Embedded Event Manager 2.2

Embedded Event Manager (EEM) is a distributed, scalable, 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 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.

EEM 2.2 introduces some new event detectors and actions, including enhanced object tracking.

History for the Embedded Event Manager 2.2 Feature

Release	Modification
12.0(26)S	This feature was introduced.
12.3(4)T	Additional functionality was added, and this feature was integrated into Cisco IOS Release 12.3(4)T.
12.3(2)XE	This feature was integrated into Cisco IOS Release 12.3(2)XE.
12.2(25)S	Additional functionality was added, and this feature was integrated into Cisco IOS Release 12.2(25)S.
12.3(14)T	Additional functionality was added.
12.4(2)T	Additional functionality was added.

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

CISCO SYSTEMS

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

Prerequisites for Embedded Event Manager 2.2

- 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 Embedded Event Manager 2.2

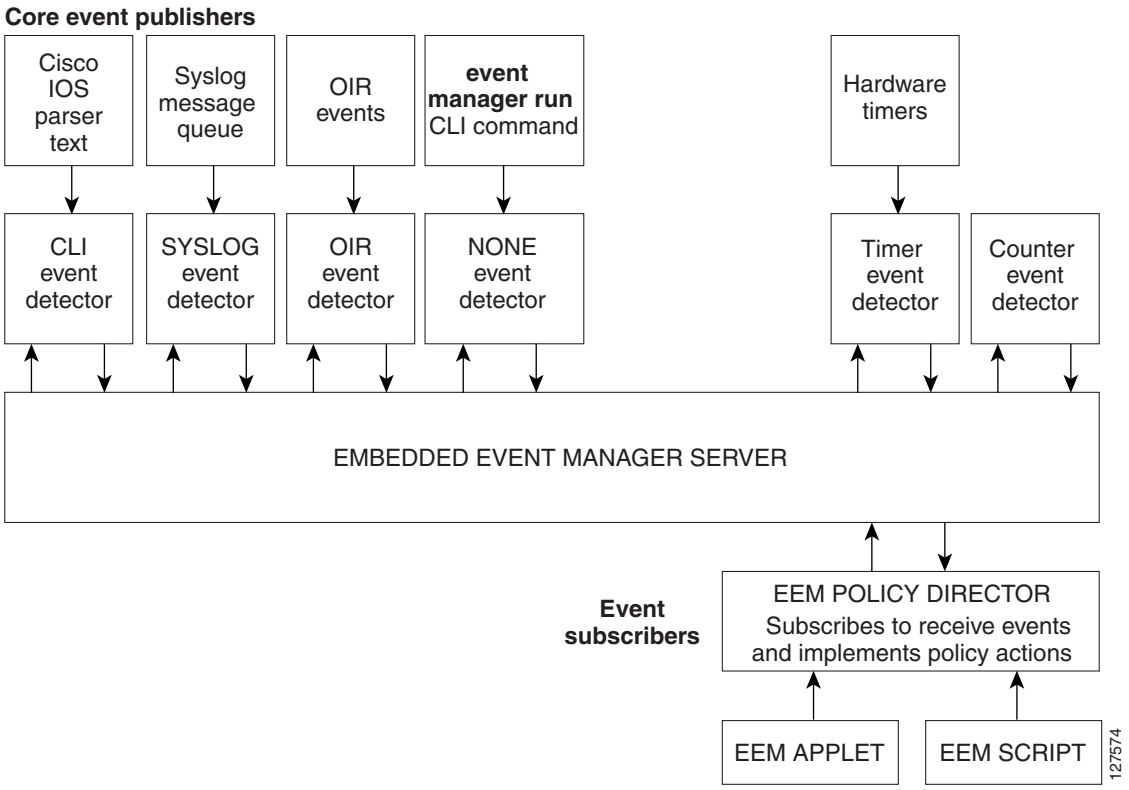
To configure Embedded Event Manager 2.2, you should understand the following concepts:

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 based 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 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.

Figure 1 Embedded Event Manager Core Event Detectors



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 Tcl.

Embedded Event Manager 2.2

EEM 2.2 is supported in Cisco IOS Release 12.4(2)T and introduces some new features. EEM 2.2 introduces the following new event detectors:

- Enhanced Object Tracking—The enhanced object tracking event detector publishes an event when the 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.

  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, Virtual Router Redundancy Protocol (VRRP), or Gateway Load Balancing Protocol (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.

- 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 introduces the following actions:

- Reading the state of a tracked object.
- Setting the state of a tracked object.

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. EEM 2.2 contains the following event detectors.

Application-Specific Event Detector

The application-specific event detector allows any Embedded Event Manager policy to publish an event.

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.
- 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.

Enhanced Object Tracking Event Detector

The enhanced object tracking 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.

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.

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 the 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. For more details about ERM, see the "Embedded Resource Manager" chapter of the *Cisco IOS Network Management Configuration Guide*, Release 12.4.

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).

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.

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.

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 Event Detector

The Cisco IOS watchdog system monitor event detector publishes an event when one of the following occurs:

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

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.

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. EEM 2.2 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.

Embedded Event Manager Environment Variables

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 system-defined—Defined by Cisco and can be read only or read/write. The read only variables are set by the system when a policy 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 1) and Cisco system-defined environment variables (see Table 2) 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 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.

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 1 describes the Cisco-defined variables used in the sample EEM policies.

Table 1 Cisco-Defined Environmental Variables and Examples

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**

Table 1 Cisco-Defined Environmental Variables and Examples (continued)

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.swpkg.cisco.internal.com/fm/interface_tm.cgi
_cron_entry	A CRON specification that determines when the policy will run. See the *Writing Embedded Event Manager Policies* document 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.customer.com
_email_to	The address to which e-mail is sent.	engineering@customer.com
_email_from	The address from which e-mail is sent.	devtest@customer.com
_email_cc	The address to which the e-mail must be copied.	manager@customer.com
_show_cmd	The CLI 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.*FastEthernet0/0.*

Table 2 describes the Cisco system-defined environment variables that are read only.

Table 2 Cisco System-Defined 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.

Table 2 Cisco System-Defined Environment Variables (Read Only) (continued)

Environment Variable	Description
_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.
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; 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.

Table 2 Cisco System-Defined Environment Variables (Read Only) (continued)

Environment Variable	Description
_resource_time_sent	The ERM event time, in nanoseconds.
_resource_user_id	The ERM resource user ID.
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_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.
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 (Cisco IOS) 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 (Cisco IOS) 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_util or mem_used.
Watchdog System Monitor (Cisco IOS) cpu_util Events	
_ioswd_sub1_path _ioswd_sub2_path	The 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	A 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.

Table 2 Cisco System-Defined Environment Variables (Read Only) (continued)

Environment Variable	Description
Watchdog System Monitor (Cisco IOS) mem_used Events	
_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	A process name of subevents.
_ioswd_sub1_pid _ioswd_sub2_pid	A 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.

Embedded Event Manager 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 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 this applet. If no event is specified, this applet is not considered registered and the applet is not displayed. 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, 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 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 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 the EEM event detectors and about creating EEM policies, see the *Writing Embedded Event Manager Policies* document.

Cisco includes a set of sample policies. 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
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.
tm_crash_reporter.tcl	This policy runs 5 seconds after it is registered. If the policy is saved in the configuration, it will also run each time 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.

The sample policies feature the timer and syslog event detectors. The timer event detector generates the following time-based events:

- Watchdog
- Countdown

- Absolute time
- CRON

Watchdog timer events occur when a timer counts down to zero and automatically rearm when they reach zero. Countdown timer events occur when a timer counts down to zero without being rearmed. An absolute timer event occurs when an absolute time of day is passed. CRON timer events occur when the CRON string specification matches the current time.

The syslog event detector screens syslog messages using Posix regular expressions. An event can be triggered after one or more occurrences of a message match. An additional modifier can be specified to require that the number of message match occurrences happen within a set period of time in order for an event to be triggered.

For more details about the sample policies, see the "EEM Event Detector Demo: Example" section on page 31.

How to Configure Embedded Event Manager 2.2

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 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.

EEM Policies

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 Tcl.

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* **syslog** [**priority** *priority-level*] **msg** *msg-text*
6. Repeat Step 5.
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 memory-fail`	Registers the applet with the Embedded Event Manager (EEM) 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 10`	Specifies the event criteria that cause the EEM applet to run. • In this example, an EEM event is triggered when one of the fields specified by an SNMP object ID crosses a defined threshold. • Exit criteria are optional, and if not specified, event monitoring is reenabled immediately.

	Command or Action	Purpose
Step 5	**action** *label* **syslog** [**priority** *priority-level*] **msg** *msg-text* **Example:** `Router(config-applet)# action 1.0 syslog priority critical msg "Memory exhausted; current available memory is $_snmp_oid_val bytes"`	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.
Step 6	Repeat Step 5. **Example:** `Router(config-applet)# action 2.0 force-switchover`	(Optional) Repeat Step 5 to add other action CLI commands to the applet.
Step 7	**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 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.

EEM Policies

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 Tcl.

Prerequisites

You must have a policy available that is written in the Tcl scripting language. Three sample policies—sl_intf_down.tcl, tm_cli_cmd.tcl, and tm_crash_reporter.tcl—are provided, 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 for all the required environment variables.
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 every second minute, every hour of every day.
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.

	Command or Action	Purpose
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. • 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
```

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 either be run manually or be run when an EEM applet is triggered. To run the policy, use either the **action policy** command in applet configuration mode or the **event manager run** command in global configuration 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: Example" section on page 29.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **event manager applet** *applet-name*
4. **event none** *policy-filename*
5. **exit**
6. **event manager run** *policy-filename*
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 manual-policy`	Registers the applet with the Embedded Event Manager and enters applet configuration mode.
Step 4	**event none** *policy-filename* **Example:** `Router(config-applet)# event none manual-policy`	Specifies that an EEM policy is to be registered with the EEM and can be run manually.
Step 5	**exit** **Example:** `Router(config-applet)# exit`	Exits applet configuration mode and returns to global configuration mode.
Step 6	**event manager run** *policy-filename* **Example:** `Router(config)# event manager run manual-policy`	Manually runs a registered EEM policy.
Step 7	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode and returns to privileged EXEC mode.

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* [**type** {**system** | **user**}] [**trap**]

5. **exit**
6. Repeat Step 2 to ensure that the policy is 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**] **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 is 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   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
```

Suspending Embedded Event Manager Policy Execution

Perform this task to immediately suspend the execution of all EEM policies. Suspending instead of unregistering policies 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 policy 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**] **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.

	Command or Action	Purpose
Step 4	**event manager scheduler policy suspend** **Example:** `Router(config)# event manager scheduler policy suspend`	Immediately suspends the execution of all EEM policies.
Step 5	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode and returns the router 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
```

Managing Embedded Event Manager Policies

Perform this task to define a location in the local file system containing an installed modular Cisco IOS image to run on all the nodes in a system, or on just one specified node.

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.
Step 4	**event manager directory user** [**library** | **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
```

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.

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 enhanced object tracking event detector publishes an event when the tracked object changes.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **track** *object-number* **stub**
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** **Example:** `Router(config)# track 2 stub`	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 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 the Embedded Event Manager (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 the tracked object represented by the 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 the tracked object represented by the 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 the tracked object represented by the number 2.
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. • Only the syntax applicable to this task is used in this example. For more details, see the *Cisco IOS IP Application Services Command Reference*, Release 12.4T.

Examples

In the following example, the **show track** privileged EXEC command is used to display information about the tracked object represented by the 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 EEM registered 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 10
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 10
 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 Embedded Event Manager 2.2

This section contains the following configuration examples:

Embedded Event Manager Applet Configuration: Example

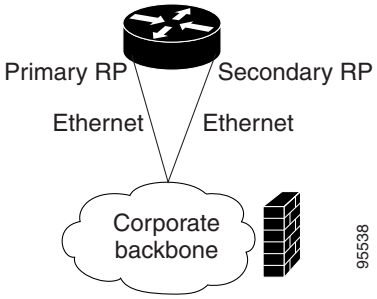
The following example shows how to configure an EEM applet that causes a switch to the secondary (redundant) Route Processor (RP) when the primary RP runs low on memory.

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 2 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 2 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 10
 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 10
 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
```

Configuration for the Primary RP and Secondary RP

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 10
 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: Example

The following examples show how 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"
!
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"
!
event manager applet manual-policy-two
 event none
 action 1.0 policy manual-policy
!
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, configure an EEM applet to be run when the tracked object changes. Actions are specified to both set and read the state of the object.

```
track 10 stub
 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 Event Detector Configuration: Example

The following example shows how to configure three EEM applets to demonstrate how the watchdog system monitor 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"
```

EEM Event Detector Demo: Example

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

- EEM Sample Policy Descriptions
- Event Manager Environment Variables for the Sample Policies
- Registration of Some EEM Policies
- Basic Configuration Details for All Sample Policies
- Using the Sample Policies

EEM Sample Policy Descriptions

This configuration example features the three sample EEM policies:

- sl_intf_down.tcl—Will be run when a configurable syslog message is logged. It will execute up to two configurable CLI commands and will e-mail the results.
- tm_cli_cmd.tcl—Will be run using a configurable CRON entry. It will execute a configurable CLI command and will e-mail the results.
- tm_crash_reporter.tcl—Will be run 5 seconds after it is registered and 5 seconds after the router boots up. When triggered, the script will attempt to find the reload reason. If the reload reason was due to a crash, the policy will search for the related crashinfo file and send this information to a URL location specified by the user in the environment variable _crash_reporter_url.

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 1 on page 7 for a list of the e-mail variables); only _email_cc is optional. Other required variable settings are outlined in the following tables.

Table 4 describes the EEM environment variables that must be set before the tm_cli_cmd.tcl sample policy is run.

Table 4 Environment Variables Required for the tm_cli_cmd.tcl Policy

Environment Variable	Description	Example
_cron_entry	A CRON specification that determines when the policy will run. See the *Writing Embedded Event Manager Policies* document for more information about how to specify a cron entry.	0-59/1 0-23/1 * * 0-7
_show_cmd	The CLI command to be executed when the policy is run.	**show version**

Table 5 describes the EEM environment variables that must be set before the sl_intf_down.tcl sample policy is run.

Table 5 Environment Variables Required for the sl_intf_down.tcl Policy

Environment Variable	Description	Example
_syslog_pattern	A regular expression pattern match string that is used to compare syslog messages to determine when the policy runs.	.*UPDOWN.*FastEthernet0/0.*
_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**

Table 6 describes the EEM environment variables that must be set before the tm_crash_reporter.tcl sample policy is run.

Table 6 Environment Variables Required for 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.swpkg.cisco.internal.com/fm/interface_tm.cgi

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 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:

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

Using the Sample Policies

This section contains the following configuration scenarios to demonstrate how to use the three sample Tcl policies:

- Running the sl_intf_down.tcl Sample Policy
- Running the tm_cli_cmd.tcl Sample Policy
- Running the tm_crash_reporter.tcl Sample Policy

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, if specified, _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 will run when an interface goes down. Enter the **show event manager environment** command to display the current environment variable values. Unplug the cable 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 the interface process crash. 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
config 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" will cause this policy to be run every other minute. 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 entering the **configure terminal** command to reach global configuration mode, the tm_cli_cmd.tcl policy can be registered 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
config 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, then the policy will be triggered 5 seconds after bootup. When triggered, the script will attempt to find the reload reason. If the reload reason was due to a crash, the policy will search for the related crashinfo file and send 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 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 running 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 will display 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 entering the **configure terminal** command to reach global configuration mode, the tm_crash_reporter.tcl policy can be registered 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
 exit
show event manager policy registered
```

Where to Go Next

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 Embedded Event Manager 2.2.

Related Documents

Related Topic	Document Title
EEM commands: complete command syntax, defaults, command mode, command history, usage guidelines, and examples	*Cisco IOS Network Management Command Reference*, Release 12.4
Embedded Event Manager policy writing using Tcl	*Writing Embedded Event Manager Policies*
Embedded Resource Manager	"Embedded Resource Manager" module
Configuring enhanced object tracking	"Configuring Enhanced Object Tracking" module
CNS event agent	*CNS Event Agent* feature document, Release 12.2(2)T
CNS Configuration Engine	*Cisco CNS Configuration Registrar: Installing and Configuring the IE2100*

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

Command Reference

This section documents new and modified commands only.

New Commands

- action track read
- action track set
- default-state
- event resource
- event rf
- event track
- track stub

Modified Commands

- show track

action track read

To specify the action of reading the state of a tracked object when an Embedded Event Manager (EEM) applet is triggered, use the **action track read** command in applet configuration mode. To remove the **action track read** command from the configuration, use the **no** form of this command.

action *label* **track read** *object-number*

no action *label* **track read** *object-number*

Syntax Description

label	Unique identifier that can be any string value. Actions are sorted and run in ascending alphanumeric key sequence using the label as the sort key. If the string contains embedded blanks, enclose it in double quotation marks.
object-number	Tracked object number in the range from 1 to 500, inclusive. The number is defined using the **track stub** command.

Command Default

The state of a tracked object is not read.

Command Modes

Applet configuration

Command History

Release	Modification
12.4(2)T	This command was introduced.

Usage Guidelines

This command generates the following result variable:

_track_state—State of the specified tracked object. The text string returned is either up or down. If the state is up, it means that the object exists and is in an up state. If the state is down, it means that the object either does not exist or is in a down state.

This command is used to help track objects using EEM. Each tracked object is identified by a unique number that is specified on the tracking command-line interface (CLI). Client processes such as EEM 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 enhanced object tracking event detector publishes an EEM event when the tracked object changes.

Examples

The following example shows how to specify event criteria based on a tracked object:

```
event manager applet track-ten
 event track 10 state any
 action 1.0 track set 10 state up
 action 2.0 track read 10
```

Related Commands

Command	Description
action track set	Specifies the action of setting the state of a tracked object when an EEM applet is triggered.
event manager applet	Registers an event applet with the Embedded Event Manager and enters applet configuration mode.
show track	Displays tracking information.
track stub	Creates a stub object to be tracked.

action track set

To specify the action of setting the state of a tracked object when an Embedded Event Manager (EEM) applet is triggered, use the **action track set** command in applet configuration mode. To remove the **action track set** command from the configuration, use the **no** form of this command.

action *label* **track set** *object-number* **state** {**up** | **down**}

no action *label* **track set** *object-number* **state** {**up** | **down**}

Syntax Description

label	Unique identifier that can be any string value. Actions are sorted and run in ascending alphanumeric key sequence using the label as the sort key. If the string contains embedded blanks, enclose it in double quotation marks.
object-number	Tracked object number in the range from 1 to 500, inclusive. The number is defined using the **track stub** command.
state	Specifies the state to which the tracked object will be set.
up	Specifies that the state of the tracked object will be set to up.
down	Specifies that the state of the tracked object will be set to down.

Command Default

The state of a tracked object is not set.

Command Modes

Applet configuration

Command History

Release	Modification
12.4(2)T	This command was introduced.

Usage Guidelines

This command generates the following result variable:

- _track_state—State of the specified tracked object. The text string returned is either up or down. If the state is up, it means that the object exists and is in an up state. If the state is down, it means that the object either does not exist or is in a down state.

This command is used to help track objects using EEM. Each tracked object is identified by a unique number that is specified on the tracking command-line interface (CLI). Client processes such as EEM 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 enhanced object tracking event detector publishes an EEM event when the tracked object changes.

Examples

The following example shows how to specify event criteria based on a tracked object:

```
event manager applet track-ten
 event track 10 state any
 action 1.0 track set 10 state up
 action 2.0 track read 10
```

Related Commands

Command	Description
action track read	Specifies the action of reading the state of a tracked object when an EEM applet is triggered.
event manager applet	Registers an event applet with the Embedded Event Manager and enters applet configuration mode.
show track	Displays tracking information.
track stub	Creates a stub object to be tracked.

default-state

To set the default state for a stub object, use the **default-state** command in tracking configuration mode. To reset the default state to its internal default state, use the **no** form of this command.

default-state {**up** | **down**}

no default-state {**up** | **down**}

Syntax Description

up	Sets the current default state of a stub object to up.
down	Sets the current default state of a stub object to down.

Command Default

Internal default state is the default.

Command Modes

Tracking configuration

Command History

Release	Modification
12.4(2)T	This command was introduced.

Usage Guidelines

Use the **default-state** command to set the default state of a stub object that has been created by the **track stub** command. The stub object can be tracked and manipulated by an external process, Embedded Event Manager (EEM).

EEM is a distributed, scalable, 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 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.

Examples

The following example shows how to create a stub object and configure a default state for the stub object:

```
track 2 stub
 default-state up
```

Related Commands

Command	Description
show track	Displays tracking information.
track stub	Creates a stub object to be tracked.

event resource

To specify the event criteria for an Embedded Event Manager (EEM) applet that is run on the basis of an Embedded Resource Manager (ERM) event report for a specified policy, use the **event resource** command in applet configuration mode. To remove the report event criteria, use the **no** form of this command.

event [*label*] **resource policy** *policy-filename*

no event [*label*] **resource policy** *policy-filename*

Syntax Description

label	(Optional) Unique identifier that can be any string. If the string contains embedded blanks, enclose it in double quotation marks.
policy	Indicates that a specific policy is identified.
policy-filename	Policy name.

Command Default

No EEM event criteria are specified.

Command Modes

Applet configuration

Command History

Release	Modification
12.4(2)T	This command was introduced.

Usage Guidelines

The resource event detector publishes an event when the 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 the 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 CPU, buffer, and memory.

Examples

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"
```

Related Commands

Command	Description
event manager applet	Registers an event applet with the Embedded Event Manager and enters applet configuration mode.

event rf

To specify the event criteria for an Embedded Event Manager (EEM) applet that is run on the basis of Redundancy Framework (RF) state change notifications, use the **event rf** command in applet configuration mode. To remove the RF event criteria, use the **no** form of this command.

event rf event *rf-state-name*

no event rf event *rf-state-name*

Syntax Description

event	Compares the regular expression contained in the *rf-state-name* argument with an RF state change notification. If there is a match, an event is triggered. The *rf-state-name* argument takes one of the following values: • RF_EVENT_CLIENT_PROGRESSION • RF_EVENT_CONTINUE_PROGRESSION • RF_EVENT_GO_ACTIVE • RF_EVENT_GO_ACTIVE_EXTRALOAD • RF_EVENT_GO_ACTIVE_HANDBACK • RF_EVENT_GO_STANDBY • RF_EVENT_KEEP_ALIVE • RF_EVENT_KEEP_ALIVE_TMO • RF_EVENT_LOCAL_PROG_DONE • RF_EVENT_NEGOTIATE • RF_EVENT_NOTIFICATION_TMO • RF_EVENT_PEER_PROG_DONE • RF_EVENT_STANDBY_PROGRESSION • RF_EVENT_START_PROGRESSION • RF_EVENT_SWACT_INHIBIT_TMO • RF_PROG_ACTIVE • RF_PROG_ACTIVE_DRAIN • RF_PROG_ACTIVE_FAST • RF_PROG_ACTIVE_POSTCONFIG • RF_PROG_ACTIVE_PRECONFIG • 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_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

Command Default

No EEM events are triggered.

Command Modes

Applet configuration

Command History

Release	Modification
12.4(2)T	This command was introduced.

Usage Guidelines

An EEM event is triggered when the expression in the *rf-state-name* argument matches an RF state change notification. The RF event detector publishes an event when one or more RF events occur during synchronization in a dual Route Processor (RP) system.

Examples

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"
```

Related Commands

Command	Description
event manager applet	Registers an event applet with the Embedded Event Manager and enters applet configuration mode.

event track

To specify the event criteria for an Embedded Event Manager (EEM) applet that is run on the basis of a Cisco IOS Object Tracking subsystem report for the specified object number, use the **event track** command in applet configuration mode. To remove the report event criteria, use the **no** form of this command.

event [*label*] **track** *object-number* [**state** {**up** | **down** | **any**}]

no event [*label*] **track** *object-number* [**state** {**up** | **down** | **any**}]

Syntax Description

label	(Optional) Unique identifier that can be any string. If the string contains embedded blanks, enclose it in double quotation marks.
object-number	Tracked object number in the range from 1 to 500, inclusive. The number is defined using the **track stub** command.
state	(Optional) Specifies that the tracked object transition will cause an event to be raised.
up	(Optional) Specifies that an event will be raised when the tracked object transitions from a down state to an up state.
down	(Optional) Specifies that an event will be raised when the tracked object transitions from an up state to a down state.
any	(Optional) Specifies that an event will be raised when the tracked object transitions to or from any state. This is the default.

Command Default

No EEM event criteria are specified.

Command Modes

Applet configuration

Command History

Release	Modification
12.4(2)T	This command was introduced.

Usage Guidelines

There are two entry variables associated with this command:

- _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.”

This command is used to help track objects using EEM. Each tracked object is identified by a unique number that is specified on the tracking command-line interface (CLI). Client processes such as EEM 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.

Examples

The following example shows how to specify event criteria based on a tracked object:

```
event manager applet track-ten
 event track 10 state any
 action 1.0 track set 10 state up
 action 2.0 track read 10
```

Related Commands

Command	Description
action track read	Specifies the action of reading the state of a tracked object when an EEM applet is triggered.
action track set	Specifies the action of setting the state of a tracked object when an EEM applet is triggered.
event manager applet	Registers an event applet with the Embedded Event Manager and enters applet configuration mode.
show track	Displays tracking information.
track stub	Creates a stub object to be tracked.

show track

To display information about objects that are tracked by the tracking process, use the **show track** command in privileged EXEC mode.

show track [*object-number* [**brief**] | **interface** [**brief**] | **ip route** [**brief**] | **resolution** | **timers**]

Syntax Description

object-number	(Optional) Object number that represents the object to be tracked. Range is from 1 to 500.
brief	(Optional) Displays a single line of information related to the preceding argument or keyword.
interface	(Optional) Displays tracked interface objects.
ip route	(Optional) Displays tracked IP-route objects.
resolution	(Optional) Displays resolution of tracked parameters.
timers	(Optional) Displays polling interval timers.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(15)T	This command was introduced.
12.3(8)T	The output was enhanced to include the track-list objects.
12.4(2)T	The output was enhanced to display stub objects.

Usage Guidelines

Use this command to display information about objects that are tracked by the tracking process. When no arguments or keywords are specified, information for all objects is displayed.

Examples

The following example shows information about the state of IP routing on the interface that is being tracked:

```
Router# show track 1

Track 1
 Interface Ethernet0/2 ip routing
 IP routing is Down (no IP addr)
  1 change, last change 00:01:08
 Tracked by:
  HSRP Ethernet0/3 1
```

The following example shows information about the line-protocol state on the interface that is being tracked:

```
Router# show track 1

Track 1
 Interface Ethernet0/1 line-protocol
 Line protocol is Up
  1 change, last change 00:00:05
 Tracked by:
  HSRP Ethernet0/3 1
```

The following example shows information about the reachability of a route that is being tracked:

```
Router# show track 1

Track 1
 IP route 10.16.0.0 255.255.0.0 reachability
 Reachability is Up (RIP)
  1 change, last change 00:02:04
 First-hop interface is Ethernet0/1
 Tracked by:
  HSRP Ethernet0/3 1
```

The following example shows information about the threshold metric of a route that is being tracked:

```
Router# show track 1

Track 1
 IP route 10.16.0.0 255.255.0.0 metric threshold
 Metric threshold is Up (RIP/6/102)
  1 change, last change 00:00:08
 Metric threshold down 255 up 254
 First-hop interface is Ethernet0/1
 Tracked by:
  HSRP Ethernet0/3 1
```

The following example shows the object type, the interval in which it is polled, and the time until the next poll:

```
Router# show track timers

 Object type   Poll Interval  Time to next poll
 interface     1              expired
 ip route      30             29.364
```

Table 7 describes the significant fields shown in the displays.

Table 7 show track Field Descriptions

Field	Description
Track	Object number that is being tracked.
Interface Ethernet0/2 ip routing	Interface type, number, and object that is being tracked.
IP routing is	State value of the object, displayed as Up or Down. If the object is down, the reason is displayed.
1 change, last change	Number of times that the state of a tracked object has changed and the time (in hh:mm:ss) since the last change.
Tracked by	Client process that is tracking the object.

Table 7 show track Field Descriptions (continued)

Field	Description
First-hop interface is	Displays the first-hop interface.
Object type	Object type that is being tracked.
Poll Interval	Interval (in seconds) in which the tracking process polls the object.
Time to next poll	Period of time, in seconds, until the next polling of the object.

The following output shows that there are two objects. Object 1 has been configured with a weight of 10 "down," and object 2 has been configured with a weight of 20 "up." Object 1 is down (expressed as 0/10) and object 2 is up. The total weight of the tracked list is 20 with a maximum of 30 (expressed as 20/30). The "up" threshold is 20, so the list is "up."

```
Router# show track

 Track 6
 List threshold weight
  Threshold weight is Up (20/30)
   1 change, last change 00:00:08
   object 1 Down (0/10)
   object 2 weight 20 Up (20/30)
  Threshold weight down 10 up 20
   Tracked by:
    HSRP Ethernet0/3 1
```

The following example shows information about the Boolean configuration:

```
Router# show track

 Track 3
 List boolean and
 Boolean AND is Down
  1 change, last change 00:00:08
   object 1 not Up
   object 2 Down
 Tracked by:
  HSRP Ethernet0/3 1
```

Table 8 describes the significant fields shown in the displays.

Table 8 show track Field Descriptions

Field	Description
Track	Object number that is being tracked.
Boolean AND is Down	Each object defined in the list must be in a down state.
1 change, last change	Number of times that the state of a tracked object has changed and the time (in hh:mm:ss) since the last change.
Tracked by	Client process that is tracking the object; in this case, HSRP.

The following example shows information about a stub object that has been created to be tracked using Embedded Event Manager (EEM):

```
Router# show track

Track 1
  Stub-object
  State is Up
    1 change, last change 00:00:04, by Undefined
```

The following example shows information about a stub object when the **brief** keyword is used:

```
Router# show track brief

Track   Object                          Parameter        Value Last Change
1       Stub-object Undefined                            Up    00:00:12
```

Table 9 describes the significant fields shown in the displays.

Table 9 show track brief Field Descriptions

Field	Description
Track	Object number that is being tracked.
Object	Definition of stub object.
Parameter	Tracking parameters.
Value	State value of the object, displayed as Up or Down.
Last Change	Time (in hh:mm:ss) since the state of a tracked object last changed.

Related Commands

Command	Description
track interface	Configures an interface to be tracked and enters tracking configuration mode.
track ip route	Tracks the state of an IP route and enters tracking configuration mode.

track stub

To create a stub object that can be tracked by Embedded Event Manager (EEM) and to enter tracking configuration mode, use the **track stub** command in global configuration mode. To remove the stub object, use the **no** form of this command.

track *object-number* **stub**

no track *object-number* **stub**

Syntax Description

object-number	Object number that represents the object to be tracked. The range is from 1 to 500.

Command Default

No stub objects are created.

Command Modes

Global configuration

Command History

Release	Modification
12.4(2)T	This command was introduced.

Usage Guidelines

Use the **track stub** command to create a stub object, which is an object that can be tracked and manipulated by an external process, EEM. After the stub object is created, the **default-state** command can be used to set the default state of the stub object.

EEM is a distributed, scalable, 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 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.

Examples

In the following example, stub object 1 is created and configured with a default state of up.

```
track 1 stub
 default-state up
```

Related Commands

Command	Description
default-state	Sets the default state for a stub object.
show track	Displays tracking information.