



Writing Embedded Event Manager Policies Using the Cisco IOS CLI

First Published: October 31, 2005
Last Updated: February 28, 2007

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 30.

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

- [Prerequisites for Writing EEM Policies Using the Cisco IOS CLI](#), page 2
- [Information About Writing EEM Policies Using the Cisco IOS CLI](#), page 2
- [How to Write EEM Policies Using the Cisco IOS CLI](#), page 11
- [Configuration Examples for Writing EEM Policies Using the Cisco IOS CLI](#), page 23



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

© 2006–2007 Cisco Systems, Inc. All rights reserved.

- [Where to Go Next, page 29](#)
- [Additional References, page 29](#)
- [Feature Information for Writing EEM Policies Using the Cisco IOS CLI, page 30](#)

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 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, page 2](#)
- [EEM Event Detectors Available by Cisco IOS Release, page 3](#)
- [EEM Actions Available by Cisco IOS Release, page 4](#)
- [Embedded Event Manager Built-In Environment Variables Used in EEM Applets, page 5](#)

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.

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	12.0(26)S 12.3(4)T	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
Application-Specific	—	Yes	Yes	Yes	Yes
CLI	—	—	Yes	Yes	Yes
Counter	—	Yes	Yes	Yes	Yes
Enhanced Object Tracking	—	—	—	Yes	—

Table 1 Availability of Event Detectors by Cisco IOS Release (continued)

Event Detector	12.0(26)S 12.3(4)T	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
GOLD	—	—	—	—	Yes
Interface Counter	—	Yes	Yes	Yes	Yes
None	—	—	Yes	Yes	Yes
OIR	—	—	Yes	Yes	Yes
Resource	—	—	—	Yes	Yes
RF	—	—	—	Yes	Yes
SNMP	Yes	Yes	Yes	Yes	Yes
Syslog	Yes	Yes	Yes	Yes	Yes
System Manager	—	—	—	—	Yes
Timer	—	Yes	Yes	Yes	Yes
IOSWDSysMon (Cisco IOS watchdog)	—	Yes	Yes	Yes	Yes
WDSysMon (Cisco IOS Software Modularity watchdog)	—	—	—	—	Yes

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	12.0(26)S 12.3(4)T	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
Execute a CLI command	—	—	Yes	Yes	Yes
Generate a CNS event	Yes	Yes	Yes	Yes	Yes
Set or modify a named counter	—	Yes	Yes	Yes	Yes
Switch to a secondary RP	Yes	Yes	Yes	Yes	Yes
Request system information	—	—	Yes	Yes	Yes
Send a short e-mail	—	—	Yes	Yes	Yes
Manually run an EEM policy	—	—	Yes	Yes	Yes

Table 2 Availability of Actions by Cisco IOS Release (continued)

Action	12.0(26)S 12.3(4)T	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
Publish an application-specific event	—	Yes	Yes	Yes	Yes
Reload the Cisco IOS software	Yes	Yes	Yes	Yes	Yes
Generate an SNMP trap	—	Yes	Yes	Yes	Yes
Generate a prioritized syslog message	Yes	Yes	Yes	Yes	Yes
Read the state of a tracked object	—	—	—	Yes	—
Set the state of a tracked object	—	—	—	Yes	—

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 3](#) lists the Cisco built-in environment variables that are read-only alphabetically by event detector and subevent.

Table 3 EEM Built-in Environment Variables (Read Only)

Environment Variable	Description
All Events	
_event_pub_time	The time at which the event type was published.
_event_type_string	The event type that triggered 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.

Table 3 EEM Built-in Environment Variables (Read Only) (continued)

Environment Variable	Description
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	
_gold_card	The card on which a GOLD failure event was detected.
_gold_sub_card	The subcard on which a GOLD failure event was detected.
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.
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.

Table 3 EEM Built-in Environment Variables (Read Only) (continued)

Environment Variable	Description
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.
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 SNMP object ID delta value when the event was published.
_snmp_oid_val	The SNMP object ID value when the event was published.
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.

Table 3 EEM Built-in Environment Variables (Read Only) (continued)

Environment Variable	Description
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.
_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_subx_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.

Table 3 *EEM Built-in Environment Variables (Read Only) (continued)*

Environment Variable	Description
_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.
_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 3 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_subx_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_subx_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, page 11](#)
- [Registering and Defining an Embedded Event Manager Policy to Run Manually, page 14](#)
- [Unregistering Embedded Event Manager Policies, page 15](#)
- [Suspending Embedded Event Manager Policy Execution, page 17](#)
- [Configuring and Tracking a Stub Object Using Embedded Event Manager, page 18](#)
- [Displaying Embedded Event Manager History Data, page 21](#)
- [Displaying Embedded Event Manager Registered Policies, page 22](#)

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 23.

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 4](#) describes the e-mail-specific environment variables that can be used in EEM policies.

Table 4 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.	mailserver.example.com
_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 syslog** [**priority** *priority-level*] **msg** *msg-text*
9. **action label mail server** *server-address* **to** *to-address* **from** *from-address* [**cc** *cc-address*] **subject** *subject* **body** *body-text*
10. Add more action commands as required.
11. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	enable Example: Router> enable	Enables privileged EXEC mode. <ul style="list-style-type: none">• Enter your password if prompted.
Step 2	show event manager environment [all <i>variable-name</i>] Example: Router# show event manager environment all	(Optional) Displays the name and value of EEM environment variables. <ul style="list-style-type: none">• The optional all keyword displays all the EEM environment variables.• The optional <i>variable-name</i> argument displays information about the specified environment variable.
Step 3	configure terminal Example: Router# configure terminal	Enters global configuration mode.

	Command or Action	Purpose
Step 4	<pre>event manager environment variable-name string</pre> <p>Example: Router(config)# event manager environment _email_to engineering@example.com </p>	<p>Configures the value of the specified EEM environment variable.</p> <ul style="list-style-type: none"> 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	<pre>event manager applet applet-name</pre> <p>Example: Router(config)# event manager applet memory-fail </p>	Registers the applet with the Embedded Event Manager (EEM) and enters applet configuration mode.
Step 7	<pre>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</pre> <p>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 </p>	<p>Specifies the event criteria that cause the EEM applet to run.</p> <ul style="list-style-type: none"> 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 8	<pre>action label syslog [priority priority-level] msg msg-text</pre> <p>Example: Router(config-applet)# action 1.0 syslog priority critical msg "Memory exhausted; current available memory is \$_snmp_oid_val bytes" </p>	<p>Specifies the action to be taken when an EEM applet is triggered.</p> <ul style="list-style-type: none"> 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 <i>priority-level</i> argument must be defined. The <i>msg-text</i> argument can be character text, an environment variable, or a combination of the two.
Step 9	<pre>action label mail server server-address to to-address from from-address [cc cc-address] subject subject body body-text</pre> <p>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" </p>	<p>Specifies the action of sending a short e-mail when an EEM applet is triggered.</p> <ul style="list-style-type: none"> The <i>server-address</i> argument specifies the fully qualified domain name of the e-mail server to be used to forward the e-mail. The <i>to-address</i> argument specifies the e-mail address where the e-mail is to be sent. The <i>from-address</i> argument specifies the e-mail address from which the e-mail is sent. The <i>subject</i> argument specifies the subject line content of the e-mail as an alphanumeric string. The <i>body-text</i> argument specifies the text content of the e-mail as an alphanumeric string.

	Command or Action	Purpose
Step 10	Add more action commands as required.	—
Step 11	<code>end</code>	Exits applet configuration mode and returns to privileged EXEC mode.
	Example: <code>Router(config-applet)# end</code>	

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 27.

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*
6. **end**
7. **event manager run** *applet-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	<code>enable</code> Example: Router> enable	Enables privileged EXEC mode. <ul style="list-style-type: none"> Enter your password if prompted.
Step 2	<code>configure terminal</code> Example: Router# configure terminal	Enters global configuration mode.
Step 3	<code>event manager applet <i>applet-name</i></code> Example: Router(config)# event manager applet manual-policy	Registers the applet with the Embedded Event Manager and enters applet configuration mode.
Step 4	<code>event none</code> 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	<code>action <i>label</i> syslog [<i>priority</i> <i>priority-level</i>]</code> <code>msg <i>msg-text</i></code> Example: Router(config-applet)# action 1.0 syslog msg "Manual-policy triggered"	Specifies the action to be taken when an EEM applet is triggered. <ul style="list-style-type: none"> 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 <i>priority-level</i> argument must be defined. The <i>msg-text</i> argument can be character text, an environment variable, or a combination of the two.
Step 6	<code>end</code> Example: Router(config-applet)# end	Exits applet configuration mode and returns to privileged EXEC mode.
Step 7	<code>event manager run <i>applet-name</i></code> 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** [*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. <ul style="list-style-type: none"> • Enter your password if prompted.
Step 2	show event manager policy registered [<i>event-type event-name</i>] [system user] [time-ordered name-ordered] Example: Router# show event manager policy registered	(Optional) Displays the EEM policies that are currently registered. <ul style="list-style-type: none"> • 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 <i>policy-filename</i> Example: Router(config)# no event manager policy IPSLAping1	Removes the EEM policy from the configuration, causing the policy to be unregistered.
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 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 [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	<code>enable</code> Example: Router> enable	Enables privileged EXEC mode. <ul style="list-style-type: none"> Enter your password if prompted.
Step 2	<code>show event manager policy registered</code> [<i>event-type event-name</i>] [<i>system user</i>] [<i>time-ordered name-ordered</i>] Example: Router# show event manager policy registered	(Optional) Displays the EEM policies that are currently registered. <ul style="list-style-type: none"> 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	<code>configure terminal</code> Example: Router# configure terminal	Enters global configuration mode.
Step 4	<code>event manager scheduler suspend</code> Example: Router(config)# event manager scheduler suspend	Immediately suspends the execution of all EEM policies.
Step 5	<code>exit</code> 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. <ul style="list-style-type: none">• Enter your password if prompted.
Step 2	configure terminal Example: Router# configure terminal	Enters global configuration mode.
Step 3	track <i>object-number</i> stub-object Example: Router(config)# track 2 stub-object	Creates a stub object to be tracked using EEM and enters tracking configuration mode. <ul style="list-style-type: none">• Use the <i>object-number</i> argument to assign a number to the tracked object.
Step 4	default-state {up down} Example: Router(config-track)# default-state up	Sets the default state for a stub object. <ul style="list-style-type: none">• In this example, the default state of the object is set to up.

	Command or Action	Purpose
Step 5	<code>exit</code> Example: Router(config-track)# exit	Exits tracking configuration mode and returns to global configuration mode.
Step 6	<code>event manager applet applet-name</code> Example: Router(config)# event manager applet track-two	Registers an applet with EEM and enters applet configuration mode.
Step 7	<code>event [label] track object-number [state {up down any}]</code> Example: Router(config-applet)# event track 2 state down	Specifies the event criteria that cause the EEM applet to run. <ul style="list-style-type: none"> 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	<code>action label track set object-number state {up down}</code> Example: Router(config-applet)# action 1.0 track set 2 state up	Specifies the action to be taken when an EEM applet is triggered. <ul style="list-style-type: none"> In this example, the action to be taken is to set the state of tracked object number 2 to up.
Step 9	<code>action label track read object-number</code> Example: Router(config-applet)# action 2.0 track read 2	Specifies the action to be taken when an EEM applet is triggered. <ul style="list-style-type: none"> In this example, the action to be taken is to read the state of tracked object number 2. The <code>_track_state</code> read-only variable gets set when this command is run.
Step 10	<code>end</code> Example: Router(config-applet)# end	Exits applet configuration mode and returns to privileged EXEC mode.
Step 11	<code>show track [object-number [brief]]</code> Example: Router# show track 2	(Optional) Displays information about objects that are tracked by the tracking process. <ul style="list-style-type: none"> The optional <code>object-number</code> 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"

```

Configuration Examples for Writing EEM Policies Using the Cisco IOS CLI

This section contains the following configuration examples:

- [Embedded Event Manager Applet Configuration: Examples, page 23](#)
- [Embedded Event Manager Manual Policy Execution: Examples, page 27](#)
- [Configuring and Tracking a Stub Object Using Embedded Event Manager: Example, page 27](#)
- [Embedded Event Manager Watchdog System Monitor \(Cisco IOS\) Event Detector Configuration: Example, page 28](#)

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 11](#).

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
```

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"
```

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"
```

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.

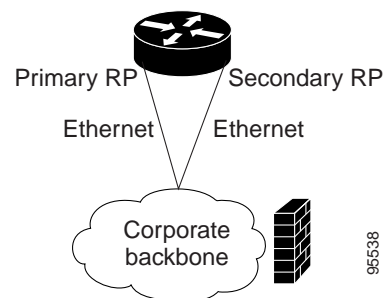
```
event manager applet snmpcpu75
  event snmp oid 1.3.6.1.4.1.9.9.109.1.1.1.3.1 get-type exact entry-op ge entry-val 75
  poll-interval 10
  action 1.0 cli command "enable"
  action 2.0 cli command "show process cpu"
  action 3.0 mail server "192.168.1.146" to "engineer@cisco.com" from "devtest@cisco.com"
  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
```

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"
```

Where to Go Next

- For information about writing EEM policies using Tcl, see the “[Writing Embedded Event Manager Policies Using Tcl](#)” module.
- For more details about other network management technologies, see the *Cisco IOS Network Management Configuration Guide*, Release 12.4.

Additional References

The following sections provide references related to writing EEM policies Using the Cisco IOS CLI.

Related Documents

Related Topic	Document Title
Software Modularity commands (including EEM commands): complete command syntax, defaults, command mode, command history, usage guidelines, and examples	<i>Cisco IOS Software Modularity Command Reference</i> , Release 12.2(18)SXF4
Network Management commands (including EEM commands): complete command syntax, defaults, command mode, command history, usage guidelines, and examples	<ul style="list-style-type: none"> • <i>Cisco IOS Network Management Command Reference</i>, Release 12.4T • <i>Cisco IOS Network Management Command Reference</i>, Release 12.2SB • <i>Cisco IOS Network Management Command Reference</i>, Release 12.2SR
Embedded Event Manager technical overview	“ Embedded Event Manager Overview ” module
Embedded Event Manager policy writing using Tcl	“ Writing Embedded Event Manager Policies Using Tcl ” module
Configuration of other network management technologies	<i>Cisco IOS Network Management Configuration Guide</i> , Release 12.4T.
Embedded Resource Manager	“ Embedded Resource Manager ” module
Configuring enhanced object tracking	“ Configuring Enhanced Object Tracking ” module
CNS event agent	<i>CNS Event Agent</i> feature document, Release 12.2(2)T
CNS configuration engine	<i>Cisco CNS Configuration Registrar: Installing and Configuring the IE2100</i>

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 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 Writing EEM Policies Using the Cisco IOS CLI

[Table 5](#) 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, 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 5](#) 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 5 *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	<p>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.</p> <p>The following sections provide information about this feature:</p> <ul style="list-style-type: none"> • Embedded Event Manager Policies, page 2 • EEM Event Detectors Available by Cisco IOS Release, page 3 • EEM Actions Available by Cisco IOS Release, page 4 • Registering and Defining an Embedded Event Manager Applet, page 11 • Displaying Embedded Event Manager Registered Policies, page 22 <p>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.</p>

Table 5 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	<p>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.</p> <p>The following sections provide information about this feature:</p> <ul style="list-style-type: none"> • Embedded Event Manager Policies, page 2 • EEM Event Detectors Available by Cisco IOS Release, page 3 • EEM Actions Available by Cisco IOS Release, page 4 • Registering and Defining an Embedded Event Manager Policy to Run Manually, page 14 • Unregistering Embedded Event Manager Policies, page 15 • Suspending Embedded Event Manager Policy Execution, page 17 • Displaying Embedded Event Manager History Data, page 21 • Embedded Event Manager Applet Configuration: Examples, page 23 • Embedded Event Manager Watchdog System Monitor (Cisco IOS) Event Detector Configuration: Example, page 28 <p>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.</p>

Table 5 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	<p>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).</p> <p>The following sections provide information about this feature:</p> <ul style="list-style-type: none"> • Embedded Event Manager Policies, page 2 • EEM Event Detectors Available by Cisco IOS Release, page 3 • EEM Actions Available by Cisco IOS Release, page 4 • Registering and Defining an Embedded Event Manager Policy to Run Manually, page 14 • Embedded Event Manager Applet Configuration: Examples, page 23 <p>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.</p>

Table 5 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	<p>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.</p> <p>The following sections provide information about this, feature:</p> <ul style="list-style-type: none"> • Embedded Event Manager Policies, page 2 • EEM Event Detectors Available by Cisco IOS Release, page 3 • EEM Actions Available by Cisco IOS Release, page 4 • Embedded Event Manager Applet Configuration: Examples, page 23 <p>The following commands were introduced by this feature: event gold, event process, show event manager metric process.</p> <p>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.</p>
Embedded Event Manager 2.2	12.4(2)T 12.2(31)SB3 12.2(33)SRB	<p>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.</p> <p>The following sections provide information about this feature:</p> <ul style="list-style-type: none"> • Embedded Event Manager Policies, page 2 • EEM Event Detectors Available by Cisco IOS Release, page 3 • EEM Actions Available by Cisco IOS Release, page 4 • Configuring and Tracking a Stub Object Using Embedded Event Manager, page 18 • Configuring and Tracking a Stub Object Using Embedded Event Manager: Example, page 27 <p>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.</p>
SNMP event detector delta environment variable ¹	12.4(11)T	A new SNMP event detector environment variable, <code>__snmp_old_delta_val</code> , was introduced.

1. This is a minor enhancement. Minor enhancements are not typically listed in Feature Navigator.

CCVP, the Cisco logo, and Welcome to the Human Network are trademarks of Cisco Systems, Inc.; Changing the Way We Work, Live, Play, and Learn is a service mark of Cisco Systems, Inc.; and Access Registrar, Aironet, Catalyst, CCDA, CCDP, CCIE, CCIP, CCNA, CCNP, CCSP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Cisco Unity, Enterprise/Solver, EtherChannel, EtherFast, EtherSwitch, Fast Step, Follow Me Browsing, FormShare, GigaDrive, HomeLink, Internet Quotient, IOS, iPhone, IP/TV, iQ Expertise, the iQ logo, iQ Net Readiness Scorecard, iQuick Study, LightStream, Linksys, MeetingPlace, MGX, Networkers, Networking Academy, Network Registrar, PIX, ProConnect, ScriptShare, SMARTnet, StackWise, The Fastest Way to Increase Your Internet Quotient, and TransPath are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

All other trademarks mentioned in this document or Website are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (0711R)

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

© 2006–2007 Cisco Systems, Inc. All rights reserved.

