "test expression"`	Sets the object to a comment to explain the use or meaning of the expression. Here the comment given is “test expression”.
Step 5	`setany -v2c $SNMP_HOST private expExpression.9 -D '$1 + $2'`	Sets the object expExpression to an expression that needs to be evaluated. In this expression the “$1” corresponds to the “ifInOctets” and the “$2” corresponds to the ifOutOctets and the expression signifies the addition of the 2 counter objects.
Step 6	`setany -v2c $SNMP_HOST private expObjectID.9.1 -d ifInOctets` `setany -v2c $SNMP_HOST private expObjectID.9.2 -d ifOutOctets`	The object identifiers used in the expression mentioned in the above set for calculation. Here “set” the number “9” suffixing the object expObjectID corresponds to the unique identifier used for identifying the expression and the number “1” after the number “9” is another unique identifier used for identifying an object within the expression. Set the expObjectID to the 2 objects used in forming the expression.

Step 7	`setany -v2c $SNMP_HOST private expObjectSampleType.9.1 -i 2`
	
	`setany -v2c $SNMP_HOST private expObjectSampleType.9.2 -i 2`
		Sets the type of sampling to be done for the objects in the expression. There are 2 types of sampling: a) Absolute b) Delta. Here we are setting the sample type to “Delta”.

Step 8	`setany -v2c $SNMP_HOST private expObjectIDWildcard.9.1 -i 1`
	
	`setany -v2c $SNMP_HOST private expObjectIDWildcard.9.2 -i 1`
		Specifies whether the expObjectID is wild carded or not. In this case both the expObjectID are wildcarded.

	Command	Purpose

Step 9	`setany -v2c $SNMP_HOST private expObjectStatus.9.1 -i 1`
	
	`setany -v2c $SNMP_HOST private expObjectStatus.9.2 -i 1`
		Sets the rows in the expObjectTable to active.

Step 10	`setany -v2c $SNMP_HOST private expNameStatus.116.101.115.116 -i 1`	Sets the row in the expNameTable to active so that the value of the expression can be evaluated. The value of the expression can now be obtained from the expValueTable.

Configuring Expression MIB using CLI

Expression MIB can be configured using SNMP directly. However, in the Cisco IOS Release 12.4(20)T, Expression MIB feature is enhanced to add CLIs to configure expressions. You should be familiar with expressions, object identifiers and sampling methods before configuring Expression MIB.

The following sections contain the tasks to configure Expression MIB:

- Configuring Expression MIB Scalar Objects, page 63
- Configuring Expressions, page 64

Configuring Expression MIB Scalar Objects

Expression MIB has the following scalar objects:

- expResourceDeltaMinimum
- expResourceDeltaWildcardInstanceMaximum

Perform this task to configure Expression MIB scalar objects.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp mib expression delta minimum** *seconds*
4. **snmp mib expression delta wildcard maximum** *number-of-instances*
5. **exit**

DETAILED STEPS

Step 1	`enable`
	
	Example:
	`Router> enable`
		Enables privileged EXEC mode.
		• Enter your password if prompted.

Step 2	`configure terminal`
	
	Example:
	`Router# configure terminal`
		Enters global configuration mode.

Step 3	**snmp mib expression delta minimum** *seconds* **Example:** `Router(config)# snmp mib expression delta minimum 20`	(Optional) Sets the minimum delta interval in seconds. **Note** Application may use larger values for this minimum delta interval to lower the impact of constantly computing deltas. For larger delta sampling intervals, the application samples less often and has less overhead. By using this command, you can enforce a lower overhead for all expressions created after the delta interval is set.
Step 4	**snmp mib expression delta wildcard maximum** *number-of-instances* **Example:** `Router(config)# snmp mib expression delta maximum 120`	(Optional) Limits the maximum number of dynamic instance entries for wildcarded delta objects in expressions. For a given delta expression, the number of dynamic instances is the number of values that meet all criteria to exist, times the number of delta values in the expression. There is no preset limit for the instance entries and it is dynamic based on a system's resources.
Step 5	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode.

Configuring Expressions

Perform this task to configure an expression.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp mib expression owner** *expression-owner* **name** *expression-name*
4. **description** *expression-description*
5. **expression** *expression*
6. **delta interval** *seconds*
7. **value type** {**counter32 | unsigned32 | timeticks | integer32 | ipaddress | octetstring | objectid | counter64**}
8. **enable**
9. **object** *object-number*
10. **id** *object-identifier*
11. **wildcard**
12. **discontinuity object** *discontinuity-object-id* [**wildcard**] [**type** {**timeticks | timestamp | date-and-time**}]
13. **conditional object** *conditional-object-id*
14. **sample** {**absolute | delta | changed**}
15. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**snmp mib expression owner** *expression-owner* **name** *expression-name* **Example:** `Router(config-expression)# snmp mib expression owner owner1 name ExpA`	Enables the expression to be configured.
Step 4	**description** *expression-description* **Example:** `Router(config-expression)# description this expression is created for the sysLocation MIB object`	Configures description for expression.
Step 5	**expression** *expression* **Example:** `Router(config-expression)# expression ($1+$2)*800/$3`	Configures the expression to be evaluated. **Note** The expression are in ANSI C syntax. However, the variables in an expression are defined as combination of the dollar sign ($) and an integer that corresponds to the object number of the object used in evaluating the expression.
Step 6	**delta interval** *seconds* **Example:** `Router(config-expression)# delta interval 180`	Configures the sampling interval for objects in the expression if the sampling method is delta.
Step 7	**value type** {**counter32** | **unsigned32** | **timeticks** | **integer32** | **ipaddress** | **octetstring** | **objectid** | **counter64**} **Example:** `Router(config-expression)# value type counter32`	Sets the specified value type for expression.
Step 8	**enable** **Example:** `Router(config-expression)# enable`	Enables expression for evaluation.

	Command or Action	Purpose
Step 9	**object** *object-number* **Example:** `Router(config-expression)# object 2`	Configures the objects that are used for evaluating an expression. The object number is used to associate the object with the variables in the Expression. The variable corresponding to the object is $ and the object number. Thus the variable in the example used here corresponds to $10.
Step 10	**id** *object-identifier* **Example:** `Router(config-expression-object)# id ifInOctets`	Configures the object identifier.
Step 11	**wildcard** **Example:** `Router(config-expression-object)# wildcard`	(Optional) Enables wildcarded search for objects used in evaluating expression.
Step 12	**discontinuity object** *discontinuity-object-id* [**wildcard**][**type** {**timeticks** |**timestamp** | **date-and-time**}] **Example:** `Router(config-expression-object)# discontinuity object sysUpTime`	(Optional) Configures the discontinuity properties for the object if the object sampling type is set to delta or changed. The discontinuity object ID supports normal checking for a discontinuity in a counter. • Using the wildcard keyword, you can enable wildcarded search for the objects with discontinuity properties. • Using the type keyword, you can set value for objects with discontinuity properties.
Step 13	**conditional object** *conditional-object-id* **[wildcard]** **Example:** `Router(config-expression-object)# conditional object mib-2.90.1.3.1.1.2.3.112.99.110.4.101.120.112.53`	(Optional) Configures the conditional object identifier. • Using the wildcard keyword, you can enable wildcarded search for the conditional objects with discontinuity properties.

	Command or Action	Purpose
Step 14	**sample** {**absolute** | **delta** | **changed**} **Example:** `Router(config-expression-object)# sample delta`	Enables the specified sampling method for the object. This example uses the delta sampling method. You can set any of the three sampling methods; absolute, delta, and changed. • Absolute sampling—Uses the value of the MIB object during sampling. • Delta sampling—Uses the last sampling value maintained in the application. This method requires the applications to do continuous sampling. • Changed sampling—Uses the changed value of the object since the last sample.
Step 15	**exit** **Example:** `Router(config-expression-object)# exit`	Exits expression object configuration mode.

Configuration Examples for SNMP Support

This section provides the following configuration examples:

Configuring SNMPv1, SNMPv2c, and SNMPv3: Example

The following example shows how to enable SNMPv1, SNMPv2c, and SNMPv3. The configuration permits any SNMP manager to access all objects with read-only permissions using the community string named public. This configuration does not cause the router to send traps.

```
snmp-server community public
```

The following example shows how to permit SNMP access to all objects with read-only permission using the community string named public. The router also will send ISDN traps to the hosts 172.16.1.111 and 172.16.1.33 using SNMPv1 and to the host 172.16.1.27 using SNMPv2c. The community string named public is sent with the traps.

```
snmp-server community public
snmp-server enable traps isdn
snmp-server host 172.16.1.27 version 2c public
snmp-server host 172.16.1.111 version 1 public
```

```
snmp-server host 172.16.1.33 public
```

The following example shows how to allow read-only access for all objects to members of access list 4 that specify the comaccess community string. No other SNMP managers have access to any objects. SNMP Authentication Failure traps are sent by SNMPv2c to the host example.com using the community string named public.

```
snmp-server community comaccess ro 4
snmp-server enable traps snmp authentication
snmp-server host example.com version 2c public
```

The following example shows how to configure a remote user to receive traps at noAuthNoPriv security level when the SNMPv3 security model is enabled:

```
snmp-server group group1 v3 noauth
snmp-server user remoteuser1 group1 remote 10.12.8.4
snmp-server host 10.12.8.4 informs version 3 noauth remoteuser config
```

The following example shows how to configure a remote user to receive traps at the authNoPriv security level when the SNMPv3 security model is enabled:

```
snmp-server group group2 v3 auth
snmp-server user AuthUser group2 remote 10.12.8.4 v3 auth md5 password1
```

The following example shows how to configure a remote user to receive traps at the priv security level when the SNMPv3 security model is enabled:

```
snmp-server group group3 v3 priv
snmp-server user PrivateUser group3 remote 10.12.8.4 v3 auth md5 password1 priv access
des56
```

The following example shows how to send Entity MIB inform notifications to the host example.com. The community string is restricted. The first line enables the router to send Entity MIB notifications in addition to any traps or informs previously enabled. The second line specifies that the notifications should be sent as informs, specifies the destination of these informs, and overwrites the previous **snmp-server host** commands for the host example.com.

```
snmp-server enable traps entity
snmp-server host informs example.com restricted entity
```

The following example shows how to send the SNMP and Cisco environmental monitor enterprise-specific traps to address 172.30.2.160:

```
snmp-server enable traps
snmp-server host 172.30.2.160 public snmp envmon
```

The following example shows how to enable the router to send all traps to the host example.com using the community string public:

```
snmp-server enable traps
snmp-server host example.com public
```

The following example shows a configuration in which no traps are sent to a host. The BGP traps are enabled for all hosts, but only the ISDN traps are enabled to be sent to a host.

```
snmp-server enable traps bgp
snmp-server host host1 public isdn
```

The following example shows how to enable a router to send all informs to the host example.com using the community string named public:

```
snmp-server enable traps
snmp-server host example.com informs version 2c public
```

In the following example, the SNMP manager is enabled and the session timeout is set to a value greater than the default:

```
snmp-server manager
snmp-server manager session-timeout 1000
```

Configuring IfAlias Long Name Support: Example

In the following example a long description is applied to the Ethernet interface in slot 1, port adapter 0, and port 0:

```
Router# configure terminal
Router(config)# interface Ethernet1/0/0
Router(config-if)# description ethernet1/0/0 this is a test of a description that exceeds
64 characters in length
Router(config-if)# ip address 192.168.134.55 255.255.255.0
Router(config-if)# no ip directed-broadcast
Router(config-if)# no ip route-cache distributed
```

Assuming that ifAlias long name support is not yet enabled (the default), the following example shows the results of a mibwalk operation from an NMS:

```
***** SNMP QUERY STARTED *****
 .
 .
 .
 ifXEntry.18.10 (octets) (zero-length)
 ifXEntry.18.11 (octets) ethernet1/0/0 this is a test of a description that exceeds 64 ch
 ifXEntry.18.12 (octets) (zero-length)
 .
 .
 .
```

The following output shows the description that is displayed at the CLI:

```
Router# show interface Ethernet0/0/0

Ethernet1/0/0 is administratively down, line protocol is down
  Hardware is Lance, address is 0010.7b4d.7046 (bia 0010.7b4d.7046)
  Description: ethernet1/0/0 this is a test of a description that exceeds 64 chh
  MTU 1500 bytes, BW 10000 Kbit, DLY 1000 usec,
     reliability 252/255, txload 1/255, rxload 1/255
 .
 .
 .
```

In the following example, ifAlias long name support is enabled and the description is displayed again:

```
Router(config)# snmp ifmib ifalias long
Router(config)# interface Ethernet1/0/0
Router(config-if)# description ethernet1/0/0 this is a test of a description that exceeds
64 characters in length
Router(config)# end
Router# show interface Ethernet1/0/0

Ethernet1/0/0 is administratively down, line protocol is down
  Hardware is Lance, address is 0010.7b4d.7046 (bia 0010.7b4d.7046)
  Description: ethernet1/0/0 this is a test of a description that exceeds 64 characters in
length
  MTU 1500 bytes, BW 10000 Kbit, DLY 1000 usec,
     reliability 252/255, txload 1/255, rxload 1/255
 .
 .
```

```
 .
***** SNMP QUERY STARTED *****
 .
 .
 .
 ifXEntry.18.10 (octets) (zero-length)
 ifXEntry.18.11 (octets) ethernet1/0/0 this is a test of a description that exceeds 64
characters in length
 ifXEntry.18.12 (octets) (zero-length)
 .
 .
 .
```

Configuring IfIndex Persistence: Example

The following example shows how to enable IfIndex persistence globally:

```
Router# configure terminal
Router(config)# snmp-server ifindex persist
```

The following example shows how to enable IfIndex persistence on Ethernet interface:

```
Router# configure terminal
Router(config)# interface ethernet 0/1
Router(config)# snmp-server ifindex persist
```

Configuring SNMP Support for VPNs: Example

In the following example all SNMP notifications are sent to example.com over the VRF named trap-vrf:

```
Router(config)# snmp-server host example.com vrf trap-vrf
```

In the following example the VRF named “traps-vrf” is configured for the remote server 172.16.20.3:

```
Router(config)# snmp-server engineID remote 172.16.20.3 vrf traps-vrf
80000009030000B064EFE100
```

Enabling Event MIB Persistence: Example

The following example shows how to enable Event MIB Persistence using the **snmp mib persist event** command in global configuration mode:

```
Router(config)# snmp mib persist event
Router# write mib-data
```

Enabling Expression MIB Persistence: Example

The following example shows how to enable Expression MIB Persistence using the **snmp mib persist expression** command in global configuration mode:

```
Router(config)# snmp mib persist expression
Router# write mib-data
```

Configuring Event MIB: Example

The following example shows how to configure scalar variables for an event:

```
Router# configure terminal
Router(config)# snmp mib event sample minimum 10
Router(config)# snmp mib event sample instance maximum 50
Router(config)# exit
```

The following example shows how to configure object list for an event:

```
Router# configure terminal
Router(config)# snmp mib event object list owner owner1 name objectA number 1
Router(config-event-objlist)# object id ifInOctets
Router(config-event-objlist)# wildcard
Router(config-event-objlist)# exit
```

The following example shows how to configure an event:

```
Router# configure terminal
Router(config)# snmp mib event owner owner1 event EventA
Router(config-event)# description "eventA is an RMON event."
Router(config-event)# enable
Router(config-event)# exit
```

The following example shows how to set the notification action for an event:

```
Router(config-event)# action notification
Router(config-event-action-notification)# object id ifInOctets
Router(config-event-action-notification)# exit
```

The following example shows how to set actions for an event:

```
Router(config-event)# action set
Router(config-event-action-set)# object id ifInOctets
Router(config-event-action-set)# value 10
Router(config-event-action-set)# exit
```

The following example shows how to configure trigger for an event:

```
Router# configure terminal
Router(config)# snmp mib event trigger owner owner1 name EventTriggerA
Router(config-event-trigger)# description EventTriggerA is an RMON alarm.
Router(config-event-trigger)# frequency 120
Router(config-event-trigger)# object list owner owner1 name ObjectListA
Router(config-event-trigger)# object id ifInOctets
Router(config-event-trigger)# enable
Router(config-event-trigger)# exit
```

The following example shows how to configure existence trigger test:

```
Router(config-event-trigger)# test existence
Router(config-event-trigger-existence)# event owner owner1 name EventA
Router(config-event-trigger-existence)# object list owner owner1 name ObjectListA
Router(config-event-trigger-existence)# type present
Router(config-event-trigger-existence)# startup present
Router(config-event-trigger-existence)# exit
```

The following example shows how to configure Boolean trigger test:

```
Router(config-event-trigger)# test boolean
Router(config-event-trigger-boolean)# comparison unequal
Router(config-event-trigger-boolean)# value 10
Router(config-event-trigger-boolean)# object list owner owner1 name ObjectListA
Router(config-event-trigger-boolean)# event owner owner1 name EventA
```

```
Router(config-event-trigger-boolean)# startup
Router(config-event-trigger-boolean)# exit
```

The following example shows how to configure threshold trigger test:

```
Router(config-event-trigger)# test threshold
Router(config-event-trigger-threshold)# object list owner owner1 name ObjectListA
Router(config-event-trigger-threshold)# rising 100
Router(config-event-trigger-threshold)# rising event owner owner1 name EventA
Router(config-event-trigger-threshold)# falling 50
Router(config-event-trigger-threshold)# falling event owner owner1 name EventA
Router(config-event-trigger-threshold)# delta rising 30
Router(config-event-trigger-threshold)# delta rising event owner owner1 name EventA
Router(config-event-trigger-threshold)# delta falling 10
Router(config-event-trigger-threshold)# delta falling event owner owner1 name EventA
Router(config-event-trigger-threshold)# startup rising
Router(config-event-trigger-threshold)# exit
```

Configuring Expression MIB: Example

The following example shows how to configure Expression MIB using the **snmp mib expression** command in global configuration mode:

```
Router(config)# snmp mib expression owner pcn name exp6
Router(config-expression)# description this expression is created for the sysLocation MIB
object
Router(config-expression)# expression ($1+$2)*800/$3
Router(config-expression)# delta interval 120
Router(config-expression)# value type counter32
Router(config-expression)# enable
Router(config-expression)# object 2
Router(config-expression-object)# id ifInOctets
Router(config-expression-object)# wildcard
Router(config-expression-object)# discontinuity object sysUpTime
Router(config-expression-object)# conditional object
mib-2.90.1.3.1.1.2.3.112.99.110.4.101.120.112.53 wildcard
Router(config-expression-object)# sample delta
Router(config-expression-object)# exit
```

Additional References

The following sections provide references related to configuring SNMP support.

Related Documents

Related Topic	Document Title
SNMP commands: complete command syntax, command mode, command history, defaults, usage guidelines, and examples	*Cisco IOS Network Management Command Reference*
Cisco IOS implementation of RFC 1724, RIP Version 2 MIB Extensions	*RIPv2 Monitoring with SNMP Using the RFC 1724 MIB Extensions* feature module
DSP Operational State Notifications for notifications to be generated when a digital signaling processor (DSP) is used	*DSP Operational State Notifications* feature module

Standards

Standard	Title
CBC-DES (DES-56) standard	*Symmetric Encryption Protocol*
STD: 58	*Structure of Management Information Version 2 (SMIv2)*

MIBs

MIB	MIBs Link
• Circuit Interface Identification MIB • Cisco SNMPv2 • Ethernet-like Interfaces MIB • Event MIB • Expression MIB Support for Delta, Wildcarding, and Aggregation • Interfaces Group MIB (IF-MIB) • Interfaces Group MIB Enhancements • MIB Enhancements for Universal Gateways and Access Servers • MSDP MIB • NTP MIB • Response Time Monitor MIB	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
RFC 1067	*A Simple Network Management Protocol*
RFC 1091	*Telnet terminal-type option*

RFC	Title
RFC 1098	*Simple Network Management Protocol (SNMP)*
RFC 1157	*Simple Network Management Protocol (SNMP)*
RFC 1213	*Management Information Base for Network Management of TCP/IP-based internets:MIB-II*
RFC 1215	*Convention for defining traps for use with the SNMP*
RFC 1901	*Introduction to Community-based SNMPv2*
RFC 1905	*Common Management Information Services and Protocol over TCP/IP (CMOT)*
RFC 1906	*Telnet X Display Location Option*
RFC 1908	*Simple Network Management Protocol (SNMP)*
RFC 2104	*HMAC: Keyed-Hashing for Message Authentication*
RFC 2206	*RSVP Management Information Base using SMIv2*
RFC 2213	*Integrated Services Management Information Base using SMIv2*
RFC 2214	*Integrated Services Management Information Base Guaranteed Service Extensions using SMIv2*
RFC 2271	*An Architecture for Describing SNMP Management Frameworks*
RFC 2570	*Introduction to Version 3 of the Internet-standard Network Management Framework*
RFC 2578	*Structure of Management Information Version 2 (SMIv2)*
RFC 2579	*Textual Conventions for SMIv2*
RFC 2580	*Conformance Statements for SMIv2*
RFC 2981	*Event MIB*
RFC 2982	*Distributed Management Expression MIB*
RFC 3413	*SNMPv3 Applications*
RFC 3415	*View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)*

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/techsupport

Feature Information for Configuring SNMP Support

Table 2 lists the features in this module and provides links to specific configuration information. Only features that were introduced or modified in Cisco IOS Releases 12.2(1) or 12.0(3)S or a later release appear in the table.

For information about a feature in this technology that is not documented here, see the SNMP Features Roadmap.

Not all commands may be available in your Cisco IOS software release. For release information about a specific command, see the command reference documentation.

Use Cisco Feature Navigator to find information about platform support and software image support. Cisco Feature Navigator enables you to determine which Cisco IOS and Catalyst OS software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Note Table 2 lists only the Cisco IOS software release that introduced support for a given feature in a given Cisco IOS software release train. Unless noted otherwise, subsequent releases of that Cisco IOS software release train also support that feature.

Table 2 Feature Information for Configuring SNMP Support

Feature Name	Releases	Feature Information
Distributed Management Event and Expression MIB Persistence	12.0(5)T 12.0(12)S 12.1(3)T 12.2(4)T 12.2(4)T3	The MIB Persistence features allow the SNMP data of a MIB to be persistent across reloads; that is, MIB information retains the same set object values each time a networking device reboots. MIB Persistence is enabled by using the **snmp mib persist** command, and the MIB data of all MIBs that have had persistence enabled using this command is then written to NVRAM storage by using the **write mib-data** command. Any modified MIB data must be written to NVRAM memory using the **write mib-data** command. The following sections provide information about this module: • MIB Persistence, page 10 • Configuring MIB Persistence, page 45
Interface Index Display and Interface Alias Long Name Support for SNMP	12.2(2)T	The Interface Index Display for SNMP feature introduces new commands and command modifications that allow advanced users of SNMP to view information about the interface registrations directly on the managed agent. You can display MIB information from the agent without using an external NMS. This feature addresses three objects in the Interfaces MIB: *ifIndex*, *ifAlias*, and *ifName*. For complete definitions of these objects, see the IF-MIB.my file available from the Cisco SNMPv2 MIB website at ftp://ftp.cisco.com/pub/mibs/v2/. The following sections provide information about this feature: • Detailed Interface Registration Information, page 8 • Configuring Interface Index Display and Interface Indexes and Long Name Support, page 38

Table 2 Feature Information for Configuring SNMP Support (continued)

Feature Name	Releases	Feature Information
SNMP Notification Logging	12.0(22)S 12.2(13)T	The SNMP Notification Logging feature adds Cisco IOS CLI commands to change the size of the notification log, to set the global ageout value for the log, and to display logging summaries at the command line. The following sections provide information about this feature: • SNMP Notification Logging, page 13 • Configuring SNMP Notifications, page 31
SNMP Support for VPNs	12.0(23)S 12.2(2)T 12.2(33)SXH 12.2(33)SB	The SNMP Support for VPNs feature allows SNMP traps and informs to be sent and received using VRF tables. In particular, this feature adds support to Cisco IOS software for sending and receiving SNMP traps and informs specific to individual VPNs. The following sections provide information about this feature: • SNMP Support for VPNs, page 9 • Configuring SNMP Support for VPNs, page 41
Circuit Interface Identification Persistence for SNMP feature	12.1(3)T	This feature can be used to identify individual circuit-based interfaces for SNMP monitoring. The following section provides information about this feature: • Circuit Interface Identification Persistence, page 11
Circuit Interface Identification MIB	Cisco IOS XE Release 2.1	This feature was introduced on Cisco ASR 1000 series routers.
Distributed Management Event MIB Conformance to RFC 2981	Cisco IOS XE Release 2.1	This feature was introduced on Cisco ASR 1000 series routers.
SNMP (Simple Network Management Protocol)	Cisco IOS XE Release 2.1	This feature was introduced on Cisco ASR 1000 series routers.
SNMP Version 3	Cisco IOS XE Release 2.1	This feature was introduced on Cisco ASR 1000 series routers.
SNMPv2C	Cisco IOS XE Release 2.1	This feature was introduced on Cisco ASR 1000 series routers.
Interface IfIndex Persistence	12.2(15)T	This feature allows interfaces to be identified with unique values which will remain constant even when a device is rebooted. These interface identification values are used for network monitoring and management using SNMP. The following sections provide information about this feature: • Interface IfIndex Persistence, page 9 • Configuring Interface IfIndex Persistence, page 43
SNMP Diagnostics	12.4(20)T 12.2(33)SRE	The SNMP Diagnostics feature adds Cisco IOS CLI commands to display the object identifiers that are recently requested by the network management system, and to display the SNMP debug messages. The following commands were introduced or modified: **show snmp stats oid** and **debug snmp detail**.

Table 2 Feature Information for Configuring SNMP Support (continued)

Feature Name	Releases	Feature Information
Event MIB and Expression MIB CLIs	12.4(20)T 12.2(33)SRE	The Event MIB and Expression MIB feature introduces CLIs to configure the Event MIB and Expression MIB. The following section provides information about configuring Event MIB: • Configuring Event MIB Using SNMP, page 48 • Configuring Event MIB Using CLI, page 50 The following section provides information about configuring Expression MIB: • Configuring Expression MIB Using SNMP, page 61 • Configuring Expression MIB using CLI, page 63 The following commands were introduced by this feature: **action (event)**, **comparison**, **conditional object**, **delta (test threshold)**, **delta interval**, **description (event)**, **description (expression)**, **description (trigger)**, **discontinuity object**, **enable (event)**, **enable (expression)**, **event owner**, **enable (expression)**, **expression**, **falling (test threshold)**, **frequency (event trigger)**, **object (expression)**, **object-id (action notification)**, **object id (action set)**, **object id (event trigger)**, **object list (trigger test)**, **object wildcard**, **rising (test threshold)**, **sample (event-trigger)**, **sample (expression)**, **snmp mib event object list**, **snmp mib event owner**, **snmp mib event trigger**, **snmp mib expression delta**, **snmp mib expression owner**, **startup (test existence)**, **startup (test boolean)**, **startup (test threshold)**, **test (event trigger)**, **type (test existence)**, **value (test boolean)**, **value (event configuration)**, **value type**, **wildcard (event and expression)**.
SNMP Trap Simulations	12.2(33)SXI 12.2 (33)SRE	The SNMP Trap Simulation feature introduces the **test snmp trap** CLIs to verify the reception of the SNMP, syslog, and config-copy notifications by the SNMP manager, in a simulated scenario.

Glossary

ifAlias—SNMP Interface Alias. The ifAlias is an object in the Interfaces MIB (IF-MIB). The ifAlias is an alias name for the interface as specified by a network manager that provides a nonvolatile description for the interface. For a complete definition, see the IF-MIB.my file.

ifIndex—SNMP Interface Index. The ifIndex is an object in the Interfaces MIB (IF-MIB). The ifIndex is a unique integer assigned to every interface (including subinterfaces) on the managed system when the interface registers with the IF-MIB. For a complete definition, see the IF-MIB.my file.

OID—MIB object identifier. An object identifier is expressed as a series of integers or text strings. Technically, the numeric form is the *object name* and the text form is the *object descriptor*. In practice, both are called object identifiers, or OIDs. For example, the object name for the interfaces MIB is 1.3.6.1.2.1.2, and the object descriptor is 'iso.internet.mgmt.mib-2.interfaces' but either can be referred to as the OID. An OID can also be expressed as a combination of the two, such as iso.internet.2.1.2.

AES and 3-DES Encryption Support for SNMP Version 3

First Published: May 2005
Last Updated: October 2, 2009

The AES and 3-DES Encryption Support for SNMP Version 3 feature enhances the encryption capabilities of SNMP version 3. Data Encryption Standard (DES) support was introduced in Cisco IOS Release 12.0 and expanded in Cisco IOS Release 12.1. This support for Simple Network Management Protocol (SNMP) version 3 User-Based Security Model (USM) is complaint with RFC 3414, which defines DES as the only required method of message encryption for SNMP version 3 authPriv mode.

The AES and 3-DES Encryption Support for SNMP Version 3 feature adds Advanced Encryption Standard (AES) 128-bit encryption in compliance with RFC 3826. RFC 3826 extensions have been included in the SNMP-USM-AES-MIB. In addition, Cisco-specific extensions to support Triple-Data Encryption Algorithm (3-DES) and AES 192-bit and 256-bit encryption have been added to the CISCO-SNMP-USM-MIB. Additional information can be found in the Internet-Draft titled *Extension to the User-Based Security Model (USM) to Support Triple-DES EDE in "Outside" CBC Mode,* which can be found at the following URL: http://www.snmp.com/eso/draft-reeder-snmpv3-usm-3desede-00.txt.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the "Feature Information for AES and 3-DES Encryption Support for SNMP Version 3" section on page 7.

Use Cisco Feature Navigator to find information about platform support and Cisco IOS and Catalyst OS software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

Contents

Prerequisites for AES and 3-DES Encryption Support for SNMP Version 3

- The network management station (NMS) must support SNMP version 3 to use this feature of the SNMP agent.
- This feature is available in only Cisco IOS software images where encryption algorithms are supported.

Information About AES and 3-DES Encryption Support for SNMP Version 3

To configure the AES and 3-DES Encryption Support for SNMP Version 3 feature, you should understand the following concepts:

SNMP Architecture

The architecture for describing Internet Management Frameworks contained in RFC 3411 describes the SNMP engine as composed of the following components:

- Dispatcher
- Message Processing Subsystem
- Security Subsystem
- Access Control Subsystem

Applications make use of the services of these subsystems. It is important to understand the SNMP architecture and the terminology of the architecture to understand where the Security Model fits into the architecture and interacts with the other subsystems within the architecture. The information is contained in RFC 3411 and your are encouraged to review this RFC to obtain an understanding of the SNMP architecture and subsystem interactions.

Encryption Key Support

In the AES and 3-DES Encryption Support for SNMP Version 3 feature the Cipher Block Chaining/Data Encryption Standard (CBC-DES) is the privacy protocol. Originally only DES was supported (as per RFC 3414). This feature adds support for AES-128 (as per RFC 3826) and AES-192, AES-256 and 3-DES (as per CISCO-SNMP-USM-OIDS-MIB).

- AES encryption uses the Cipher Feedback (CFB) mode with encryption key sizes of 128, 192, or 256 bits.
- 3DES encryption uses the 168-bit key size for encryption.

The AES Cipher Algorithm in the SNMP User-based Security Model draft describes the use of AES with 128-bit key size. However, the other options are also implemented with the extension to use the USM. There is currently no standard for generating localized keys for 192- or 256-bit size keys for AES or for 168-bit size key for 3-DES. There is no authentication protocol available with longer keys.

Management Information Base Support

The AES and 3-DES Encryption Support for SNMP Version 3 feature supports the selection of privacy protocols through the CLI and the Management Information Base (MIB). A new standard MIB, SNMP-USM-AES-MIB, provides support for the 128-bit key in AES. The extended options of AES with 192- or 256-bit keys and 3-DES are supported as extensions to the SNMP-USM-MIB, in the Cisco-specific MIB, CISCO-SNMP-USM-EXT-MIB.

How to Configure AES and 3-DES Encryption Support for SNMP Version 3

This section contains the following procedures:

- Adding a New User to an SNMP Group, page 3
- Verifying SNMP User Configuration, page 4

Adding a New User to an SNMP Group

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp-server user** *username group-name* [**remote** *host* [**udp-port** *port*] [**vrf** *vrf-name*]] {**v1** | **v2c** | **v3** [**encrypted**] [**auth** {**md5** | **sha**} *auth-password*]} [**access** [**ipv6** *nacl*] [**priv** {**des** | **3des** | **aes** {**128** | **192** |**256**}} *privpassword*] {*acl-number* | *acl-name*}]

DETAILED STEPS

	Command	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enters privileged EXEC mode. • Enter your password when prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**snmp-server user** *username group-name* [**remote** *host* [**udp-port** *port*][**vrf** *vrf-name*]] {**v1** | **v2c** | **v3** [**encrypted**] [**auth** {**md5** | **sha**} *auth-password*]} [**access** [**ipv6** *nacl*] [**priv** {**des** | **3des** | **aes** {**128** | **192** |**256**}} *privpassword*] {*acl-number* | *acl-name*}] **Example:** `Router(config)# snmp-server user new-user new-group v3 auth md5 secureone priv aes 128 privatetwo 2`	Adds an SNMP user, specifies a group to which the user belongs, specifies the authorization algorithm to be used (MD5 or SHA), specifies the privacy algorithm to be used (DES, 3-DES, AES, AES-192, or AES-256), and specifies the password to be associated with this privacy protocol.

Verifying SNMP User Configuration

To display information about the configured characteristics of Simple Network Management Protocol (SNMP) users, use the **show snmp user** command in privileged EXEC mode.

SUMMARY STEPS

1. **enable**
2. **show snmp user** [*username*]

Note The **show snmp user** command displays all the users configured on the router. However, unlike other SNMP configurations, the **snmp-server user** command will not appear on the "show running" output.

DETAILED STEPS

Step 1 **enable**

Enters privileged EXEC mode. Enter your password when prompted.

Step 2 **show snmp user** [*username*]

The following example specifies the username as abcd, the engine ID string as 00000009020000000C025808, and the storage type as nonvolatile:

```
Router# show snmp user abcd

User name: abcd
Engine ID: 00000009020000000C025808
storage-type: nonvolatile       active access-list: 10
```

```
Rowstatus: active
Authentication Protocol: MD5
Privacy protocol: 3DES
Group name: VacmGroupName


Group name: VacmGroupName
```

Additional References

The following sections provide references related to the AES and 3-DES Encryption Support for SNMP Version 3 feature.

Related Documents

Related Topic	Document Title
SNMP configuration tasks	*Cisco IOS Network Management Configuration Guide*
SNMP commands: complete command syntax, command mode, command history, defaults, usage guidelines, and examples	*Cisco IOS Network Management Command Reference*

Standards

Standard	Title
draft-reeder-snmpv3-usm-3desede-00.txt	*Extension to the User-Based Security Model (USM) to Support Triple-DES EDE in "Outside" CBC Mode*

MIBs

MIB	MIBs Link
• SNMP-USM-AES-MIB • CISCO-SNMP-USM-OIDS-MIB	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
RFC 2574	*User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)*
RFC 3411	*Architecture for Describing Internet Management Frameworks*
RFC 3414	*User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)*
RFC 3826	*The Advanced Encryption Standard (AES) Cipher Algorithm in the SNMP User-based Security Model*

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/techsupport

Feature Information for AES and 3-DES Encryption Support for SNMP Version 3

Table 1 lists the release history for this feature.

Not all commands may be available in your Cisco IOS software release. For release information about a specific command, see the command reference documentation.

Use Cisco Feature Navigator to find information about platform support and software image support. Cisco Feature Navigator enables you to determine which Cisco IOS and Catalyst OS software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Note Table 1 lists only the Cisco IOS software release that introduced support for a given feature in a given Cisco IOS software release train. Unless noted otherwise, subsequent releases of that Cisco IOS software release train also support that feature.

Table 1 Feature Information for AES and 3-DES Encryption Support for SNMP Version 3

Feature Name	Releases	Feature Information
AES and 3-DES Encryption Support for SNMP Version 3	12.4(2)T 12.2(33)SRB 12.2(33)SB	The AES and 3-DES Encryption Support for SNMP Version 3 feature enhances the encryption capabilities of SNMP version 3. DES support was introduced in Cisco IOS Release 12.0 and expanded in Cisco IOS Release 12.1. This support for SNMP version 3 USM is complaint with RFC 3414, which defines DES as the only required method of message encryption for SNMP version 3 authPriv mode. The AES and 3-DES Encryption Support for SNMP Version 3 feature adds AES 128-bit encryption in compliance with RFC 3826. AES and 3-DES Encryption Support for SNMP Version 3 was introduced in Cisco IOS Release 12.4(2)T. AES and 3-DES Encryption Support for SNMP Version 3 was integrated into Cisco IOS Release 12.2(33)SRB. AES and 3-DES Encryption Support for SNMP Version 3 was integrated into Cisco IOS Release 12.2(33)SB.

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

SNMP Support for VLAN Subinterfaces

Feature History

Release	Modification
Cisco IOS	For information about feature support in Cisco IOS software, use Cisco Feature Navigator.

This feature module describes the SNMP Support for VLAN Subinterfaces feature. It includes information on the benefits of the new feature, supported platforms, supported standards, and the commands necessary to configure the SNMP Support for VLAN Subinterfaces feature.

This document includes the following sections:

Feature Overview

The SNMP Support for VLAN Subinterfaces feature provides mib-2 interfaces sparse table support for Fast Ethernet subinterfaces. This enhancement is similar to the functionality supported in Frame Relay subinterfaces.

Benefits

Sparse table support for the interfaces table on Fast Ethernet subinterfaces provides customers accustomed to Frame Relay subinterfaces the same functionality.

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

Supported Platforms

- Cisco 2600 series
- Cisco 3600 series
- Cisco 4000-m series
- Cisco 7200 series
- Cisco 7500 series

Supported Standards, MIBs, and RFCs

Standards

None

MIBs

None

For descriptions of supported MIBs and how to use MIBs, see the Cisco MIB web site on CCO at http://www.cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml.

RFCs

- RFC 1573

Configuration Tasks

To configure SNMP Support for VLAN Subinterfaces, complete the tasks in the following section:

- Enabling the SNMP Agent on VLAN Subinterfaces

Enabling the SNMP Agent on VLAN Subinterfaces

To enable the SNMP agent on VLAN subinterfaces, use the following commands in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# snmp community public`	Enables the SNMP agent for remote access.
Step 2	`Router(config)# interface FastEthernet slot/port`	Selects a particular Fast Ethernet interface for configuration.
Step 3	`Router(config)#  encapsulation isl vlan-identifier`	Enables the Inter-Switch Link.
Step 4	`Router(config)# ip address ip-address mask`	Sets a primary or secondary IP address for an interface.

Verifying VLAN Subinterfaces

To display traffic count on subinterfaces, use the following command in privileged EXEC mode:

Command	Purpose
`Router# show vlans`	Displays VLAN subinterfaces.

Configuration Examples

This section provides the following configuration example:

- Enabling the SNMP Agent for VLAN Subinterfaces Example

Enabling the SNMP Agent for VLAN Subinterfaces Example

The following configuration example shows you how to enable the SNMP agent on the router with VLAN subinterfaces to monitor the SNMP application remotely:

```
snmp community public
!
interface FastEthernet4/0.100
 encapsulation isl 100
 ip address 192.168.10.21 255.255.255.0
!
interface FastEthernet4/0.200
 encapsulation isl 200
 ip address 172.21.200.11 255.255.255.0
!
interface FastEthernet4/1.1
 encapsulation isl 10
 ip address 171.69.2.111 255.255.255.0
```

Command Reference

The following commands are introduced or modified in the feature or features documented in this module. For information about these commands, see the *Cisco IOS Network Management Command Reference* at http://www.cisco.com/en/US/docs/ios/netmgmt/command/reference/nm_book.html. For information about all Cisco IOS commands, go to the Command Lookup Tool at http://tools.cisco.com/Support/CLILookup or to the *Cisco IOS Master Commands List*.

- **show vlans**

SNMP MIB

Periodic MIB Data Collection and Transfer Mechanism

First Published: January 20, 2003
Last Updated: July 4, 2008

This document describes how to periodically transfer selected MIB data from Cisco IOS-based devices to specified Network Management Systems (NMS).

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the "Feature Information for Periodic MIB Data Collection and Transfer Mechanism" section on page 19.

Use Cisco Feature Navigator to find information about platform support and Cisco IOS and Catalyst OS software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Contents

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

Prerequisites for Periodic MIB Data Collection and Transfer Mechanism

To use this feature, you should be familiar with the Simple Network Management Protocol (SNMP) model of management information. You should also know what MIB information you want to monitor on your network devices, and the OIDs or object names for the MIB objects to be monitored.

Restrictions for Periodic MIB Data Collection and Transfer Mechanism

Cisco Data Collection MIB configuration using SNMP is not currently implemented.

For specific restrictions, see the tasks in the "How to Configure Periodic MIB Data Collection and Transfer Mechanism" section on page 4.

Information About Periodic MIB Data Collection and Transfer Mechanism

To configure the Periodic MIB Data Collection and Transfer Mechanism, you must understand the following concepts:

Note In the Cisco IOS CLI, the Periodic MIB Data Collection and Transfer Mechanism is referred to as the Bulk Statistics feature.

SNMP Objects and Instances

A type (or class) of SNMP management information is called an object. A specific instance from a type of management information is called an object instance (or SNMP variable). To configure a bulk statistics collection, you must specify the object types to be monitored using a bulk statistics object list and the specific instances of those objects to be collected using a bulk statistics schema.

MIBs, MIB tables, MIB objects, and object indices can all be specified using a series of numbers called an object identifier (OID). OIDs are used in configuring a bulk statistics collection in both the bulk statistics object lists (for general objects) and in the bulk statistics schemas (for specific object instances).

Bulk Statistics Object Lists

To group the MIB objects to be polled, you will need to create one or more object lists. A bulk statistics object list is a user-specified set of MIB objects that share the same MIB index. Object lists are identified using a name that you specify. Named bulk statistics object lists allow the same configuration to be reused in different bulk statistics schemas.

All the objects in an object list must share the same MIB index. However, the objects do not need to be in the same MIB and do not need to belong to the same MIB table. For example, it is possible to group ifInOctets and an Ethernet MIB object in the same schema, because the containing tables for both objects are indexed by the ifIndex.

Bulk Statistics Schemas

Data selection for the Periodic MIB Data Collection and Transfer Mechanism requires the definition of a schema with the following information:

- Name of an object list.
- Instance (specific or wildcarded) that needs to be retrieved for objects in above object list.
- How often the specified instances need to be sampled (polling interval).

A bulk statistics schema is also identified using a name that you specify. This name is used when configuring the transfer options.

Bulk Statistics Transfer Options

After configuring the data to be collected, a single virtual file (VFile or “bulk statistics file”) with all collected data is created. This file can be transferred to a network management station (NMS) using FTP, rcp, or TFTP. You can specify how often this file should be transferred. The default transfer interval is once every 30 minutes. You can also configure a secondary destination for the file to be used if, for whatever reason, the file cannot be transferred to the primary network management station.

The value of the transfer interval is also the collection period (collection interval) for the local bulk statistics file. After the collection period ends, the bulk statistics file is frozen, and a new local bulk statistics file is created for storing data. The frozen bulk statistics file is then transferred to the specified destination.

By default, the local bulk statistics file is deleted after successful transfer to an NMS. However, you can configure the routing device to keep the bulk statistics file in memory for a specified amount of time.

An SNMP notification (trap) can be sent to the NMS if a transfer to the primary or secondary NMS is not successful. Additionally, a syslog message will be logged on the local device if transfers are unsuccessful.

Benefits of the Periodic MIB Data Collection and Transfer Mechanism

The Periodic MIB Data Collection and Transfer Mechanism (Bulk Statistics feature) allows many of the same functions as the Bulk File MIB (CISCO-BULK-FILE-MIB.my), but offers some key advantages.

The main advantage is that this feature can be configured through the CLI and does not require an external monitoring application.

The Periodic MIB Data Collection and Transfer Mechanism is mainly targeted for medium to high-end platforms that have sufficient local storage (volatile or permanent) to store bulk statistics files. Locally storing bulk statistics files helps minimize loss of data during temporary network outages.

This feature also has more powerful data selection features than the Bulkfile MIB; it allows grouping of MIB objects from different tables into data groups (object lists). It also incorporates a more flexible instance selection mechanism, where the application is not restricted to fetching an entire MIB table.

How to Configure Periodic MIB Data Collection and Transfer Mechanism

Configuring a Bulk Statistics Object List

The first step in configuring the Periodic MIB Data Collection and Transfer Mechanism is to configure one or more object lists.

Restrictions

All the objects in a bulk statistics object list have to be indexed by the same MIB index. However, the objects in the object list do not need to belong to the same MIB or MIB table.

When specifying an object name instead of an OID (using the **add** command), only object names from the Interfaces MIB (IF-MIB.my), Cisco Committed Access Rate MIB (CISCO-CAR-MIB.my) and the MPLS Traffic Engineering MIB (MPLS-TE-MIB.my) may be used.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **snmp mib bulkstat object-list** *list-name*
4. **add** {*oid* | *object-name*}
5. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**snmp mib bulkstat object-list** *list-name* **Example:** `Router(config)# snmp mib bulkstat object-list ifMib`	Defines an SNMP bulk statistics object list and enters Bulk Statistics Object List configuration mode.

Step 4	**add** {*oid* | *object-name*}
	
	Example:
	`Router(config-bulk-objects)# add 1.3.6.1.2.1.2.2.1.11`
	`Router(config-bulk-objects)# add ifAdminStatus`
	`Router(config-bulk-objects)# add ifDescr`
	`.`
	`.`
	`.`
		Adds a MIB object to the bulk statistics object list.
		• Repeat as desired until all objects to be monitored in this list are added.

Step 5	**exit** **Example:** `Router(config-bulk-objects)# exit`	Exits from Bulk Statistics Object List configuration mode.

Configuring a Bulk Statistics Schema

The next step in configuring the Periodic MIB Data Collection and Transfer Mechanism is to configure one or more schemas.

Prerequisites

The bulk statistics object list to be used in the schema must be defined.

Restrictions

Only one object list can be associated with a schema at a time.

SUMMARY STEPS

1. **snmp mib bulkstat schema** *schema-name*
2. **object-list** *list-name*
3. **instance** {**exact** | **wild**} {**interface** *interface-id* [**sub-if**] | **controller** *controller-id* [**sub-if**] | **oid** *oid*}

4. **instance range start** *oid* **end** *oid* (optional)
5. **instance repetition** *oid-instance* **max** *repeat-number* (optional)
6. **poll-interval** *minutes*
7. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**snmp mib bulkstat schema** *schema-name* **Example:** `Router(config)# snmp mib bulkstat schema intE0`	Names the bulk statistics schema and enters Bulk Statistics Schema (config-bulk-sc) configuration mode.
Step 2	**object-list** *list-name* **Example:** `Router(config-bulk-sc)# object-list ifMib`	Specifies the bulk statistics object list to be included in this schema. Specify only one object list per schema. (If multiple **object-list** commands are executed, the earlier ones are overwritten by newer commands.)

Step 3	**instance** {**exact** | **wild**} {**interface** *interface-id* [**sub-if**] | **controller** *controller-id* [**sub-if**] | **oid** *oid*}
	
	Example:
	`Router(config-bulk-sc)# instance wild oid 1`
	or
	
	Example:
	`Router(config-bulk-sc)# instance exact interface FastEthernet 0/1 subif`
		Specifies the instance information for objects in this schema.
		• The **instance exact** command indicates that the specified instance, when appended to the object list, is the complete OID.
		• The **instance wild** command indicates that all subindices of the specified OID belong to this schema. The **wild** keyword allows you to specify a partial, "wild carded" instance.
		• Instead of specifying an instance OID, you can specify a specific interface. The **interface** *interface-id* syntax allows you to specify an interface name and number (for example, interface Ethernet 0) instead of specifying the ifIndex OID for the interface. Similarly, the **controller** *controller-id* syntax allows you to specify a controller card (interface). This option is platform dependent.
		• The optional **sub-if** keyword, when added after specifying an interface or controller, includes the ifIndexes for all subinterfaces of the interface you specified.
		• Only one **instance** command can be configured per schema. (If multiple instance commands are executed, the earlier ones are overwritten by new commands.)

Step 4	**instance range start** *oid* **end** *oid* **Example:** `instance range start 1 end 2`	(Optional) When used in conjunction with the **snmp mib bulkstat schema** command, the **instance range** command can be used to configure a range of instances on which to collect data.

	Command or Action	Purpose
Step 5	**instance repetition** *oid-instance* **max** *repeat-number* **Example:** `instance repetition 1 max 4`	(Optional) When used in conjunction with the **snmp mib bulkstat schema** command, the **instance repetition** command can be used to configure data collection to repeat for a certain number of instances of a MIB object.
Step 6	**poll-interval** *minutes* **Example:** `Router(config-bulk-sc)# poll-interval 10`	Sets how often data should be collected from the object instances specified in this schema, in minutes. The default is once every 5 minutes. The valid range is from 1 to 20000.
Step 7	**exit** **Example:** `Router(config-bulk-objects)# exit`	Exits from Bulk Statistics Schema configuration mode.

Configuring a Bulk Statistics Transfer Options

The final step in configuring the Periodic MIB Data Collection and Transfer Mechanism is to configure the transfer options. The collected MIB data are kept in a local file-like entity called a VFile (virtual file, referred to as a bulk statistics file in this document). This file can be transferred to a remote network management station (NMS) at intervals you specify.

Prerequisites

The bulk statistics object lists and bulk statistics schemas should be defined before configuring the bulk statistics transfer options.

Restrictions

Transfers can only be performed using schemaASCII (cdcSchemaASCII) format. SchemaASCII is an ASCII format that contains parser-friendly hints for parsing data values.

SUMMARY STEPS

1. **snmp mib bulkstat transfer** *transfer-id*
2. **buffer-size** *bytes* (optional)
3. **format** {**bulkBinary | bulkASCII | schemaASCII**} (optional)
4. **schema** *schema-name*
5. **transfer-interval** *minutes* (optional)
6. **url primary** *url*
7. **url secondary** *url* (optional)
8. **retry** *number* (optional)
9. **retain** *minutes* (optional)
10. **enable**

11. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**snmp mib bulkstat transfer** *transfer-id* **Example:** `Router(config)# snmp mib bulkstat transfer bulkstat1`	Identifies the transfer configuration with a name (*transfer-id*) and enters Bulk Statistics Transfer configuration mode.
Step 2	**buffer-size** *bytes* **Example:** `Router(config-bulk-tr)# buffer-size 3072`	(Optional) Specifies the maximum size for the bulk statistics data file, in bytes. The valid range is from 1024 to 2147483647 bytes. The default buffer size is 2048 bytes. **Note** A configurable buffer size limit is available only as a safety feature. Normal bulk statistics files should not generally meet or exceed the default value.
Step 3	**format** {**bulkBinary** | **bulkASCII** | **schemaASCII**} **Example:** `Router(config-bulk-tr)# format schemaASCII`	(Optional) Specifies the format of the bulk statistics data file (VFile). The default is schemaASCII. **Note** Transfers can only be performed using schemaASCII (cdcSchemaASCII) format. SchemaASCII is a human-readable format that contains parser-friendly hints for parsing data values.

Step 4	**schema** *schema-name*
	
	Example:
	`Router(config-bulk-tr)# schema ATM2/0-IFMIB`
	`Router(config-bulk-tr)# schema ATM2/0-CAR`
	`Router(config-bulk-tr)# schema Ethernet2/1-IFMIB`
	`.`
	`.`
	`.`
		Specifies the bulk statistics schema to be transferred. Repeat this command as desired. Multiple schemas can be associated with a single transfer configuration; all collected data will be in a single bulk data file (VFile).

Step 5	**transfer-interval** *minutes* **Example:** `Router(config-bulk-tr)# transfer-interval 20`	(Optional) Specifies how often the bulk statistics file should be transferred, in minutes. The default value is once every 30 minutes. The transfer interval is the same as the collection interval.
Step 6	**url primary** *url* **Example:** `Router(config-bulk-tr)# url primary ftp://user:password@host/folder/bulkstat1`	Specifies the network management system (host) that the bulk statistics data file should be transferred to, and the protocol to use for transfer. The destination is specified as a Uniform Resource Locator (URL). • FTP, rcp, or TFTP can be used for the bulk statistics file transfer.

	Command or Action	Purpose
Step 7	**url secondary** *url* **Example:** `Router(config-bulk-tr)# url secondary tftp://10.1.0.1/tftpboot/user/bulkstat1`	(Optional) Specifies a backup transfer destination and protocol for use in the event that transfer to the primary location fails. • FTP, rcp, or TFTP can be used for the bulk statistics file transfer.
Step 8	**retry** *number* **Example:** `Router(config-bulk-tr)# retry 1`	(Optional) Specifies the number of transmission retries. The default value is 0 (in other words, no retries). • If an attempt to send the bulk statistics file fails, the system can be configured to attempt to send the file again using this command. One retry includes an attempt first to the primary destination then, if the transmission fails, to the secondary location; for example, if the retry value is 1, an attempt will be made first to the primary URL, then to the secondary URL, then to the primary URL again, then to the secondary URL again. • The valid range is from 0 to 100.
Step 9	**retain** *minutes* **Example:** `Router(config-bulk-tr)# retain 60`	(Optional) Specifies how long the bulk statistics file should be kept in system memory, in minutes, after the completion of the collection interval and a transmission attempt is made. The default value is 0. • Zero (**0**) indicates that the file will be deleted immediately after a successful transfer. **Note** If the **retry** command is used, you should configure a retain interval larger than 0. The interval between retries is the retain interval divided by the retry number. For example, if **retain 10** and **retry 2** are configured, retries will be attempted once every 5 minutes. Therefore, if retain 0 is configured, no retries will be attempted. • The valid range is from 0 to 20000.
Step 10	**enable** **Example:** `Router(config-bulk-tr)# enable`	Begins the bulk statistics data collection and transfer process for this configuration. • For successful execution of this action, at least one schema with non-zero number of objects should be configured. • Periodic collection and file transfer operations will commence only if this command is configured. Conversely, the **no enable** command will stop the collection process. A subsequent **enable** will start the operations again. • Each time the collection process is started using the **enable** command, data is collected into a new bulk statistics file. When the **no enable** command is used, the transfer process for any collected data will immediately begin (in other words, the existing bulk statistics file will be transferred to the specified management station).
Step 11	**exit** **Example:** `Router(config-bulk-tr)# exit`	Exits from Bulk Statistics Transfer configuration mode.

Troubleshooting Tips

If the maximum buffer size for a bulk statistics file is reached before the transfer interval time expires, the transfer operation will still be initiated, and bulk statistics data will be collected into a new file in the system buffer. To correct this behavior, you can decrease the polling frequency, or increase the size of the bulk statistics buffer. If **retain 0** is configured, no retries will be attempted. This is because the interval between retries is the retain value divided by the retry value. For example, if **retain 10** and **retry 2** are configured, retries will be attempted once every 5 minutes. Therefore, if you configure the **retry** command, you should also configure an appropriate value for the **retain** command.

Enabling Monitoring for Bulk Statistics Collection

Optionally, you can enable SNMP notifications to be sent, which provide information on the transfer status of the Periodic MIB Data Collection and Transfer Mechanism (Bulk Statistics feature).

SUMMARY STEPS

1. **configure terminal**
2. **snmp-server community** *string* [**view** *view-name*] [**ro** | **rw**] [*acl-number*]
3. **snmp-server enable traps bulkstat** [**collection** | **transfer**]
4. **snmp-server host** *host-address* [**traps** | **informs**] [**version** {**1** | **2c** | **3** [**auth** | **noauth** | **priv**]}] *community-string* [**udp-port** *port*] [**bulkstat**]
5. **do copy running-config startup-config**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 2	**snmp-server community** *string* [**view** *view-name*] [**ro** | **rw**] [*acl-number*] **Example:** `Router(config)# snmp-server community public`	Specifies the SNMP community and access options for the device.

	Command or Action	Purpose
Step 3	**snmp-server enable traps bulkstat** [**collection** | **transfer**] **Example:** `Router(config)# snmp-server enable traps bulkstat`	Enables the sending of bulk statistics SNMP notifications (traps or informs). The following notifications (defined in the CISCO-DATA-COLLECTION-MIB) are enabled with this command: • transfer (cdcFileXferComplete)—Sent when a transfer attempt is successful and when a transfer attempt fails. (The varbind cdcFilXferStatus object in the trap defines tells if the transfer is successful or not). • collection (cdcVFileCollectionError)—Sent when data collection could not be carried out successfully.One possible reason for this condition could be insufficient memory on the device to carry out data collection.
Step 4	**snmp-server host** *host-address* [**traps** | **informs**] [**version** {**1** | **2c** | **3** [**auth** | **noauth** | **priv**]}] *community-string* [**udp-port** *port*] [**bulkstat**] **Example:** `Router(config)# snmp-server host informs public bulkstat`	Specifies the recipient (host) for the SNMP notifications, and additional transfer options.
Step 5	**do copy running-config startup-config** **Example:** `Router(config)# do copy running-config startup-config`	(Optional) Saves the current configuration to NVRAM as the startup configuration file. • The **do** command allows you to execute EXEC mode commands in any configuration mode.

Monitoring and Troubleshooting Periodic MIB Data Collection and Transfer Mechanism

The **show** command for this feature displays the status of the bulk statistics processes. The **debug** command enables the standard set of debugging messages for technical support purposes.

SUMMARY STEPS

1. **show snmp mib bulkstat transfer** [*transfer-name*]
2. **debug snmp bulkstat**

DETAILED STEPS

	Command or Action	Purpose

Step 1	**show snmp mib bulkstat transfer** [*transfer-name*]
	
	Example:
	`Router# show snmp mib bulkstat transfer`
	
	`Transfer Name : ifmib`
	` Retained files`
	
	` File Name : Time Left (in seconds)`
	`:STATE`
	` ----------`
	`----`
	` ifmib_Router_020421_100554683 : 173 : Retry (2`
	`Retry attempt(s) Left)`
	
	` ifmib_Router_020421_100554683 : 53 : Retained`
		(Optional) The **show** command for this feature lists all bulk statistics virtual files (VFiles) on the system that have finished collecting data. (Data files that are not complete are not displayed.)
		
		The output lists all of the completed local bulk statistics files, the remaining time left before the bulk statistics file is deleted (remaining retention period), and the state of the bulk statistics file.
		
		The "STATE" of the bulk statistics file will be one of the following:
		• Queued—Indicates that the data collection for this bulk statistics file is completed (in other words, the transfer interval has been met) and that the bulk statistics file is waiting for transfer to the configured destination(s).
		• Retry—Indicates that one or more transfer attempts have failed and that the file transfer will be attempted again. The number of retry attempts remaining will be displayed in parenthesis.
		• Retained—Indicates that the bulk statistics file has either been successfully transmitted or that the configured number of retries have been completed.
		
		Tip To determine if a transfer was successful, enable the bulk statistics SNMP notification.
		
		To display only the status of a named transfer (as opposed to all configured transfers), specify the name of the transfer in the *transfer-name* argument.

Step 2	**debug snmp bulkstat** **Example:** `Router# debug snmp bulkstat`	(Optional) Enables standard debugging output for the Bulk Statistics feature. Debugging output includes messages about the creation, transfer, and deletion of bulk statistics files.

Configuration Examples for Periodic MIB Data Collection and Transfer Mechanism

This section provides the following configuration example:

- Configuring Periodic MIB Data Collection and Transfer Mechanism: Example, page 12

Configuring Periodic MIB Data Collection and Transfer Mechanism: Example

This section provides a complete example of configuring the Periodic MIB Data Collection and Transfer Mechanism (Bulk Statistics feature). The example is described in the following subsections:

Transfer Parameters

The following transfer parameters are used for the "Configuring the Periodic MIB Data Collection and Transfer Mechanism" example:

- Transfer interval (collection interval)—30 minutes
- Primary URL—ftp://john:pswrd@cbin2-host/users/john/bulkstat1
- Secondary URL—tftp://john@10.1.1.1/tftpboot/john/bulkstat1
- Transfer format—schemaASCII
- Retry interval—Retry after 6 minutes (retry = 5, retain = 30; 5 retry attempts over the 30-minute retention interval.)

Polling Requirements

The following polling requirements for ATM interface 2/0 and Ethernet interface 2/1 are used for the "Configuring the Periodic MIB Data Collection and Transfer Mechanism" example:

ATM interface 2/0

- Objects to be polled—ifInOctets, ifOutOctets, ifInUcastPkts, ifInDiscards, CcarStatSwitchedPkts, CcarStatSwitchedBytes, CcarStatFilteredBytes
- Polling interval—Once every 5 minutes
- Instances—Main interface and all subinterfaces
- For CAR MIB objects, poll all instances related to the specified interface

Ethernet Interface 2/1

- Objects to be polled—ifInOctets, ifOutOctets, ifInUcastPkts, ifInDiscards, CcarStatSwitchedPkts, CcarStatSwitchedBytes, CcarStatFilteredBytes
- Polling interval—Once every 10 minutes
- Instances—Only main interface is to be monitored
- For CAR MIB objects, only include instances pertaining to packets in the incoming direction (on the main interface)

Object List Configuration

Note that since the IF-MIB objects and the CAR-MIB objects do not have the same index, they will have to be a part of different schemas. However, since the objects required are the same for the ATM interface and the Ethernet interface, the object list can be reused for each schema. Therefore, in the following example, an object list is created for the for the IF-MIB objects and another object list is created for the CAR-MIB objects.

```
snmp mib bulkstat object-list ifmib
 add ifInoctets
 add ifOutoctets
 add ifInUcastPkts
 add ifInDiscards
 exit
snmp mib bulkstat object-list CAR-mib
 add CcarStatSwitchedPkts
 add CcarStatSwitchedBytes
 add CcarStatFilteredBytes
 exit
```

Schema Definition Configuration

For the following bulk statistics schema configuration, two schemas are defined for each interface—one for the IF-MIB object instances and one for the CAR-MIB object instances.

```
! ATM IF-MIB schema
snmp mib bulkstat schema ATM2/0-IFMIB
! The following command points to the IF-MIB object list, defined above.
 object-list ifmib
 poll-interval 5
 instance exact interface ATM2/0 subif
 exit

! ATM CAR-MIB schema
snmp mib bulkstat schema-def ATM2/0-CAR
 object-list CAR-mib
 poll-interval 5
 instance wildcard interface ATM2/0 subif
 exit

!Ethernet IF-MIB schema
snmp mib bulkstat schema Ethernet2/1-IFMIB
 object-list ifmib
 poll-interval 5
 instance exact interface Ethernet2/1
 exit

! Ethernet CAR-MIB schema
snmp mib bulkstat schema Ethernet2/1-CAR
 object-list CAR-mib
 poll-interval 5
! Note: ifindex of Ethernet2/1 is 3
 instance wildcard oid 3.1
 exit
```

Transfer Parameter Configuration

For the transfer of the bulk statistics file, the transfer configuration is given the name bulkstat1. All of the four schema definitions are included in the following transfer configuration.

```
snmp mib bulkstat transfer bulkstat1
 schema ATM2/0-IFMIB
 schema ATM2/0-CAR
 schema Ethernet2/1-IFMIB
 schema Ethernet2/1-CAR
 url primary ftp://username1:pswrd@cbin2-host/users/username1/bulkstat1
 url secondary tftp://username1@10.1.0.1/tftpboot/username1/bulkstat1
 format schemaASCII
 transfer-interval 30
 retry 5
 buffer-size 1024
 retain 30
 end
copy running-config startup-config
```

Displaying Status

The following sample output for the **show snmp mib bulkstat transfer** command shows that the initial transfer attempt and the first retry has failed for the newest file, and four additional retry attempts will be made:

```
Router# show snmp mib bulkstat transfer
Transfer Name : bulkstat1

Primary URL ftp://user:XXXXXXXX@192.168.200.162/
Secondary ftp://user:XXXXXXXX@192.168.200.163/

      Retained files

      File Name              : Time Left (in seconds)     : STATE
     ------------------------------------------------------------------------
      bulkstat1_Router_030307_102519739: 1196         :Retry(4 Retry attempt(s) Left)
      bulkstat1_Router_030307_102219739: 1016         :Retained
      bulkstat1_Router_030307_101919739:  836         :Retained
```

The filename for the bulk statistics file is generated with the following extensions to the name you specify in the **url** command:

specified-filename_device-name_date_time-stamp

The device name is the name of the sending device, as specified in the CLI prompt.

The time-stamp format will depend on your system configuration. Typically, the format for the date is YYYYMMDD or YYMMDD. The time stamp uses a 24-hour clock notation, and the format is HHMMSSmmm (where mmm are milliseconds).

In the example above, the files were created on March 7, 2003, at 10:25 a.m., 10:22 a.m., and 10:19 a.m.

Bulk Statistics Output File

The following is sample output as it appears in the bulk statistics file received at the transfer destination. In this output, the name of the bulk statistics file is bulkstat1_Router_20030131_193354234. Also, note that the schema definition (Schema-def) for the schema Ethernet2/1-IFMIB was added to the file as the configuration was changed (see comment lines indicated by "!").

```
Schema-def  ATM2/0-IFMIB "%u, %s,  %u, %u, %u, %u"
epochtime ifDescr instanceoid ifInOctets ifOutOctets ifInUcastPkts ifInDiscards
Schema-def  ATM2/0-CAR  "%u, %s,  %s, %u, %u, %u, %u "
```

```
epochtime ifDescr instanceoid CcarStatSwitchedPkts  ccarStatSwitchedBytes
CcarStatSwitchedPkts  ccarStatSwitchedBytes
Schema-def  Ethernet2/1-IFMIB "%u, %u,  %u, %u, %u, %u"
epochtime ifDescr instanceoid ifInOctets ifOutOctets ifInUcastPkts ifInDiscards
Schema-def  Ethernet2/1-CAR  "%u, %s, %u, %u, %u, %u "
Epochtime instanceoid CcarStatSwitchedPkts  ccarStatSwitchedBytes CcarStatSwitchedPkts
ccarStatSwitchedBytes
Schema-def GLOBAL "%s, %s,  %s, %u, %u, %u, %u"
            hostname data timeofday sysuptime cpu5min cpu1min cpu5sec

ATM2/0-IFMIB: 954417080, ATM2/0, 2, 95678, 23456, 234, 3456
ATM2/0-IFMIB: 954417080, ATM2/0.1, 8, 95458, 54356, 245, 454
ATM2/0-IFMIB: 954417080, ATM2/0.2, 9, 45678, 8756, 934, 36756
ATM2/0-CAR: 954417083, ATM2/0, 2.1.1, 234, 345, 123, 124
ATM2/0-CAR: 954417083, ATM2/0, 2.2.1, 452, 67, 132, 145
ATM2/0-CAR: 954417083, ATM2/0.1, 8.1.1, 224, 765, 324 234
ATM2/0-CAR: 954417083, ATM2/0.1, 8.2.1, 234, 345, 123, 124
ATM2/0-CAR: 954417083, ATM2/0.2, 9.1.1, 234, 345, 123, 124
ATM2/0-CAR: 954417083, ATM2/0.2, 9.2.1, 452, 67, 132, 145
Ethernet2/1-IFMIB: 954417090, Ethernet2/1, 3, 45678, 8756, 934, 36756
Ethernet2/1-CAR: 954417093, 3.1.1, 234, 345, 123, 124
Ethernet2/1-CAR: 954417093, 3.1.2, 134, 475, 155, 187
ATM2/0-IFMIB: 954417100, ATM2/0, 2, 95678, 23456, 234, 3456
ATM2/0-IFMIB: 954417101, ATM2/0.1, 8, 95458, 54356, 245, 454
ATM2/0-IFMIB: 954417102, ATM2/0.2, 9, 45678, 8756, 934, 36756
ATM2/0-CAR: 954417106, ATM2/0, 2.1.1, 234, 345, 123, 124
ATM2/0-CAR: 954417107, ATM2/0, 2.2.1, 452, 67, 132, 145
ATM2/0-CAR: 954417107, ATM2/0.1, 8.1.1, 224, 765, 324 234
ATM2/0-CAR: 954417108, ATM2/0.1, 8.2.1, 234, 345, 123, 124
ATM2/0-CAR: 954417113, ATM2/0.2, 9.1.1, 234, 345, 123, 124
ATM2/0-CAR: 954417114, ATM2/0.2, 9.2.1, 452, 67, 132, 145
! Here the Schema-def for "Ehternet2/1-IFMIB" was changed on the originating device.
Schema-def  Ethernet2/1-IFMIB "%u, %u,  %u, %u, %u, %u"
! The object ifOutDiscards has been added to the object list for this schema.
epochtime ifDescr instanceoid ifInOctets ifOutOctets ifInUcastPkts ifInDiscards
            ifOutDiscards
! The following data sample reflects the change in the configuration.
Ethernet2/1-IFMIB: 954417090, Ethernet2/1, 3, 45678, 8756, 934, 36756, 123
Ethernet2/1-CAR: 954417093, 3.1.1, 234, 345, 123, 124
Ethernet2/1-CAR: 954417093, 3.1.2, 134, 475, 155, 187
GLOBAL: Govinda, 20020129, 115131, 78337, 783337, 2%, 0%, 62%
```

Additional References

The following sections provide references related to the Periodic MIB Data Collection and Transfer Mechanism.

Related Documents

Related Topic	Document Title
SNMP configuration tasks	"Configuring SNMP Support" module in the *Cisco IOS Network Management Configuration Guide*
SNMP commands: complete command syntax, command mode, command history, defaults, usage guidelines, and examples	*Cisco IOS Network Management Command Reference*

MIBs

MIBs	MIBs Link
This feature supports all Cisco implemented MIBs. This feature uses the Cisco Data Collection MIB (CISCO-DATA-COLLECTION-MIB.my) function of reporting errors and statistics during data collection and transfer. The Cisco Data Collection MIB also supports configuring data collection using the CLI, as well as with SNMP.	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
None	—

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/techsupport

Feature Information for Periodic MIB Data Collection and Transfer Mechanism

Not all commands may be available in your Cisco IOS software release. For release information about a specific command, see the command reference documentation.

Use Cisco Feature Navigator to find information about platform support and software image support. Cisco Feature Navigator enables you to determine which Cisco IOS, Catalyst OS, and Cisco IOS XE software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Note Table 1 lists only the Cisco IOS software release that introduced support for a given feature in a given Cisco IOS software release train. Unless noted otherwise, subsequent releases of that Cisco IOS software release train also support that feature.

Table 1 Feature Information for Periodic MIB Data Collection and Transfer Mechanism

Feature Name	Releases	Feature Information
Periodic MIB Data Collection and Transfer Mechanism	12.0(24)S 12.3(2)T 12.2(25)S 12.2(33)SRA 12.2(33)SXH 12.2(33)SRC 12.2(33)SB	This feature provides the ability to periodically transfer selected MIB data from Cisco IOS-based devices to specified Network Management Systems (NMS). Using the command-line interface (CLI), data from multiple MIBs can be grouped into lists, and a polling interval (frequency of data collection) can be configured. All the MIB objects in a list are periodically polled using this specified interval. The collected data from the lists can then be transferred to a specified NMS at a user-specified transfer interval (frequency of data transfer) using TFTP, rcp, or FTP.

VPN Device Manager for XSM

VPN Device Manager Client for Cisco IOS Software (XSM Configuration)

Feature History

Release	Modification
12.1(6)E	This feature was introduced.
12.2(9)YE, 12.2(9)YO1	This feature was integrated into Cisco IOS Release 12.2YE and 12.2YO.
12.2(13)T	This feature was integrated into Cisco IOS Release 12.2T for inclusion in Release 12.3.
12.2(14)S	This feature was integrated into Cisco IOS Release 12.2S.

This document was written for Release 12.1(6)E, and last updated January 2003 for Release 12.2(14)S.

Note For the primary documentaiton of the latest version of the VPN Device Manager (version 1.2), see the "Installation Guide and Release Notes for VPN Device Manager 1.2" at http://www.cisco.com/univercd/cc/td/doc/product/rtrmgmt/vdm/vdm12rn.htm

This document describes the command-line interface (CLI) Cisco IOS commands required to activate the VPN Device Manager (VDM) client and includes the following sections:

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

Feature Overview

VDM software is installed directly onto Cisco VPN devices. It allows network administrators to use a web browser to manage and configure site-to-site VPNs on a single device. VDM implements a wizard-based GUI that allows simplified VPN configuration of the device on which it resides and peer-to-peer interfaces from that device to remote devices. VDM requires configuration of some Cisco IOS commands before it can be fully operational.

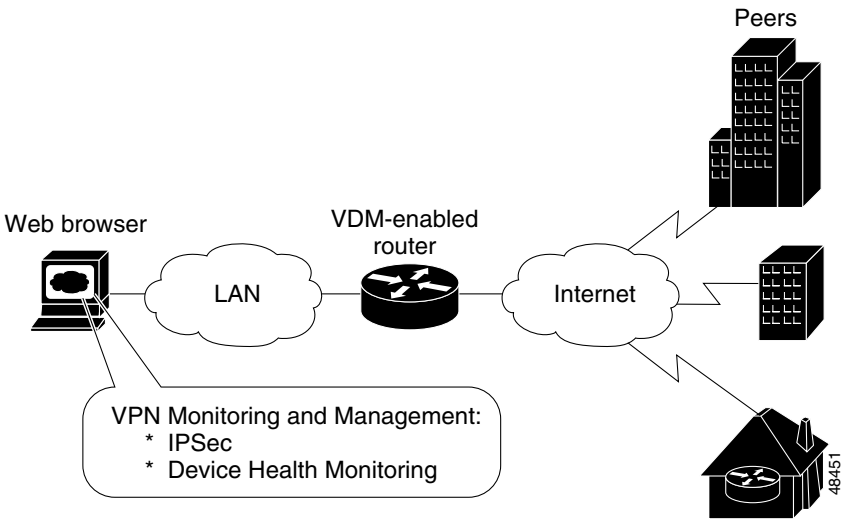
Note In addition to having the relevant Cisco IOS image installed on your device, make sure the VDM client software has been preinstalled in the device Flash memory. If it has not been, you must download it from Cisco.com. See the *Installation and Release Notes for VPN Device Manager* for the product version you are using for details on completing this task. See the VPN Device Manager index (http://www.cisco.com/warp/public/cc/pd/nemnsw/vpdvmn) for further information.

VDM also monitors general system statistics and VPN-specific information such as tunnel throughput and errors. The graphing capability allows comparison of such parameters as traffic volume, tunnel counts, and system utilization. VDM supports site-to-site VPNs. Its step-by-step wizards simplify the configuration of common VPN setups, interfaces, and policies, including:

- IPSec tunnels
- Preshared keys and Internet Key Exchange (IKE) policies

Figure 1 shows a simplified VDM deployment within a VPN.

Figure 1 Simplified VDM Deployment



XML Subscription Manager

XML Subscription Manager (XSM) is an HTTP-based service for retrieving information from a Cisco device. Once remote applications (such as VDM) are connected to the XSM server, they can subscribe to data sets called XML Request Descriptors (XRDs). These are XML-formatted messages describing configuration (access-control lists (ACLs), interfaces, crypto-maps, and others) and monitoring information (CPU, memory usage, interface statistics, and others).

XSM provides remote applications such as VDM with a constantly updated stream of data about Cisco device status by supplying real-time data without repeated device polling.

CLI Commands for VDM

This document gives details about Cisco IOS commands specific to VDM functionality. These commands are not related to general VPN functions but are designed to manage VDM itself via the XSM server. By using the Java-enabled VDM application, you can perform all VPN-related configuration and monitoring tasks within the application.

These commands are designed to complement VDM. The following tasks are performed by specific Cisco IOS XSM commands (command name in parentheses):

- Enabling VDM to receive data from the XSM feature set on the device (**xsm**)
- Enabling basic device monitoring, configuration, and data delivery for VDM (**xsm edm**)
- Enabling VPN-specific monitoring, configuration, and data delivery for VDM (**xsm vdm**)
- Enabling access to switch operations (for example, configuring switch ports and VLANs) when running VDM on a switch (**xsm dvdm**)
- Enabling collection of selected statistics generic to embedded devices on the XSM server (**xsm history edm**)
- Enabling collection of specific selected VPN statistics on the XSM server (**xsm history vdm**)
- Clearing VDM client sessions (**clear xsm**)
- Displaying information about the XSM server and VDM (**show xsm status**)
- Displaying all XRDs available to VDM (**show xsm xrd-list**)
- Setting user privilege levels for viewing VDM monitoring and configuration data (**xsm privilege monitor level** and **xsm privilege configuration level**)

For more information on VDM, the *Installation and Release Notes for VPN Device Manager* for the product version you are using or the Documentation CD-ROM that shipped with the product. See the VPN Device Manager index (http://www.cisco.com/warp/public/cc/pd/nemnsw/vpdvmn) for further information.

Related Features and Technologies

- Virtual Private Networks (VPNs)
- Security

Related Documents

- *Access VPN Solutions Using Tunneling Technology*
- *Access VPDN Dial-in Using L2TP*
- *Access VPDN Dial-in Using IPSec Over L2TP*
- Cisco IOS Dial Technologies Command Reference, Release 12.2
- Cisco IOS Security Configuration Guide, Release 12.2
- Cisco IOS Security Command Reference, Release 12.2

- "Configuring Virtual Private Networks" chapter in the Virtual Templates, Profiles, and Networks *part of the Cisco IOS Dial Technologies Configuration Guide,* Release 12.2
- Installation and Release Notes for VPN Device Manager
- VDM chapter in the *Cisco Enterprise VPN Configuration Guide*
- *VPN Device Manager*
- IPSec VPN Acceleration Services Module Installation and Configuration Note

Supported Platforms

The XSM Cisco IOS commands are available on the following VDM-enabled platforms:

- Cisco 1700 series routers
- Cisco 2600 series routers
- Cisco 3620, 3640, and 3660 routers
- Cisco 7100 series routers
- Cisco 7200 series routers
- Cisco 7400 series routers
- Cisco Catalyst 6500 series switches with IPSec VPN Acceleration Services Module installed
- Cisco 7600 series Internet routers with IPSec VPN Acceleration Services Module installed

This feature is supported on the following platforms in Cisco IOS Release 12.2(14)S:

- Cisco 7200 series
- Cisco 7400 series

Determining Platform Support Through Cisco Feature Navigator

Cisco IOS software is packaged in feature sets that support specific platforms. To get updated information regarding platform support for this feature, access Cisco Feature Navigator. Cisco Feature Navigator dynamically updates the list of supported platforms as new platform support is added for the feature.

Cisco Feature Navigator is a web-based tool that enables you to quickly determine which Cisco IOS software images support a specific set of features and which features are supported in a specific Cisco IOS image. You can search by feature or release. Under the release section, you can compare releases side by side to display both the features unique to each software release and the features in common.

To access Cisco Feature Navigator, you must have an account on Cisco.com. If you have forgotten or lost your account information, send a blank e-mail to cco-locksmith@cisco.com. An automatic check will verify that your e-mail address is registered with Cisco.com. If the check is successful, account details with a new random password will be e-mailed to you. Qualified users can establish an account on Cisco.com by following the directions at http://www.cisco.com/register.

Cisco Feature Navigator is updated regularly when major Cisco IOS software releases and technology releases occur. For the most current information, go to the Cisco Feature Navigator home page at the following URL:

http://www.cisco.com/go/fn

Availability of Cisco IOS Software Images

Platform support for particular Cisco IOS software releases is dependent on the availability of the software images for those platforms. Software images for some platforms may be deferred, delayed, or changed without prior notice. For updated information about platform support and availability of software images for each Cisco IOS software release, refer to the online release notes or, if supported, Cisco Feature Navigator.

Supported Standards, MIBs, and RFCs

Standards

No new or modified standards are supported by this feature.

MIBs

No new or modified MIBs are supported by this feature.

To obtain lists of supported MIBs by platform and Cisco IOS release, and to download MIB modules, go to the Cisco MIB website on Cisco.com at the following URL:

http://www.cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

No new or modified RFCs are supported by this feature.

Prerequisites

The VDM client software must be installed on your device. It might already have been installed if you chose the VPN option at the time of configuration.

Configuring VDM

See the following sections for configuration tasks for this feature. Each task in the list is identified as either required or optional.

- Enabling the XSM Server for VDM (required)
- Configuring XSM Privilege Levels for XRDs (optional)
- Disabling the XSM Server for VDM (optional)
- Verifying VDM Status on the XSM Server (optional)
- Clearing XSM Client Sessions (optional)
- Configuring XSM Statistics Collection (optional)

Enabling the XSM Server for VDM

Use the **xsm** command in global configuration mode to activate XSM clients (such as VDM) on your device. Enabling this command also enables the **xsm vdm** and **xsm edm** global configuration commands, so there is no need to enable them separately.

Command	Purpose
`Router(config)# xsm`	Enables XSM client access to the device.

Configuring XSM Privilege Levels for XRDs

To set the minimum required privilege levels and grant appropriate access to view, monitor, or configure the XSM client (such as VDM), use the following commands in global configuration mode. Privilege levels set on the device determine which access level users possess (configuration and monitoring, monitoring only, or neither).

Users with privilege levels lower than the required monitoring privilege level will not have access to either the configuration or monitoring data required for subscription to XML Request Descriptors (XRDs). The higher the number, the higher the privilege level. The privilege level for the **xsm privilege configuration level** command must be greater than or equal to that of the **xsm privilege monitor level** command.

Command	Purpose
`Router(config)# xsm privilege configuration level` *number*	Enables configuration privilege level to subscribe to XRDs. • *number*—Privilege level (1–15). Privilege level 15 is the default.
`Router(config)# xsm privilege monitor level` *number*	Enables monitor privilege level to subscribe to XRDs. • *number*—Privilege level (1–15). Privilege level 15 is the default.

Disabling the XSM Server for VDM

To disable the XSM server, use the command below in global configuration mode. Disabling this command also disables the **xsm vdm** and **xsm edm** global configuration commands.

Command	Purpose
`Router(config)# no xsm`	Disables XSM server.

Verifying VDM Status on the XSM Server

Use the **show xsm status** command to verify the status of clients (such as VDM) on the XSM server.

Command	Purpose
`Router# show xsm status`	Displays information and status about clients subscribed to the XSM server.

Use the **show xsm xrd-list** command to verify all XML Request Descriptors (XRDs) for XSM clients (such as VDM) made available by subscription to the XSM server.

Command	Purpose
`Router# show xsm xrd-list`	Displays all XRDs for clients subscribed to the XSM server.

Clearing XSM Client Sessions

Use the **clear xsm** command to clear data from XSM clients (such as VDM) on the XSM server. To disconnect a specific client, you must identify the session number. Use the **show xsm status** command to obtain specific session numbers.

Command	Purpose
`Router# clear xsm [session number]`	Clears XSM client sessions. • **session**—XSM session ID. • *number*—Number of the specific XSM client session you are clearing.

Configuring XSM Statistics Collection

To configure the XSM server and its related clients (such as VDM) for Embedded Device Manager (EDM) or VPN-specific statistics collection of up to 5 days of data, use the following commands in global configuration mode.

Command	Purpose
`Router(config)# xsm history edm`	Enables statistics collection for the EDM on the XSM server.
`Router(config)# xsm history vdm`	Enables specific VPN statistics collection on the XSM server.

Configuration Examples for VDM

This section provides the following configuration examples:

- Enabling the XSM Server for VDM Example
- Configuring XSM Privilege Levels for XRDs Example
- Disabling the XSM Server for VDM Example
- Configuring XSM Statistics Collection Example

Enabling the XSM Server for VDM Example

The following example shows how to enable the XSM client on the device:

```
xsm
```

Configuring XSM Privilege Levels for XRDs Example

The following example shows how to set a privilege level of 11, for subscription to XRDs:

```
xsm privilege monitor level 11
```

Disabling the XSM Server for VDM Example

The following example shows how to enable and then disable the XSM client on the device to troubleshoot VDM:

```
no xsm
xsm
```

Configuring XSM Statistics Collection Example

The following example shows how to configure the XSM server and its related clients (such as VDM) for Embedded Device Manager (EDM) or VPN-specific statistics collection of up to 5 days of data:

```
xsm history edm
xsm history vdm
```

Command Reference

The following modified commands are pertinent to this feature. To see the command pages for these commands and other commands used with this feature, go to the *Cisco IOS Master Commands List*, Release 12.4, at http://www.cisco.com/univercd/cc/td/doc/product/software/ios124/124mindx/124index.htm.

- **clear xsm**
- **crypto mib topn**
- **show xsm status**
- **show xsm xrd-list**
- **xsm**
- **xsm dvdm**
- **xsm edm**
- **xsm history edm**
- **xsm history vdm**
- **xsm privilege configuration level**
- **xsm privilege monitor level**
- **xsm vdm**

Glossary

Internet Key Exchange (IKE)—A key management protocol standard used in conjunction with IPSec and other standards. IPSec can be configured without IKE, but IKE enhances IPSec by providing additional features, flexibility, and ease of configuration for the IPSec standard. IKE authenticates the IPSec peers, negotiates IPSec keys, and negotiates IPSec security associations. Before any IPSec traffic can be passed, each router/firewall/host must be able to verify the identity of its peer. This can be done by manually entering preshared keys into both hosts or by a CA service.

IP security (IPSec)—A framework of open standards that provides data confidentiality, data integrity, and data authentication between participating peers. IPSec provides these security services at the IP layer.

Virtual Private Network (VPN)—A virtual network that uses advanced encryption and tunneling to permit organizations to establish secure, end-to-end, private network connections over public IP infrastructure networks, such as the Internet or extranets.

VPN Device Manager (VDM)—A browser-based tool for configuring and monitoring VPNs on a VPN-enabled device. VDM allows users to configure and monitor advanced VPN functionality within Cisco devices.

XML Subscription Manager (XSM)— A Cisco IOS subsystem that allows embedded device managers such as VDM to receive XML-based configuration and monitoring information for managing network devices.

XML Request Descriptor (XRD)—A specific requested type of data from XSM.

Embedded Device Manager (EDM)—An XSM adapter that publishes general network device configuration and monitoring information for device managers such as VDM.

XML-PI

XML-PI

First Published: July 11, 2008
Last Updated: November 20, 2009

The eXtensible Markup Language Programmatic Interface (XML-PI) Release 1.0 leverages the Network Configuration Protocol (NETCONF) and offers new data models that collect **show** command output down to the keyword level and running configurations without the complexity and expense of screen-scraping technologies or external XML-to-Command Line Interface (CLI) gateways. XML-PI allows you to quickly develop XML-based network management applications that remotely adapt and control the behavior of any number of network devices simultaneously. XML-PI uses an industry standard protocol that allows Cisco network devices to be managed in a more automatic and programmatic way and is CLI accessible.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the "Feature Information for XML-PI" section on page 26.

Use Cisco Feature Navigator to find information about platform support and Cisco IOS and Catalyst OS software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Contents

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

Prerequisites for XML-PI

Note Be sure you have enough lines configured for the network devices you will be collecting command output from. XML-PI requires that you configure at least two vty lines per NETCONF session to handle the formatting.

- You must be familiar with NETCONF and the *Programmer's Guide for Cisco Enhanced Device Interface 2.2*.
- You must be familiar with RFC 4741, *NETCONF Configuration Protocol* and RFC 4742, *Using the NETCONF Configuration Protocol over Secure SHell (SSH)*.
- NETCONF and Secure Shell Version 2 (SSHv2) are both required to run XML-PI. SSHv2 is the only transport protocol supported for XML-PI Release 1.0. Together, NETCONF and SSHv2 terminate the session layer and provide a secure connection. See the *Network Configuration Protocol* document for additional prerequisites and information about NETCONF and SSHv2.

Restrictions for XML-PI

XML-PI Supported Only on Crypto Image Files

Use of NETCONF and SSHv2 with XML-PI functionality is supported only on Cisco IOS crypto reformation images, such as IPBASEK9. Use Cisco Feature Navigator to find information about platform and software support for Cisco IOS crypto security images; see the "Finding Feature Information" section on page 1 in this document for more information about Feature Navigator.

Spec Files Must Be Local

Spec files (described in the "ODM Tool and Spec Files" section) must reside locally on the network device. Using spec files from a remote filesystem is not supported.

XML-PI and NETCONF

There are two ways XML-PI can deliver XML output from **show** commands: using either NETCONF or via the Cisco CLI from the console. In cases where non-CLI access to XML-PI is desirable, only NETCONF can be used to retrieve **show** command output.

Configuration changes using XML-PI can only be done using NETCONF. XML cannot be directly entered on the console CLI.

The Cisco IOS running configuration can be retrieved from the console by executing the **show running-config | format** command, in addition to being available via NETCONF.

Syntax Check is Not Supported

The <edit-config> operation may not work correctly.

Invalid XML Response with <get-config> Operation

The <get-config> operation with the config-format-xml filter returns missing or wrong closing tags for <X-Interface>, as shown in the following examples:

```
<LineVty0-Configuration>
    <X-Interface> password cisco<X-Interface> <X-Interface> transport input
    all<X-Interface> </LineVty0-Configuration>
```

XML Tag for Parameters Is Not Interpreted Correctly

The <edit-config> operation with a merge or create containing an invalid XML tag for parameters is not interpreted correctly. You must be sure to enter the string with proper capitalization.

In the following example, the router hostname becomes “systemnetworkname” (text in bold for purpose of example):

```
<?xml version="1.0"?>

<rpc message-id="7" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <xml-config-data>
        <Device-Configuration>
        <hostname>
          <systemnetworkname operation="create">XmlpiRouter</systemnetworkname>
        </hostname>
        </Device-Configuration>
      </xml-config-data>
    </config>
  </edit-config>
</rpc>
```

In the following example, the router hostname becomes “XmlpiRouter” because the “Systemnetworkname” string was entered correctly with an initial capital letter:

```
<?xml version="1.0"?>

<rpc message-id="7" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <xml-config-data>
        <Device-Configuration>
        <hostname>
          <Systemnetworkname operation="create">XmlpiRouter</Systemnetworkname>
        </hostname>
        </Device-Configuration>
      </xml-config-data>
    </config>
  </edit-config>
</rpc>
```

Information About XML-PI

This section contains the following information about XML-PI:

XML-PI Overview

XML-PI Release 1.0 offers new NETCONF data models that collect **show** command output down to the keyword level and running configurations without the complexity and expense of screen-scraping technologies or external XML-to-CLI gateways. XML-PI allows the native conversion of Cisco IOS **show** command output into tagged XML and provides the associated schema definition. The resulting output is in a consistent, unambiguous format that is easily interpreted. Additional tools allow the output format to be customized for individual user requirements.

The following XML-PI Release 1.0 capabilities will help you quickly develop XML-based network management applications:

- Execute selected **show** commands and retrieve the output in well-formed XML.

  Use a **format** modifier that feeds the **show** command output through an XML converter.
- Retrieve the XML Schema Definition (XSD) for selected **show** commands.

  Execute the **show xsd-format** command to display the XSD to which the XML output conforms.
- Execute the **show format** command to display a list of commands with a *spec file entry* (SFE) in the spec file, display the XML format SFE for a specific command, or validate a spec file. For more information on spec files and SFEs, see the "ODM Tool and Spec Files" section on page 8
- Retrieve the running configuration in XML.

  XML-PI Release 1.0 provides native XML output for the **show running-config** command.
- Change the running configuration on a network device by sending an XML fragment of a configuration change.
- Quickly adapt capabilities of XML-PI using fully formed sample applications.

  You can use a built-in file containing definitions for the most commonly used **show** commands to get started on application development immediately.

The commands and output files are associated with NETCONF using the **netconf format** global configuration command. Commands are also available to help you see XML tag hierarchy, list the **show** commands that have been converted, and debug output.

NETCONF Overview

The following sections summarize the NETCONF operations:

- Enhancement for Retrieving show Commands, page 7

NETCONF Enhancements

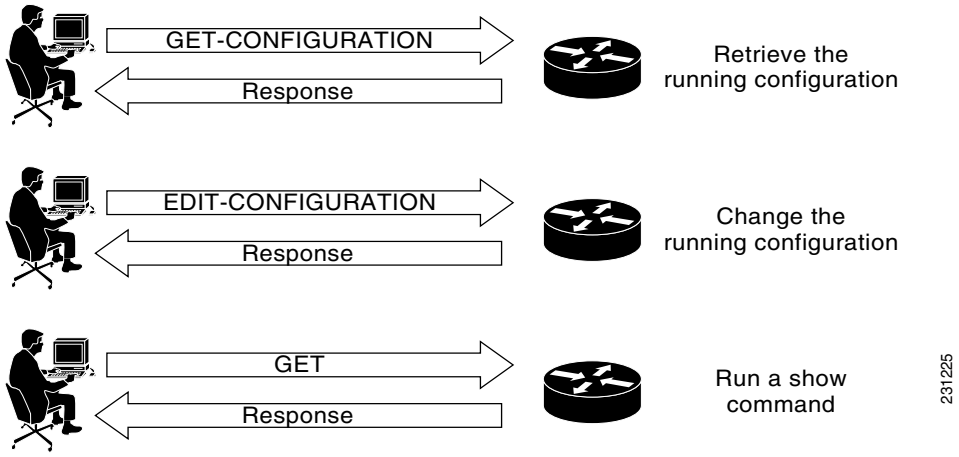
XML-PI is integrated as a data model for NETCONF, which builds on top of the industry standard protocol that allows Cisco network devices to be managed in a more automatic and programmatic way.

In XML-PI, each command keyword, parameter, and submode change is wrapped in XML tokens, which are generated based on, respectively, the keyword, help, and submode strings.

Figure 1 shows the key enhancements to the get-config, edit-config, and get operations, which are entered as <get-config>, <edit-config>, and <get> strings respectively in the enhanced device interface for XML-PI Release 1.0.

The following sections summarize these enhancements. Refer to the *Programmer's Guide for Cisco Enhanced Device Interface 2.2* for more information.

Figure 1 XML-PI Release 1.0 Key Features



Enhancement to Retrieve show running-config Output

The following subtree is added to the <get-config> operation to allow XML output for the **show running-config** to be retrieved using NETCONF:

```
<get-config>
    <source><running/></source>
    <filter type="cli"><config-format-xml options".."></config-format-xml></filter>
</get-config>
```

The NETCONF <get-config> operation with the filter containing the string <config-format-xml> in the request expects a response in XML-PI format. Only the running configuration is supported. Following is an example:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="4" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <get-config>
  <source><running/></source>
  <filter type="cli"><config-format-xml options="all"></config-format-xml></filter>
 </get-config>
</rpc>]]>]]>
```

Enhancement to Change the Running Configuration

The following subtree is added to the Config node to allow the running configuration to be changed using NETCONF:

```
<xml-config-data>  …entire subtree with C2X encoded payload </xml-config-data>
```

XML-PI configuration mode is allowed using the NETCONF <edit-config> operation only. The mode is identified when the config-format-xml XML tag is seen in an <edit-config> operation. The response is standard NETCONF success or fail. The configuration carried in the <edit-config> operation is converted to CLI using the X2C algorithm. All standard NETCONF options such as syntax check and rollback-on-error are supported. If the CLI generated from XML causes an error, an operation failed message is sent back to the request originator.

The ability for a NETCONF <edit-config> operation to accept XML-PI formatted requests is not related to the spec files. The understanding of the XML-PI configuration format is built into Cisco IOS and is an algorithmic conversion, so it cannot be modified dynamically like the spec files for the **show** commands.

A partial configuration as a subset of the full device configuration can be sent to the network device provided that the partial configuration unambiguously maps to a CLI configuration. The partial configuration must have context information such as interface or other submode information, if required, and must support rollback if the configuration cannot be applied.

Note Rollback is supported only when “archive” is configured on the network device, which is a Cisco IOS requirement.

Adding Two IP Hosts: Example

The following is an example of using the <edit-config> operation to modify the running configuration by adding two IP hosts:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="2"  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target><running/></target>
    <config>
      <xml-config-data>
        <Device-Configuration>
          <ip>
            <host>
              <NameHost>host1</NameHost>
              <HostIPAddress>10.2.3.4</HostIPAddress>
            </host>
          </ip>
          <ip>
            <host>
              <NameHost>host2</NameHost>
              <HostIPAddress>10.2.3.5</HostIPAddress>
            </host>
          </ip>
        </Device-Configuration>
      </xml-config-data>
    </config>
  </edit-config>
</rpc>]]>]]>
```

Deleting Two IP Hosts: Example

The following is an example of using the <edit-config> operation to modify the running configuration by deleting two IP hosts:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="3"  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target><running/></target>
    <config>
      <xml-config-data>
        <Device-Configuration>
          <ip>
            <host operation="delete">
              <NameHost>host1</NameHost>
              <HostIPAddress>10.2.3.4</HostIPAddress>
            </host>
          </ip>
          <ip>
             <host operation="delete">
              <NameHost>host2</NameHost>
              <HostIPAddress>10.2.3.5</HostIPAddress>
             </host>
           </ip>
        </Device-Configuration>
      </xml-config-data>
    </config>
  </edit-config>
</rpc>]]>]]>
```

<edit-config> Response

The reply to the <edit-config> operation is either the standard ok or an rpc-error.

Enhancement for Retrieving show Commands

NETCONF for retrieving **show** commands has the ability to collect command output down to the keyword level. The following subtree is added under the <get> operation:

```
    <filter type="cli">
      <config-format-text-block><text-filter-spec>| inc
netconf</text-filter-spec></config-format-text-block>
      <oper-data-format-xml><show xsd="true">…</show><show>…</show></oper-data-format-xml>
    </filter>
```

<get> Response

The reply to the <get> operation generates the following response:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="XXXX" xmlns="urn:ietf:params:netconf:base:1.0">
  <data>
    <cli-config-data-xml>… config gets embedded here …</cli-config-data-xml>
    <cli-oper-data-xml>
      <item>
        <show>…</show>
        <xsd>  … xsd text gets embedded here … </xsd>
      </item>
      …multiple items …
    </cli-oper-data-xml>
  </data>
</rpc-reply>]]>]]>
```

ODM Tool and Spec Files

The Operational Data Model (ODM) tool developed by Cisco Enhanced Device Interface (E-DI) provides an interface for creating a new ODM *spec file* from a CLI data file, for a particular **show** command. Spec files are defined by an E-DI metalanguage and contain a pattern-matching algorithm that collects output from Cisco IOS EXEC **show** commands and places it into a specific schema. The output of each **show** command is associated with an ODM spec file.

The spec file represents spatial information to extract or parse data and structural information to model the data. A benefit of using spec files is that different format descriptions can be embedded in them, thereby making the task of customizing applications easy.

The spec file can contain many individual command definitions stored as an SFE. Each SFE is delimited by a line containing three pound signs (###). The lines immediately following the ### delimiter contain the name of the command to convert. Following the command name line is spec file data, which must begin with an XML header, for example <?xml version="1.0" encoding="UTF-8"?> . The ### is both a start and stop delimiter unless the end of file (EOF) string is encountered, as shown in the following sample format:

```
###
show ip arp
<?xml version="1.0" encoding="UTF-8"?>
 … the spec conversion for ip arp
###
show ip interface brief
<?xml version="1.0" encoding="UTF-8"?>
… the spec conversion for show ip interface brief
###
show interfaces *
show another cli
<?xml version="1.0" encoding="UTF-8"?>
… The spec conversion for ip interface
```

A wildcard character (*) can be used to match command names, and uses the following search order: Find an exact match or, if not an exact match, use the wildcard character to match the maximum number of characters. Table 1 provides examples of how the wildcard character can be used in the spec file to match command names.

Table 1 Wildcard Character Command Name Matching

String	Example of Characters Matched
show interfaces	Matches "show interfaces"
show interfaces s*	Matches "show interfaces summary"
show interfaces *	Matches "show interfaces FastEthernet 0/0"

You can change the spec filename, and you can modify and customize the SFE to specific interpretation formats. If the contents of the SFE do not comply with the spec file format and language, the conversion is not loaded and no interpretation of data occurs. An error message stating the SFE is uninterpretable is generated. The format of the error message depends on the source of the request to access the spec file. NETCONF requests return a Remote Procedure Call (RPC) get rpc-reply with an error condition; CLI-based requests return get error messages on the console. Limited format debug capability is provided by the **debug format all** command. Each SFE is treated independently, and a badly formatted SFE does not affect any other SFE in the file.

You can use the **show format** command to display a list of commands with an SFE in the spec file, display the XML format SFE for a specific command, or validate a spec file.

Note Sample spec files are available for most commonly used Cisco IOS **show** commands and can be downloaded from Cisco.com. You can use the sample files "as is" or modify them for your application.

XML-CLI Conversion Algorithms

The X2C and C2X conversion algorithms are used to convert XML into CLI and CLI into XML, respectively. There are no schema used with these algorithms. The following sections provide more information about these algorithms:

- X2C Algorithm, page 9
- C2X Algorithm, page 9

X2C Algorithm

The X2C algorithm builds a Document Object Model (DOM) tree from XML. Each node in the tree can be classified as one of three node types, depending on its name, as follows:

- KEYWORD_NODE—The tag name starts with a lowercase letter or an underscore. [a...z, _]. The underscore is used to prefix any numeric value that is a keyword.
- SUBMODE_NODE—The tag name ends with -Configuration.
- PARAM_NODE—Any other nonzero length string.

The X2C algorithm then decodes a DOM tree by recursively descending the tree. In the following example, this_node is used to track the current DOM node and this_cmd is the CLI string being built:

```
decode_node(this_node)
    if (this_node is KEYWORD_NODE) {
        if  (this_node has attribute isNegation) {
          prepend "no" to this_cmd
        }
        convert this_node name to be a keyword.
        Add keyword to end of this_cmd
        iterate children of this_node through decode_node.
    } else if (this_node is PARAM_NODE) {
        add the node body data to this_cmd
    } else if (this_node is SUBMODE_NODE) {
      this_cmd is finalised and reset to ""
      iterate children of this_node through decode_node.
    }
}
```

C2X Algorithm

For the C2X algorithm, each CLI word is categorized as one of the three node types, the same as described in the "X2C Algorithm" section on page 9. The Cisco IOS CLI parser is used to generate the running configuration of the network device. As each line is generated, each word in the line is parsed through and, depending upon whether the parser encounters a KEYWORD_NODE or a PARAM_NODE, the appropriate XML tag conversion is made. If traversing through to the next line causes a SUBMODE_NODE change, the submode XML wrapper is entered or closed depending on whether the mode is entered or exited.

The C2X algorithm converts Cisco IOS CLI into XML based on keywords and parameters. CLI keywords become XML tags and parameters become the bodies of tags whose names are made by parsing the CLI help strings.

The following example is the CLI view of an **interface** command:

```
interface GigabitEthernet0/1
 ip address 10.4.0.13 255.0.0.0
 duplex auto
 speed auto
 media-type rj45
 no negotiation auto
```

The following example shows the C2X equivalent:

```
<Device-Configuration>
  <interface>
    <Param>GigabitEthernet0/1</Param>
    <ConfigIf-Configuration>
      <ip>
        <address>
          <IPAddress>10.4.0.13</IPAddress>
          <IPSubnetMask>255.0.0.0</IPSubnetMask>
        </address>
      </ip>
      <duplex><auto/></duplex>
      <speed><auto/></speed>
      <media-type><rj45/></media-type>
      <negotiation operation="delete" ><auto/></negotiation>
    </ConfigIf-Configuration>
  </interface>
</Device-Configuration>
```

How to Use XML-PI

This section contains the following procedures:

- Configuring NETCONF for XML-PI, page 10 (required)
- Generating XML Format for Commands, page 13 (required)
- Generating XSD Format for Commands, page 14 (required)
- Troubleshooting ODM Errors, page 14 (optional)
- Managing Files, page 15 (recommended)

Configuring NETCONF for XML-PI

Perform this required task to configure a secure login environment and define the file to use for XML-formatted requests.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto key generate rsa**

4. Enter the Rivest, Shamir, and Adelman (RSA) key modulus, when prompted.
5. **ip ssh timeout** *seconds*
6. **ip ssh authentication-retries** *integer*
7. **ip ssh version 2**
8. **line vty** *starting-line-number ending-line-number*
9. **login local**
10. **transport input ssh**
11. **exit**
12. **username** *name* **privilege** *level* **password** *secret*
13. **format global** *location*:*local-filename*
14. **netconf ssh**
15. **end**

DETAILED

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**crypto key generate rsa** **Example:** `Router(config)# crypto key generate rsa`	Generates RSA key pairs. **Note** If the crypto key has already been generated, the response of the command will be: % You already have RSA keys defined named *xxxx-nnn*.cisco.com. % Do you really want to replace them? [yes/no]: In most cases the reply is "no" because the crypto key has been previously generated and is stored on the NETCONF agent side. Reply "yes" if you need to reset the crypto key on the NETCONF agent side.

Step 4	Enter the RSA key modulus, when prompted.
	
	Example:
	`How many bits in the modulus [512]: 1024`
	`% Generating 1024 bit RSA keys ...[OK]`
		Prompts for the RSA key modulus when not supplied as part of the command.
		• The key modulus size must be in the range from 360 to 2048 for general purpose keys. The configuration for XML-PI requires a minimum key modulus size of 768.
		Note The system may require a few minutes to react to a key modulus greater than 512.

	Command or Action	Purpose
Step 5	**ip ssh timeout** *seconds* **Example:** `Router(config)# ip ssh timeout 60`	(Optional) Configures the time interval that the network device waits for the SSH client to respond.
Step 6	**ip ssh authentication-retries** *integer* **Example:** `Router(config)# ip ssh authentication-retries 3`	(Optional) Configures the number of attempts after which the interface is reset.
Step 7	**ip ssh version 2** **Example:** `Router(config)# ip ssh version 2`	(Optional) Configures the network device to run only SSH Version 2.
Step 8	**line vty** *starting-line-number ending-line-number* **Example:** `Router(config)# line vty 0 8`	Enters line configuration collection mode and configures a range of virtual terminal lines for remote console access. **Note** You must configure a range of lines large enough to handle two vty lines per NETCONF session.
Step 9	**login local** **Example:** `Router(config-line)# login local`	(Optional) Enables and selects local password checking. • Authentication is based on the username specified with the **username** global configuration command.
Step 10	**transport input ssh** **Example:** `Router(config-line)# transport input ssh`	(Optional) Specifies that the SSH protocol be used for line connection.
Step 11	**exit** **Example:** `Router(config-line)# exit`	Exits the current configuration mode and returns to the next highest mode.
Step 12	**username** *name* **privilege** *level* **password** *secret* **Example:** `Router(config)# username me privilege 15 password mypassword`	(Optional) Establishes a username-based authentication system. • **privilege**—Sets the privilege level, a number from 0 to 15. • **password**—Sets the password, which can contain from 1 to 25 characters and embedded spaces, and must be the last option specified in the **username** command.
Step 13	**format global** *location*:*local-filename* **Example:** `Router(config)# format global disk2:spec3.3.odm`	(Recommended) Specifies a default ODM spec file to use for XML-formatted requests.

	Command or Action	Purpose
Step 14	**netconf ssh** **Example:** `Router(config)# netconf ssh`	Enables NETCONF over SSHv2.
Step 15	**end** **Example:** `Router(config)# end`	Ends the current configuration session.

Generating XML Format for Commands

To convert Cisco IOS **show** command output into XML format, XML-PI provides the **format** output modifier to the **show** command output. This section describes how to use this modifier. For examples of command output, see the "Generating show Command XML Format: Examples" section on page 19 and the "Generating show running-config XML Format: Examples" section on page 20.

Note The **show running-config** command output is generated natively in XML, so the spec filename could be an empty file. If a default spec file has been defined with the **format global** command, no filename is required.

SUMMARY STEPS

Choose the command in Step 1 or Step 2.

1. **show-command | format** [*location*:*local-filename*]

   or

2. **show running-config** {**all | brief | full | interface** *interface-name*} **| format** [*filename*]

DETAILED STEPS

Step 1 **show-command | format** [*location*:*local-filename*]

This command executes the **show** command then redirects the output into the **format** function that will generate XML based on the specified spec file or, if no spec file is specified, the default spec file defined by the **format global** configuration command. Command names can be truncated. The *location*:*local-filename* arguments and keyword are the location and filename of the ODM spec file. Valid locations are **bootflash:**, **flash:**, **nvram:**, and any valid disk or slot number (for example: **disk0:** or **slot1:**). ODM spec files have a .odm suffix. The following is a sample command that uses the default ODM file to generate XML:

```
Router# show arp | format slot0:spec3.3.odm
```

Step 2 **show running-config** {**all | brief | full | interface** *interface-name*} **| format** [*filename*]

If you are generating output for the **show running-config** command, you can supply the following keywords and arguments with this command:

- **all**—Configuration with defaults (default when no keywords are specified with the **show running-config** command).

- **brief**—Configuration without certificate data.
- **full**—Full configuration.
- **interface** *interface-name*—Specified interface output only. A full interface specification (**interface fastethernet0/0**, for example) is required. If the interface name does not match one that is supported on the network device, an error is returned.

The following is a sample command:

```
Router# show running-config brief | format
```

Generating XSD Format for Commands

The **show xsd-format** command is used to display the XSD to which the XML output conforms. This section describes how to use this command. For example of command output, see "Generating show Command XSD Format: Example" section on page 21.

SUMMARY STEPS

1. **show xsd-format** [*location*:*local-filename*] **cli** *command*

DETAILED STEPS

Step 1 **show xsd-format** [*location*:*local-filename*] **cli** *command*

The *location* and *local-filename* arguments are the location and filename of the ODM spec file Valid *location* keywords are **bootflash:**, **flash:**, **nvram:**, and any valid disk or slot number (for example: **disk0:** or **slot1:**). ODM spec files have a .odm suffix. These arguments are not required if you want to use a default ODM file defined with the **format global** command.

The first of the following two examples, displays XSD output from a defined default ODM spec file:

```
Router# show xsd-format cli show arp

Router# show xsd-format disk2:spec3.3.odm cli show arp
```

Note When the user is entering command names, the full command name must be entered; do not use command truncation.

Troubleshooting ODM Errors

This section describes use of the **debug format all** command to troubleshoot spec file errors.

SUMMARY STEPS

1. **enable**
2. **debug format all**
3. **show-command | format** [*location*:*local-filename*]

4. **no debug format all**

DETAILED STEPS

Step 1 **enable**

Enter this command to enable the privileged EXEC mode required to run **debug** commands.

Step 2 **debug format all**

Enter this command to enable a verbose debugging mode that displays all ODM errors.

Step 3 **show-command | format** [*location*:*local-filename*]

Enter this command to generate XML output for the **show interfaces** command. The following is sample output:

```
Router# show interfaces | format slot0:spec3.3.odm
```

Selected debug data is displayed with comments followed by the full debug output.

The debug format statements are read in groups of two lines. As the following example shows, the first line describes what the attempted match was; the second line provides the offset and the byte count from the beginning of the **show interfaces** command output that the cursor of the screen scraper is currently at:

```
*May  4 01:20:35.279: ODM: Could not match Property mcast
*May  4 01:20:35.279: offset 703: 5 minute output rate 0 bits/sec, 0 packets/sec
```

The following example shows where the SFE caused the ODM algorithm to return a truncated XML. Notice how the offset jumps from 703 to 3001. This is a large jump that implies a search between multicast and IP multicast probably caused the screen scraper to jump too far into the text. Because the cursor is not at a buffer, this condition is the likely candidate for the error. Looking at the spec file entry and doing a manual search through the **show** command output will confirm this suspicion.

```
*May  4 01:20:35.279: offset 703: 5 minute output rate 0 bits/sec, 0 packets/sec
     786 pa
*May  4 01:20:35.279: ODM: Could not match Property mcast
*May  4 01:20:35.279: offset 703: 5 minute output rate 0 bits/sec, 0 packets/sec
     786 pa
*May  4 01:20:35.279: ODM: Could not match Property IP multicasts
*May  4 01:20:35.279: offset 3001: no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0
*May  4 01:20:35.279: ODM: Could not match Property watchdog
*May  4 01:20:35.279: offset 3122: ignored, 0 abort
     0 packets output, 0 bytes, 0 underru
*May  4 01:20:35.279: ODM: Could not match Property input packets with dribble condition
detected
```

Step 4 **no debug format all**

Disable the **debug** command when troubleshooting is complete.

Managing Files

This section provides the following procedures for managing files in XML-PI:

- Displaying Files on a Cisco IOS Filesystem: Example, page 16

Displaying Files on a Cisco IOS Filesystem: Example

The following example shows how to display a list of files:

```
Router# show format slot0:?

slot0:spec3.3.odm      slot0:spec3.ALR.odm      slot0:spec3.empty.odm
```

Note The question mark (**?**) command can be used following any of the *location* keywords (**bootflash**, **slot**, and so on) in the **show format** and **show xsd-format** commands, to list all files. Spec files have a .odm file extension.

Managing Spec Files

Use the **spec-file install** privileged EXEC command to manage the spec files. The following commands allow you to make backup copies of the built-in spec file before changing the contents of the file, and to restore the contents of a previous spec file. You can also copy and remove SFEs from one spec file to another.

Valid locations for local files are **bootflash:**, **flash:**, **nvram:**, and any valid disk or slot number (example: **disk0:** or **slot1:**).

Valid URLs for remote files are **archive:**, **bootflash:**, **cns:**, **flash:**, **ftp:**, **http:**, **null:**, **nvram:**, **pram:**, **rcp:**, **scp:**, **system:**, **tar:**, **tftp:**, **tmpsys:** and any valid disk or slot number (for example, **disk0:** or **slot1:**).

In all cases, the **force** keyword performs the command without prompting you to verify the file operation by entering a “yes” or “no” response.

SUMMARY STEPS

1. **spec-file install** [**force**] *location*:*local-filename* **add-entry** *url*:*remote-filename command*
2. **spec-file install** [**force**] *location*:*local-filename* **built-in**
3. **spec-file install** [**force**] *location*:*local-filename* **file** *url*:*remote-filename*
4. **spec-file install** [**force**] *location*:*local-filename* **remove-entry** *command*
5. **spec-file install** [**force**] *location*:*local-filename* **restore**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**spec-file install** [**force**] *location*:*local-filename* **add-entry** *url*:*remote-filename command* **Example:** `Router# spec-file install slot0:spec_file.odm add-entry tftp://system1/user1/show_arp.odm show arp`	Copies an SFE from a remote location and adds it to a local spec file. • A check is performed on the loaded SFE to ensure that the command is not already present in the spec file, and that the SFE can be parsed correctly in XML. • If the spec file does not exist, you will be prompted before the file is created. • If the command SFE already exists in the spec file, you will be prompted before the command SFE is replaced. • A backup copy of the local spec file is created before the remote SFE is added.
Step 2	**spec-file install** [**force**] *location*:*local-filename* **built-in** **Example:** `Router# spec-file install slot0:spec_file.odm built-in`	Replaces the current spec file with the built-in spec file. • You will be prompted before the current file is replaced and *filename*.bak will be created.
Step 3	**spec-file install** [**force**] *location*:*local-filename* **file** *url*:*remote-filename* **Example:** `Router# spec-file install slot0:spec_file.odm file tftp://system1/user1/spec_file.odm`	Replaces a local spec file with a remote spec file. • A check of the loaded file is performed to ensure that each specified command is included only once, and that the SFE can be parsed correctly in XML.
Step 4	**spec-file install** [**force**] *location*:*local-filename* **remove-entry** *command* **Example:** `Router# spec-file install slot0:spec_file.odm remove-entry show arp`	Removes an SFE from a spec file. • A check is performed to ensure that the command SFE is present in the spec file. • If the spec file does not exist, this command fails. • A backup copy of the spec file is created before the SFE is removed.
Step 5	**spec-file install** [**force**] *location*:*local-filename* **restore** **Example:** `Router# spec-file install slot0:spec_file.odm restore`	Restores a spec file to its original contents using a backup (.bak) file. • If the .bak file does not exist, this command fails.

Validating Spec Files

This section describes use of the **show format** command to validate a spec file.

The **show format built-in validate** form of the command is used to validate the built-in spec file. The **show format** *location*:*local-filename* **validate** form of the command is used to validate a specific spec file.

Restrictions

Spec files must reside locally on the network device. Using spec files from a remote filesystem is not supported.

SUMMARY STEPS

1. **enable**
2. **show format** [**built-in** | *location*:*local-filename*] [**cli** *command* | **validate**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show format** [**built-in** | *location*:*local-filename*] [**cli** *command* | **validate**] **Example:** `Router# show format built-in validate`	Validates the built-in spec file.

Configuration Examples for XML-PI

This section provides the following configuration examples:

Configuring NETCONF for XML-PI: Example

The following example shows how to configure a secure login environment. Cisco recommends that you define a default ODM file to be used for all requests using the **format global** command. You can associate that file with NETCONF for all XML-formatted requests using the **netconf format** command. If no file is specified, the built-in spec file is used for all requests. See the **format global** and **netconf format** command reference pages for more information. The **netconf ssh** configuration command enables NETCONF over SSHv2, which terminates the session layer and provides a secure connection.

```
ip domain-name cisco.com
crypto key generate rsa
ip ssh timeout 60
```

```
ip ssh authentication-retries 3
ip ssh version 2
line vty 0 8
  login local
  transport input ssh
  exit
username me privilege 15 password mypassword
format global disk2:spec3.3.odm
netconf format disk2:spec3.3.odm
netconf ssh
end
```

Generating show Command XML Format: Examples

The following examples show how to generate XML format of standard Cisco IOS **show** command output.

Standard show Command Output

Following is an example of the Cisco IOS **show arp** command output:

```
Router# show arp

Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  10.1.1.1               67   0001.42df.59e2  ARPA   FastEthernet0/0
Internet  10.3.1.2                8   0002.55c6.19a0  ARPA   FastEthernet0/0
Internet  10.4.0.5                -   000b.60dc.9408  ARPA   FastEthernet0/0
```

Generating XML

Following is an example of generating XML output of the **show arp** command from a default ODM file:

```
Router# show arp | format

<?xml version="1.0" encoding="UTF-8"?>
  <ShowArp xmlns="ODM://disk0:/spec.odm//show_arp">
    <ARPTable>
      <entry>
        <Protocol>Internet</Protocol>
        <Address>10.1.1.1</Address>
        <Age>67</Age>
        <MAC>0001.42df.59e2</MAC>
        <Type>ARPA</Type>
        <Interface>FastEthernet0/0</Interface>
      </entry>
      <entry>
        <Protocol>Internet</Protocol>
        <Address>10.3.1.2</Address>
        <Age>8 </Age>
        <MAC>0002.55c6.19a0</MAC>
        <Type>ARPA</Type>
        <Interface>FastEthernet0/0</Interface>
      </entry>
      <entry>
        <Protocol>Internet</Protocol>
        <Address>10.4.0.5</Address>
        <MAC>000b.60dc.9408</MAC>
        <Type>ARPA</Type>
        <Interface>FastEthernet0/0</Interface>
      </entry>
    </ARPTable>
  </ShowArp>
```

Generating show running-config XML Format: Examples

The following examples show the mapping between actual **show running-config** command output and the XSD format generated by piping the output through the spec3.3.odm spec file. (For sake of brevity, output from each command has been truncated.)

show running-config Command

```
Router# show running-config

Building configuration...

Current configuration : 1190 bytes
!
upgrade fpd auto
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
service internal
!
hostname Router1
!
boot-start-marker
boot system flash:c7200-js-mz.123-5.9.T
boot-end-marker
!
logging message-counter syslog
enable password secret
!
no aaa new-model
ip cef
!
no ip domain lookup
ip domain name cisco.com
ip host host1 10.66.152.11
ip host host2 10.2.2.2
multilink bundle-name authenticated
.
.
.
```

Piped Output to Generate XML

```
Router# show running-config | format

Building configuration...

<Device-Configuration>
<upgrade><fpd><auto/></fpd></upgrade>
<version><Param>12.4</Param></version>
<service><timestamps><debug><datetime><msec/></datetime></debug></timestamps></>
<service><timestamps><log><datetime><msec/></datetime></log></timestamps></serv>
<service operation="delete" ><password-encryption/></service>
<service><internal/></service>
<hostname><SystemNetworkName>Router1</SystemNetworkName></hostname>
<boot-start-marker></boot-start-marker>
<boot><system><TFTPFileNameURL>flash:c7200-js-mz.123-5.9.T</TFTPFileNameURL></s>
<boot-end-marker></boot-end-marker>
<logging><message-counter><syslog/></message-counter></logging>
<enable><password><UnencryptedEnablePassword>secret</UnencryptedEnablePassword><>
<aaa operation="delete" ><new-model/></aaa>
<ip><cef/></ip>
```

```
<ip operation="delete" ><domain><lookup/></domain></ip>
<ip><domain><name><DefaultDomainName>cisco.com</DefaultDomainName></name></doma>
<ip><host><NameHost>host1 </NameHost><HostIPAddress>10.66.152.11</HostIPAddre>
<ip><host><NameHost>host2 </NameHost><HostIPAddress>10.2.2.2</HostIPAddress></ho>
<multilink><bundle-name><authenticated/></bundle-name></multilink>
.
.
.
```

The returned data is the requested configuration converted using the C2X algorithm.

Generating show Command XSD Format: Example

The following example shows how to generate XSD for the **show arp** command:

```
Router# show xsd-format disk2:spec3.3.odm cli show arp

<?xml version="1.0"?>
  <xsd:schema elementFormDefault="qualified" attributeFormDefault="unqualified"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:complexType name="ShowArp_def">
      <xsd:sequence>
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
          <xsd:element ref="Info"/>
          <xsd:element name="ARPTable" minOccurs="0">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="entry" minOccurs="0" maxOccurs="unbounded">
                  <xsd:complexType>
                    <xsd:sequence>
                      <xsd:element name="Protocol" minOccurs="0" type="string" />
                      <xsd:element name="Address" minOccurs="0" type="string" />
                      <xsd:element name="Age" minOccurs="0" type="integer" />
                      <xsd:element name="MAC" minOccurs="0" type="string" />
                      <xsd:element name="Type" minOccurs="0" type="string" />
                      <xsd:element name="Interface" minOccurs="0" type="string" />
                    </xsd:sequence>
                  </xsd:complexType>
                </xsd:element>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
        </xsd:choice>
      </xsd:sequence>
    </xsd:complexType>
    <xsd:element name="Info" type="xsd:string"/>
    <xsd:element name="ShowArp" type="ShowArp_def"/>
  </xsd:schema>
```

Displaying the SFEs: Example

The following example shows how to display the SFE for the **show arp** command:

```
Router# show format disk2:spec3.3.odm cli show arp

<?xml version="1.0" encoding="UTF-8"?>
<ODMSpec>
  <Command>
    <Name>show arp</Name>
  </Command>
```

```
  <OS>ios</OS>
  <DataModel>
    <Container name="ShowArp" >
      <Table name="ARPTable">
        <Header name = "Protocol" start = "0" end = "10" type = "String"/>
        <Header name = "Address" start = "10" end = "26" type = "IpAddress"/>
        <Header name = "Age (min)" alias = "Age" start = "26" end = "36" type =
"Integer"/>
        <Header name = "Hardware Addr" alias="MAC" start = "36" end = "53" type =
"String"/>
        <Header name = "Type" start = "53" end = "59" type = "String"/>
        <Header name = "Interface" start = "59" end = "-1" nullable = "true" type =
"String"/>
      </Table>
    </Container>
  </DataModel>
</ODMSpec>
```

The following example shows a list of fully expanded command names that have spec files in the default ODM file:

```
Router# show format

The following CLI are supported in slot0:spec3.3.odm
show arp
show cdp neighbors detail
show context
show flash:
show interfaces*
show inventory
show ip interface brief
show ip nat translations
show line value
show line
show processes cpu
show processes memory
show region
show spanning-tree
show stacks
show vlans
```

Displaying Spec File Tag Hierarchy: Example

The **show odm-format** command displays the spec file structure in a fixed output that you can refer to in order to understand the spec file tag hierarchy. The following example shows the fixed output from the **show odm-format** command. Refer to the *Programmer's Guide for Cisco Enhanced Device Interface 2.2* for more information about the ODM tool and tag hierarchy.

```
Router# show odm format

New Name Space ''
<NotARealTag> Either 0 or 1 allowed
  <ODMSpec> Exactly 1 required
    <Command> Exactly 1 required
      <Name> Exactly 1 required
      <AliasSet> Either 0 or 1 allowed
        <Alias> At least 1 required
    <OS> Either 0 or 1 allowed
    <DataModel> Exactly 1 required
      <Container> Exactly 1 required
        <Table> 0 or more is allowed
```

```
  <Header> At least 1 required
    <Option> 0 or more is allowed
  <EndOfTheTable> Either 0 or 1 allowed
<Property> 0 or more is allowed
  <Option> 0 or more is allowed
<Container> 0 or more is allowed
  <Table> 0 or more is allowed
    <Header> At least 1 required
      <Option> 0 or more is allowed
    <EndOfTheTable> Either 0 or 1 allowed
  <Property> 0 or more is allowed
    <Option> 0 or more is allowed
     <Container> 0 or more is allowed
  <Legends> 0 or more is allowed
    <Legend> At least 1 required
  <IgnorableLinesList> 0 or more is allowed
    <Line> At least 1 required
<Legends> 0 or more is allowed
  <Legend> At least 1 required
<IgnorableLinesList> 0 or more is allowed
  <Line> At least 1 required
```

Validating a Spec File: Example

The following example shows how to validate a built-in spec file:

```
Router# show format built-in validate

The file built-in has been validated
```

Additional References

The following sections provide references related to using XML-PI.

Related Documents

Related Topic	Document Title
Cisco IOS commands	*Cisco IOS Master Commands List, All Releases*
Cisco IOS network management commands	*Cisco IOS Network Management Command Reference*
NETCONF	*Network Configuration Protocol*
ODM tool	*Programmer's Guide for Cisco Enhanced Device Interface 2.2*

Standards

Standard	Title
XML-PI based on NETCONF standards	• *User Guide for Cisco Enhanced Device Interface 2.2* • *Programmer's Guide for Cisco Enhanced Device Interface 2.2*

MIBs

MIB	MIBs Link
None	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
RFC 4741	*NETCONF Configuration Protocol*
RFC 4742	*Using the NETCONF Configuration Protocol over Secure SHell (SSH)*

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for XML-PI

Table 2 lists the release history for this feature.

Not all commands may be available in your Cisco IOS software release. For release information about a specific command, see the command reference documentation.

Cisco IOS software images are specific to a Cisco IOS software release, a feature set, and a platform. Use Cisco Feature Navigator to find information about platform support and software image support. Cisco Feature Navigator enables you to determine which Cisco IOS and Catalyst OS software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Note Table 2 lists only the Cisco IOS software release that introduced support for a given feature in a given Cisco IOS software release train. Unless noted otherwise, subsequent releases of that Cisco IOS software release train also support that feature.

Table 2 Feature Information for XML-PI

Feature Name	Releases	Feature Information
XML-PI	12.4(20)T 12.2(33)SRE	The eXtensible Markup Language Programmatic Interface (XML-PI) Release 1.0 leverages the Network Configuration Protocol (NETCONF) and offers new data models that collect **show** command output down to the keyword level and running configurations without the complexity and expense of screen-scraping technologies or external XML-to-CLI gateways. XML-PI allows you to quickly develop XML-based network management applications. The following commands were introduced or modified by this feature: **debug format**, **format global**, **netconf format**, **show format**, **show odm-format**, **show xsd-format**, **spec-file install add-entry**, **spec-file install built-in**, **spec-file install file**, **spec-file install remove-entry**, and **spec-file install restore**. This feature was integrated into Cisco IOS Release 12.2(33)SRE. The following command was introduced or modified by this feature: **show format**.

Glossary

C2X—CLI to XML.

CLI—command-line interface. An interface that allows the user to interact with the operating system by entering commands and optional arguments.

E-DI—Enhanced Device Interface.

NETCONF—Network Configuration Protocol.

ODM—Operational Data Model.

RSA—Rivest, Shamir, and Adelman, the inventors of the technique. Public-key cryptographic system that can be used for encryption and authentication.

SSH—Secure Shell.

X2C—XML to CLI.

XML—eXtensible Markup Language.

XSD—XML Schema Definition.