

Configuring and Managing Embedded Event Manager Policies

The Cisco IOS XR Software Embedded Event Manager (EEM) functions as the central clearing house for the events detected by any portion of the Cisco IOS XR Software processor failover services. The EEM is responsible for detection of fault events, fault recovery, and process reliability statistics in a Cisco IOS XR Software system. The EEM events are notifications that something significant has occurred within the system, such as:

- Operating or performance statistics outside the allowable values (for example, free memory dropping below a critical threshold).
- Online insertion or removal (OIR).
- Termination of a process.

The EEM relies on software agents or event detectors to notify it when certain system events occur. When the EEM has detected an event, it can initiate corrective actions. Actions are prescribed in routines called *policies*. Policies must be registered before an action can be applied to collected events. No action occurs unless a policy is registered. A registered policy informs the EEM about a particular event that is to be detected and the corrective action to be taken if that event is detected. When such an event is detected, the EEM enables the corresponding policy. You can disable a registered policy at any time.

The EEM monitors the reliability rates achieved by each process in the system, allowing the system to detect the components that compromise the overall reliability or availability.

This module describes the tasks you need to perform to configure and manage EEM policies on your network and write and customize the EEM policies using Tool Command Language (Tcl) scripts to handle faults and events.

- Prerequisites for Configuring and Managing Embedded Event Manager Policies, on page 2
- Information About Configuring and Managing Embedded Event Manager Policies, on page 2
- How to Configure and Manage Embedded Event Manager Policies, on page 11

Prerequisites for Configuring and Managing Embedded Event Manager Policies

You must be in a user group associated with a task group that includes the proper task IDs. The command reference guides include the task IDs required for each command. If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.

Information About Configuring and Managing Embedded Event Manager Policies

Event Management

Embedded Event Manager (EEM) in the Cisco IOS XR Software system essentially involves system event management. An event can be any significant occurrence (not limited to errors) that has happened within the system. The Cisco IOS XR Software EEM detects those events and implements appropriate responses.

The EEM enables a system administrator to specify appropriate action based on the current state of the system. For example, a system administrator can use EEM to request notification by e-mail when a hardware device needs replacement.

The EEM interacts with routines, "event detectors," that actively monitor the system for events. The EEM relies on an event detector that it has provided to syslog to detect that a certain system event has occurred. It uses a pattern match with the syslog messages and also relies on a timer event detector to detect that a certain time and date has occurred.

When the EEM has detected an event, it can initiate actions in response. These actions are contained in routines called policy handlers. Policies are defined by Tcl scripts (EEM scripts) written by the user through a Tcl API. While the data for event detection is collected, no action occurs unless a policy for responding to that event has been registered. At registration, a policy informs the EEM that it is looking for a particular event. When the EEM detects the event, it enables the policy.

The EEM monitors the reliability rates achieved by each process in the system. These metrics can be used during testing to determine which components do not meet their reliability or availability goals so that corrective action can be taken.

System Event Processing

When the EEM receives an event notification, it takes these actions:

- Checks for established policy handlers and if a policy handler exists, the EEM initiates callback routines (*EEM handlers*) or runs Tool Command Language (Tcl) scripts (*EEM scripts*) that implement policies. The policies can include built-in EEM actions.
- Notifies the processes that have *subscribed* for event notification.
- Records reliability metric data for each process in the system.
- Provides access to EEM-maintained system information through an application program interface (API).

Embedded Event Manager Scripts

When the EEM has detected an event, it can initiate corrective actions prescribed in routines called policies. Policies must be registered before any action can be applied to collected events. No action occurs unless a policy is registered. A registered policy informs the EEM about a particular event to detect and the corrective action to take if that event is detected. When such an event is detected, the EEM runs the policy. Tool Command Language (Tcl) is used as the scripting language to define policies and all Embedded Event Manager scripts are written in Tcl. EEM scripts are identified to the EEM using the **event manager policy** configuration command. An EEM script remains available to be scheduled by the EEM until the **no event manager policy** command is entered.

In addition the onboard Tcl scripts that come with the IOS XR operating system, users may write their own TCL-based policies. Cisco provides enhancements to the Tcl language in the form of Tcl command extensions that facilitate the writing of EEM policies. For more information about EEM Tcl command extensions, see Embedded Event Manager Policy Tcl Command Extension Categories, on page 3

Writing an EEM script includes the following steps:

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

Embedded Event Manager Policy Tcl Command Extension Categories

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

Table 1: Embedded Event Manager Tcl Command Extension Categories

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

Category	Definition
EEM context Tcl command extensions	These Tcl command extensions are used to store and retrieve a Tcl context (the visible variables and their values).

Cisco File Naming Convention for Embedded Event Manager

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

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

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

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

Table 2: Two-Character Abbreviation Specification

Two-Character Abbreviation	Specification
ap	event_register_appl
ct	event_register_counter
st	event_register_stat
no	event_register_none
oi	event_register_oir
pr	event_register_process
sl	event_register_syslog
tm	event_register_timer
ts	event_register_timer_subscriber
wd	event_register_wdsysmon

Embedded Event Manager Built-in Actions

EEM built-in actions can be requested from EEM handlers when the handlers run.

This table describes each EEM handler request or action.

Table 3: Embedded Event Manager Built-In Actions

Embedded Event Manager Built-In Action	Description
Log a message to syslog	Sends a message to the syslog. Arguments to this action are priority and the message to be logged.
Execute a CLI command	Writes the command to the specified channel handler to execute the command by using the **cli_exec** command extension.
Generate a syslog message	Logs a message by using the **action_syslog** Tcl command extension.
Manually run an EEM policy	Runs an EEM policy within a policy while the **event manager run** command is running a policy in mode.
Publish an application-specific event	Publishes an application-specific event by using the **event_publish appl** Tcl command extension.
Reload the Cisco IOS software	Causes a router to be reloaded by using the EEM **action_reload** command.
Request system information	Represents the sys_reqinfo_xxx family of system-specific information commands by a policy to gather system information.
Send a short e-mail	Sends the e-mail out using Simple Mail Transfer Protocol (SMTP).
Set or modify a counter	Modifies a counter value.

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

Application-specific Embedded Event Management

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

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

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

Event Detection and Recovery

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. Event detectors are separate programs that provide an interface between other Cisco IOS XR Software components and the EEM. Event detectors (event publishers) screen events and publish them when there is a match on an event specification that is provided by event subscribers (policies). Event detectors notify the EEM server when an event of interest occurs.

An EEM event is defined as a notification that something significant has happened within the system. Two categories of events exist:

- System EEM events
- Application-defined events

System EEM events are built into the EEM and are grouped based on the fault detector that raises them. They are identified by a symbolic identifier defined within the API.

Some EEM system events are monitored by the EEM whether or not an application has requested monitoring. These are called *built-in* EEM events. Other EEM events are monitored only if an application has requested EEM event monitoring. EEM event monitoring is requested through an EEM application API or the EEM scripting interface.

Some event detectors can be distributed to other hardware cards within the same secure domain router (SDR) or within the administration plane to provide support for distributed components running on those cards.

These event detectors are supported:

System Manager Event Detector

The System Manager Event Detector has four roles:

- Records process reliability metric data.
- Screens for processes that have EEM event monitoring requests outstanding.
- Publishes events for those processes that match the screening criteria.
- Asks the System Manager to perform its default action for those events that do not match the screening criteria.

The System Manager Event Detector interfaces with the System Manager to receive process startup and termination notifications. The interfacing is made through a private API available to the System Manager. To minimize overhead, a portion of the API resides within the System Manager process space. When a process terminates, the System Manager invokes a helper process (if specified in the process.startup file) before calling the Event Detector API.

Processes can be identified by component ID, System Manager assigned job ID, or load module pathname plus process instance ID. Process instance ID is an integer assigned to a process to differentiate it from other processes with the same pathname. The first instance of a process is assigned an instance ID value of 1, the second 2, and so on.

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

Table 4: System Manager Event Detector Event Monitoring Requests

Embedded Event Manager Event	Description
Normal process termination EEM event—built in	Occurs when a process matching the screening criteria terminates.
Abnormal process termination EEM event—built in	Occurs when a process matching the screening criteria terminates abnormally.
Process startup EEM event—built in	Occurs when a process matching the screening criteria starts.

When System Manager Event Detector abnormal process termination events occur, the default action restarts the process according to the built-in rules of the System Manager.

The relationship between the EEM and System Manager is strictly through the private API provided by the EEM to the System Manager for the purpose of receiving process start and termination notifications. When the System Manager calls the API, reliability metric data is collected and screening is performed for an EEM event match. If a match occurs, a message is sent to the System Manager Event Detector. In the case of abnormal process terminations, a return is made indicating that the EEM handles process restart. If a match does not occur, a return is made indicating that the System Manager should apply the default action.

Timer Services Event Detector

The Timer Services Event Detector implements time-related EEM events. These events are identified through user-defined identifiers so that multiple processes can await notification for the same EEM event.

The Timer Services Event Detector handles EEM event monitoring requests for the Date/Time Passed EEM event. This event occurs when the current date or time passes the specified date or time requested by an application.

Syslog Event Detector

The syslog Event Detector implements syslog message screening for syslog EEM events. This routine interfaces with the syslog daemon through a private API. To minimize overhead, a portion of the API resides within the syslog daemon process.

Screening is provided for the message severity code or the message text fields.

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

Table 5: Syslog Event Detector Event Monitoring Requests

Embedded Event Manager Event	Description
Syslog message EEM event	Occurs for a just-logged message. It occurs when there is a match for either the syslog message severity code or the syslog message text pattern. Both can be specified when an application requests a syslog message EEM event.

Embedded Event Manager Event	Description
Process event manager EEM event—built in	Occurs when the event-processed count for a specified process is either greater than or equal to a specified maximum or is less than or equal to a specified minimum.

None Event Detector

The None Event Detector publishes an event when the Cisco IOS XR Software **event manager run** CLI command executes an EEM policy. EEM schedules and runs policies on the basis of an event specification that is contained within the policy itself. An EEM policy must be identified and registered to be permitted to run manually before the **event manager run** command will execute.

Event manager none detector provides user the ability to run a tcl script using the CLI. The script is registered first before running. Cisco IOS XR Software version provides similar syntax with Cisco IOS EEM (refer to the applicable EEM Documentation for details), so scripts written using Cisco IOS EEM is run on Cisco IOS XR Software with minimum change.

Watchdog System Monitor Event Detector

Watchdog System Monitor (IOSXRWDSysMon) Event Detector for Cisco IOS XR Software

The Cisco IOS XR Software Watchdog System Monitor Event Detector publishes an event when one of the following occurs:

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



Note Cisco IOS XR Software processes are used to distinguish them from Cisco IOS XR Software Modularity processes.

Two events may be monitored at the same time, and the event publishing criteria can be specified to require one event or both events to cross their specified thresholds.

The Cisco IOS XR Software Watchdog System Monitor Event Detector handles the events as shown in this table.

Table 6: Watchdog System Monitor Event Detector Requests

Embedded Event Manager Event	Description
Process percent CPU EEM event—built in	Occurs when the CPU time for a specified process is either greater than or equal to a specified maximum percentage of available CPU time or is less than or equal to a specified minimum percentage of available CPU time.
Total percent CPU EEM event—built in	Occurs when the CPU time for a specified processor complex is either greater than or equal to a specified maximum percentage of available CPU time or is less than or equal to a specified minimum percentage of available CPU time.

Embedded Event Manager Event	Description
Process percent memory EEM event—built in	Occurs when the memory used for a specified process has either increased or decreased by a specified value.
Total percent available Memory EEM event—built in	Occurs when the available memory for a specified processor complex has either increased or decreased by a specified value.
Total percent used memory EEM event—built in	Occurs when the used memory for a specified processor complex has either increased or decreased by a specified value.

Watchdog System Monitor (WDSysMon) Event Detector for Cisco IOS XR Software Modularity

The Cisco IOS XR Software Software Modularity Watchdog System Monitor Event Detector detects infinite loops, deadlocks, and memory leaks in Cisco IOS XR Software Modularity processes.

Distributed Event Detectors

Cisco IOS XR Software components that interface to EEM event detectors and that have substantially independent implementations running on a distributed hardware card should have a distributed EEM event detector. The distributed event detector permits scheduling of EEM events for local processes without requiring that the local hardware card to the EEM communication channel be active.

These event detectors run on a Cisco IOS XR Software line card:

- System Manager Fault Detector
- Wdsysmon Fault Detector
- Counter Event Detector
- OIR Event Detector
- Statistic Event Detector

Embedded Event Manager Event Scheduling and Notification

When an EEM handler is scheduled, it runs under the context of the process that creates the event request (or for EEM scripts under the Tcl shell process context). For events that occur for a process running an EEM handler, event scheduling is blocked until the handler exits. The defined default action (if any) is performed instead.

The EEM Server maintains queues containing event scheduling and notification items across client process restarts, if requested.

Reliability Statistics

Reliability metric data for the system is maintained by the EEM. The data is periodically written to checkpoint. Reliability metric data is kept for each hardware card and for each process handled by the System Manager.

Hardware Card Reliability Metric Data

Hardware card reliability metric data is recorded in a table indexed by disk ID.

Data maintained by the hardware card is as follows:

- Most recent start time
- Most recent normal end time (controlled switchover)
- Most recent abnormal end time (asynchronous switchover)
- Most recent abnormal type
- Cumulative available time
- Cumulative unavailable time
- Number of times hardware card started
- Number of times hardware card shut down normally
- Number of times hardware card shut down abnormally

Process Reliability Metric Data

Reliability metric data is kept for each process handled by the System Manager. This data includes standby processes running on either the primary or backup hardware card. Data is recorded in a table indexed by hardware card disk ID plus process pathname plus process instance for those processes that have multiple instances.

Process terminations include the following cases:

- Normal termination—Process exits with an exit value equal to 0.
- Abnormal termination by process—Process exits with an exit value not equal to 0.
- Abnormal termination by Linux—Linux operating system terminates the process.
- Abnormal termination by kill process API—API kill process terminates the process.

Data to be maintained by process is as follows:

- Most recent process start time
- Most recent normal process end time
- Most recent abnormal process end time
- Most recent abnormal process end type
- Previous ten process end times and types
- Cumulative process available time
- Cumulative process unavailable time
- Cumulative process run time (the time when the process is actually running on the CPU)
- Number of times started
- Number of times ended normally
- Number of times ended abnormally

- Number of abnormal failures within the past 60 minutes
- Number of abnormal failures within the past 24 hours
- Number of abnormal failures within the past 30 days

How to Configure and Manage Embedded Event Manager Policies

Configuring Environmental Variables

EEM environmental variables are Tcl global variables that are defined external to the policy before the policy is run. The EEM policy engine receives notifications when faults and other events occur. EEM policies implement recovery, based on the current state of the system and actions specified in the policy for a given event. Recovery actions are triggered when the policy is run.

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

You can configure the environment variable and values by using the **event manager environment***var-name var-value* command.

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

Configuration Example

This example shows how to define a set of EEM environment variables.

```
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# event manager environment _cron_entry 0-59/2 0-23/1 * * 0-7
RP/0/RP0/CPU0:Router(config)# event manager environment _email_from beta@cisco.com
RP/0/RP0/CPU0:Router(config)# event manager environment _email_to beta@cisco.com
RP/0/RP0/CPU0:Router(config)# commit
RP/0/RP0/CPU0:Router(config)# end
RP/0/RP0/CPU0:Router# show event manager environment

No. Name                           Value
1   _email_to                      beta@cisco.com
2   _cron_entry                    0-59/2 0-23/1 * * 0-7
3   _email_from                    beta@cisco.com
RP/0/RP0/CPU0:Router#
```

Registering Embedded Event Manager Policies

You should register an EEM policy to run a policy when an event is triggered. Registering an EEM policy is performed with the **event manager policy** command. An EEM script is available to be scheduled by the EEM until the **no** form of this command is entered. Prior to registering a policy, display EEM policies that are available to be registered with the **show event manager policy available** command.

The EEM schedules and runs policies on the basis of an event specification that is contained within the policy itself. When the **event manager policy** command is invoked, the EEM examines the policy and registers it to be run when the specified event occurs.

You need to specify the following while registering the EEM policy.

- **username**—Specifies the username that runs the script
- **persist-time**—Defines the number of seconds the username authentication is valid. This keyword is optional. The default **persist-time** is 3600 seconds (1 hour).
- **system** or **user**—Specifies the policy as a system defined or user defined policy. This keyword is optional.



Note AAA authorization (such as the **aaa authorization eventmanager** command) must be configured before EEM policies can be registered. See the *Configuring AAA Services* module of *Configuring AAA Services on Cisco IOS XR Software* for more information about AAA authorization configuration.

Once policies have been registered, their registration can be verified through the **show event manager policy registered** command.

Configuration Example

This example shows how to register a user defined EEM policy.

```
RP/0/RP0/CPU0:Router# show event manager policy available
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# event manager policy cron.tcl username tom type user
RP/0/RP0/CPU0:Router# show event manager policy registered
```

How to Write Embedded Event Manager Policies Using Tcl

This section provides information on how to write and customize Embedded Event Manager (EEM) policies using Tool Command Language (Tcl) scripts to handle Cisco IOS XR Software faults and events.

This section contains these tasks:

Registering and Defining an EEM Tcl Script

Perform this task to configure environment variables and register an EEM policy. EEM schedules and runs policies on the basis of an event specification that is contained within the policy itself. When an EEM policy is registered, the software examines the policy and registers it to be run when the specified event occurs.



Note A policy must be available that is written in the Tcl scripting language. Sample policies are stored in the system policy directory.

Configuration Example

This example shows how to register and define an EEM policy.

```
RP/0/RP0/CPU0:Router# show event manager environment all
```

```
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# event manager environment _cron_entry 0-59/2 0-23/1 * * 0-7
RP/0/RP0/CPU0:Router(config)# event manager policy tm_cli_cmd.tcl username user_a type
system
RP/0/RP0/CPU0:Router(config)# commit
RP/0/RP0/CPU0:Router# show event manager policy registered system
```



Note To unregister an EEM policy, use the **no event manager policy** command. This command removes an EEM policy from the running configuration file.

Suspending EEM Policy Execution

Suspending policies, instead of unregistering them, might be necessary for reasons of temporary performance or security. If required, you can immediately suspend the execution of all EEM policies by using the **event manager scheduler suspend** command.

Configuration Example

This example shows how to suspend the execution of all EEM policies.

```
RP/0/RP0/CPU0:Router# show event manager policy registered system
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# event manager scheduler suspend
RP/0/RP0/CPU0:Router(config)# commit
```

Specifying a Directory for Storing EEM Policies

A directory is essential to store the user-defined policy files or user library files. If you do not plan to write EEM policies, you do not have to create the directory. The EEM searches the user policy directory when you enter the **event manager policy** *policy-name* **user**command. To create a user policy directory before identifying it to the EEM, use the **mkdir** command. After creating the user policy directory, use the copy command to copy the policy files into the user policy directory. You can use the **show event manager directory user [library | policy]** command to display the directory to use for EEM user library files or user-defined policy files.

Configuration Example

This example shows how to specify a directory to use for storing user-library files .

```
RP/0/RP0/CPU0:Router# show event manager directory user library
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# event manager directory user library
harddisk:/mirror/EEM/usr/lib/tcl
RP/0/RP0/CPU0:Router(config)# commit
```

Programming EEM Policies with Tcl

Perform this task to help you program a policy using Tcl command extensions. We recommend that you copy an existing policy and modify it. There are two required parts that must exist in an EEM Tcl policy: the event_register Tcl command extension and the body. For detailed information about the Tcl policy structure and requirements, see EEM Policies Using TCL: Details, on page 24

Note Configure a directory under 'harddisk:/mirror/EEM' instead of 'disk0:'. If the directory path is under 'disk0:', a warning message is displayed to configure a directory under the 'harddisk:/mirror/EEM' path, since 'disk0:' will not be persistent across upgrades.

Procedure

	Command or Action	Purpose
Step 1	**show event manager policy available** [**system** | **user**] **Example:** `RP/0/RP0/CPU0:Router# show event manager policy available`	Displays EEM policies that are available to be registered.
Step 2	Cut and paste the contents of the sample policy displayed on the screen to a text editor.	—
Step 3	Define the required event_register Tcl command extension.	Choose the appropriate event_register Tcl command extension for the event that you want to detect, and add it to the policy. The following are valid Event Registration Tcl Command Extensions: • event_register_appl • event_register_counter • event_register_stat • event_register_wdsysmon • event_register_oir • event_register_process • event_register_syslog • event_register_timer • event_register_timer_subscriber • event_register_hardware • event_register_none
Step 4	Add the appropriate namespace under the ::cisco hierarchy.	Policy developers can use the new namespace ::cisco in Tcl policies to group all the extensions used by Cisco IOS XR EEM. There are two namespaces under the ::cisco hierarchy. The following are the namespaces

	Command or Action	Purpose
		and the EEM Tcl command extension categories that belongs under each namespace: • ::cisco::eem • EEM event registration • EEM event information • EEM event publish • EEM action • EEM utility • EEM context library • EEM system information • CLI library • ::cisco::lib • SMTP library **Note** Ensure that the appropriate namespaces are imported, or use the qualified command names when using the preceding commands.
Step 5	Program the must defines section to check for each environment variable that is used in this policy.	This is an optional step. Must defines is a section of the policy that tests whether any EEM environment variables that are required by the policy are defined before the recovery actions are taken. The must defines section is not required if the policy does not use any EEM environment variables. EEM environment variables for EEM scripts are Tcl global variables that are defined external to the policy before the policy is run. To define an EEM environment variable, use the EEM configuration command **event manager environment** . By convention, all Cisco EEM environment variables begin with "_" (an underscore). To avoid future conflict, customers are urged not to define new variables that start with "_". **Note** You can display the Embedded Event Manager environment variables set on your system by using the **show event manager environment** command.

	Command or Action	Purpose
		For example, EEM environment variables defined by the sample policies include e-mail variables. The sample policies that send e-mail must have the following variables set in order to function properly. The following are the e-mail-specific environment variables used in the sample EEM policies. • **_email_server**—A Simple Mail Transfer Protocol (SMTP) mail server used to send e-mail (for example, mailserver.example.com) • **_email_to**—The address to which e-mail is sent (for example, engineering@example.com) • **_email_from**—The address from which e-mail is sent (for example, devtest@example.com) • **_email_cc**—The address to which the e-mail must be copied (for example, manager@example.com)
Step 6	Program the body of the script.	In this section of the script, you can define any of the following: • The **event_reqinfo** event information Tcl command extension that is used to query the EEM for information about the detected event. • The action Tcl command extensions, such as **action_syslog**, that are used to specify actions specific to EEM. • The system information Tcl command extensions, such as **sys_reqinfo_routername**, that are used to obtain general system information. • The **context_save** and **context_retrieve** Tcl command extensions that are used to save Tcl variables for use by other policies. • Use of the SMTP library (to send e-mail notifications) or the CLI library (to run CLI commands) from a policy.

	Command or Action	Purpose
Step 7	Check the entry status to determine if a policy has previously run for this event.	If the prior policy is successful, the current policy may or may not require execution. Entry status designations may use one of three possible values: 0 (previous policy was successful), Not=0 (previous policy failed), and Undefined (no previous policy was executed).
Step 8	Check the exit status to determine whether or not to apply the default action for this event, if a default action exists.	A value of zero means that the default action should not be performed. A value of nonzero means that the default action should be performed. The exit status is passed to subsequent policies that are run for the same event.
Step 9	Set Cisco Error Number (_cerrno) Tcl global variables.	Some EEM Tcl command extensions set a Cisco Error Number Tcl global variable _cerrno. Whenever _cerrno is set, four other Tcl global variables are derived from _cerrno and are set along with it (_cerr_sub_num, _cerr_sub_err, , and _cerr_str).
Step 10	Save the Tcl script with a new filename, and copy the Tcl script to the router.	Embedded Event Manager policy filenames adhere to the following specification: • An optional prefix—Mandatory.—indicating, if present, that this is a system policy that should be registered automatically at boot time if it is not already registered. For example: Mandatory.sl_text.tcl. • A filename body part containing a two-character abbreviation (see Table 2: Two-Character Abbreviation Specification, on page 4) for the first event specified, an underscore character part, and a descriptive field part further identifying the policy. • A filename suffix part defined as .tcl. For more details, see theCisco File Naming Convention for Embedded Event Manager, on page 4. Copy the file to the harddisk on the router—typically harddisk:/mirror/.
Step 11	**configure**	Enters global configuration mode.

	Command or Action	Purpose
Step 12	**event manager directory user** {**library** *path* | **policy** *path*} **Example:** `RP/0/RP0/CPU0:Router(config)# event manager directory user library harddisk:/mirror/EEM/user_library`	Specifies a directory to use for storing user library files or user-defined EEM policies.
Step 13	**event manager policy** *policy-name* **username** *username* [**persist-time** [*seconds* | **infinite**] | **type** [**system** | **user**]] **Example:** `RP/0/RP0/CPU0:Router(config)# event manager policy test.tcl username user_a type user`	Registers the EEM policy to be run when the specified event defined within the policy occurs.
Step 14	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.
Step 15	Cause the policy to execute, and observe the policy.	—
Step 16	Use debugging techniques if the policy does not execute correctly.	—

Creating an EEM User Tcl Library Index

Perform this task to create an index file that contains a directory of all the procedures contained in a library of Tcl files. This task allows you to test library support in EEM Tcl. In this task, a library directory is created to contain the Tcl library files, the files are copied into the directory, and an index tclIndex) is created that contains a directory of all the procedures in the library files. If the index is not created, the Tcl procedures are not found when an EEM policy that references a Tcl procedure is run.

Procedure

	Command or Action	Purpose

Step 1	On your workstation (UNIX, Linux, PC, or Mac) create a library directory and copy the Tcl library files into the directory.	The following example files can be used to create a tclIndex on a workstation running the Tcl shell:
		lib1.tcl
		`proc test1 {} {`
		`  puts "In procedure test1"`
		`}`
		`proc test2 {} {`
		`  puts "In procedure test2"`
		`}`
		lib2.tcl
		`proc test3 {} {`
		`  puts "In procedure test3"`
		`}`

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

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

	Command or Action	Purpose

		`# element name is the name of a command and the value is`
		`# a script that loads the command.`
		`set auto_index(test1) [list source [file join $dir lib1.tcl]]`
		`set auto_index(test2) [list source [file join $dir lib1.tcl]]`
		`set auto_index(test3) [list source [file join $dir lib2.tcl]]`

Step 4	Copy the Tcl library files from step 1 and the tclIndex file from step 3 to the directory used for storing user library files on the target router.	—

Step 5	Copy a user-defined EEM policy file written in Tcl to the directory used for storing user-defined EEM policies on the target router.	The directory can be the same directory used in step 4.
		The following example user-defined EEM policy can be used to test the Tcl library support in EEM:
		libtest.tcl
		`::cisco::eem::event_register_none`
		`namespace import ::cisco::eem::*`
		`namespace import ::cisco::lib::*`
		`global auto_index auto_path`
		`puts [array names auto_index]`
		`if { [catch {test1} result]} {`
		`  puts "calling test1 failed result = $result $auto_path"`
		`}`
		`if { [catch {test2} result]} {`
		`  puts "calling test2 failed result = $result $auto_path"`
		`}`
		`if { [catch {test3} result]} {`
		`  puts "calling test3 failed result = $result $auto_path"`
		`}`

Step 6	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters mode.
Step 7	**event manager directory user library** *path* **Example:** `RP/0/RP0/CPU0:Router(config)# event manager directory user library harddisk:/mirror/EEM/eem_library`	Specifies the EEM user library directory; this is the directory to which the files in step 4 were copied.

	Command or Action	Purpose
Step 8	**event manager directory user policy** *path* **Example:** `RP/0/RP0/CPU0:Router(config)# event manager directory user policy harddisk:/mirror/EEM/eem_policies`	Specifies the EEM user policy directory; this is the directory to which the file in step 5 was copied.
Step 9	**event manager policy** *policy-name* **username** *username* [**persist-time** [*seconds* | **infinite**] | **type** [**system** | **user**]] **Example:** `RP/0/RP0/CPU0:Router(config)# event manager policy libtest.tcl username user_a`	Registers a user-defined EEM policy.
Step 10	**event manager run** *policy* [*argument*] **Example:** `RP/0/RP0/CPU0:Router(config)# event manager run libtest.tcl`	Manually runs an EEM policy.
Step 11	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

Creating an EEM User Tcl Package Index

Perform this task to create a Tcl package index file that contains a directory of all the Tcl packages and version information contained in a library of Tcl package files. Tcl packages are supported using the Tcl **package** keyword.

Tcl packages are located in either the EEM system library directory or the EEM user library directory. When a **package require** Tcl command is executed, the user library directory is searched first for a pkgIndex.tcl file. If the pkgIndex.tcl file is not found in the user directory, the system library directory is searched.

In this task, a Tcl package directory—the pkgIndex.tcl file—is created in the appropriate library directory using the **pkg_mkIndex** command to contain information about all the Tcl packages contained in the directory

along with version information. If the index is not created, the Tcl packages are not found when an EEM policy that contains a **package require** Tcl command is run.

Using the Tcl package support in EEM, users can gain access to packages such as XML_RPC for Tcl. When the Tcl package index is created, a Tcl script can easily make an XML-RPC call to an external entity.

Note Packages implemented in C programming code are not supported in EEM.

Procedure

	Command or Action	Purpose
Step 1	On your workstation (UNIX, Linux, PC, or Mac) create a library directory and copy the Tcl package files into the directory.	–
Step 2	**tclsh** **Example:** `workstation% tclsh`	Enters the Tcl shell.

Step 3	**pkg_mkindex** *directory_name* ***.tcl**
	Example:
	`workstation% pkg_mkindex eem_library *.tcl`
		Use the **pkg_mkindex** command to create the pkgIndex file. The pkgIndex file contains a directory of all the packages contained in the Tcl library files. We recommend that you run the **pkg_mkindex** command inside a directory, because there can be only a single pkgIndex file in any directory and you may have other Tcl files to be grouped together. Running the **pkg_mkindex** command in a directory determines which Tcl package file or files are indexed using a specific pkgIndex.
		The following example pkgIndex is created when some Tcl package files are in a library file directory and the pkg_mkindex command is run:
		pkgIndex
		`# Tcl package index file, version 1.1`
		`# This file is generated by the "pkg_mkIndex" command`
		`# and sourced either when an application starts up or`
		`# by a "package unknown" script. It invokes the`
		`# "package ifneeded" command to set up package-related`
		`# information so that packages will be loaded automatically`
		`# in response to "package require" commands. When this`

	Command or Action	Purpose

		`# script is sourced, the variable $dir must contain the`
		`# full path name of this file's directory.`
		`package ifneeded xmlrpc 0.3 [list source [file join $dir xmlrpc.tcl]]`

Step 4	Copy the Tcl package files from step 1 and the pkgIndex file from step 3 to the directory used for storing user library files on the target router.	—

Step 5	Copy a user-defined EEM policy file written in Tcl to the directory used for storing user-defined EEM policies on the target router.	The directory can be the same directory used in step 4.
		The following example user-defined EEM policy can be used to test the Tcl library support in EEM:
		packagetest.tcl
		`::cisco::eem::event_register_none maxrun 1000000.000`
		`#`
		`# test if xmlrpc available`
		`#`
		`#`
		`# Namespace imports`
		`#`
		`namespace import ::cisco::eem::*`
		`namespace import ::cisco::lib::*`
		`#`
		`package require xmlrpc`
		`puts "Did you get an error?"`

Step 6	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters mode.
Step 7	**event manager directory user library** *path* **Example:** `RP/0/RP0/CPU0:Router(config)# event manager directory user library harddisk:/mirror/EEM/eem_library`	Specifies the EEM user library directory; this is the directory to which the files in step 4 were copied.
Step 8	**event manager directory user policy** *path* **Example:** `RP/0/RP0/CPU0:Router(config)# event manager directory user policy harddisk:/mirror/EEM/eem_policies`	Specifies the EEM user policy directory; this is the directory to which the file in step 5 was copied.

	Command or Action	Purpose
Step 9	**event manager policy** *policy-name* **username** *username* [**persist-time** [*seconds* | **infinite**] | **type** [**system** | **user**]] **Example:** `RP/0/RP0/CPU0:Router(config)# event manager policy packagetest.tcl username user_a`	Registers a user-defined EEM policy.
Step 10	**event manager run** *policy* [*argument*] **Example:** `RP/0/RP0/CPU0:Router(config)# event manager run packagetest.tcl`	Manually runs an EEM policy.
Step 11	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

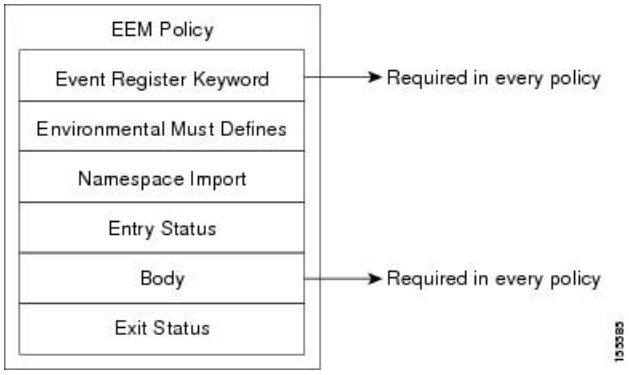
EEM Policies Using TCL: Details

This section provides detailed conceptual information about programming EEM policies using TCL.

Tcl Policy Structure and Requirements

All EEM policies share the same structure, shown in the below figure. There are two parts of an EEM policy that are required: the event_register Tcl command extension and the body. The remaining parts of the policy are optional: environmental must defines, namespace import, entry status, and exit status.

Figure 1: Tcl Policy Structure and Requirements



The start of every policy must describe and register the event to detect using an **event_register** Tcl command extension. This part of the policy schedules the running of the policy. The following example Tcl code shows how to register the **event_register_timer** Tcl command extension:

```
::cisco::eem::event_register_timer cron name crontimer2 cron_entry $_cron_entry maxrun 240
```

The following example Tcl code shows how to check for, and define, some environment variables:

```
# Check if all the env variables that we need exist.
# If any of them does not exist, print out an error msg and quit.
if {![info exists _email_server]} {
  set result \
    "Policy cannot be run: variable _email_server has not been set"
  error $result $errorInfo
}
if {![info exists _email_from]} {
  set result \
    "Policy cannot be run: variable _email_from has not been set"
  error $result $errorInfo
}
if {![info exists _email_to]} {
  set result \
    "Policy cannot be run: variable _email_to has not been set"
  error $result $errorInfo
)
```

The namespace import section is optional and defines code libraries. The following example Tcl code shows how to configure a namespace import section:

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

The body of the policy is a required structure and might contain the following:

- The **event_reqinfo** event information Tcl command extension that is used to query the EEM for information about the detected event.
- The action Tcl command extensions, such as **action_syslog**, that are used to specify actions specific to EEM.
- The system information Tcl command extensions, such as **sys_reqinfo_routername**, that are used to obtain general system information.

- Use of the SMTP library (to send e-mail notifications) or the CLI library (to run CLI commands) from a policy.
- The **context_save** and **con text_retrieve** Tcl command extensions that are used to save Tcl variables for use by other policies.

EEM Entry Status

The entry status part of an EEM policy is used to determine if a prior policy has been run for the same event, and to determine the exit status of the prior policy. If the _entry_status variable is defined, a prior policy has already run for this event. The value of the _entry_status variable determines the return code of the prior policy.

Entry status designations may use one of three possible values:

- 0 (previous policy was successful)
- Not=0 (previous policy failed),
- Undefined (no previous policy was executed).

EEM Exit Status

When a policy finishes running its code, an exit value is set. The exit value is used by the EEM to determine whether or not to apply the default action for this event, if any. A value of zero means that the default action should not be performed. A value of nonzero means that the default action should be performed. The exit status is passed to subsequent policies that are run for the same event.

EEM Policies and Cisco Error Number

Some EEM Tcl command extensions set a Cisco Error Number Tcl global variable known as _cerrno. Whenever the _cerrno variable is set, the other Tcl global variables are derived from _cerrno and are set along with it (_cerr_sub_num, _cerr_sub_err, and _cerr_str).

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

```
XYSSSSSSSSSSSSSEEEEEEEEPPPPPPPP
```

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

Table 7: _cerrno: 32-Bit Error Return Value Variables

Variable	Description
XY	The error class (indicates the severity of the error). This variable corresponds to the first two bits in the 32-bit error return value; 10 in the preceding case, which indicates CERR_CLASS_WARNING: See Table 8: Error Class Encodings, on page 27 for the four possible error class encodings specific to this variable.
SSSSSSSSSSSSS	The subsystem number that generated the most recent error(13 bits = 8192 values). This is the next 13 bits of the 32-bit sequence, and its integer value is contained in $_cerr_sub_num.

Variable	Description
EEEEEEEE	The subsystem specific error number (8 bits = 256 values). This segment is the next 8 bits of the 32-bit sequence, and the string corresponding to this error number is contained in $_cerr_sub_err.

For example, the following error return value might be returned from an EEM Tcl command extension:

```
862439AE
```

This number is interpreted as the following 32-bit value:

```
10000110001001000011100110101110
```

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

Table 8: Error Class Encodings

Error Return Value	Error Class
00	CERR_CLASS_SUCCESS
01	CERR_CLASS_INFO
10	CERR_CLASS_WARNING
11	CERR_CLASS_FATAL

An error return value of zero means SUCCESS.