



# Writing Embedded Event Manager Policies Using Tcl

---

**First Published: October 31, 2005**  
**Last Updated: May 31, 2010**

This module describes how software developers can write and customize Embedded Event Manager (EEM) policies using Tool command language (Tcl) scripts to handle Cisco IOS software faults and events. EEM is a policy-driven process by means of which faults in the Cisco IOS software system are reported through a defined application programming interface (API). The EEM policy engine receives notifications when faults and other events occur. EEM policies implement recovery on the basis of the current state of the system and the actions specified in the policy for a given event. Recovery actions are triggered when the policy is run.

## Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the [“Feature Information for Writing Embedded Event Manager Policies Using Tcl”](#) section on page 255.

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

## Contents

- [Prerequisites for Writing Embedded Event Manager Policies Using Tcl, page 2](#)
- [Information About Writing Embedded Event Manager Policies Using Tcl, page 2](#)
- [How to Write Embedded Event Manager Policies Using Tcl, page 9](#)
- [Configuration Examples for Writing Embedded Event Manager Policies Using Tcl, page 37](#)



---

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

- [Where to Go Next, page 59](#)
- [Additional References, page 60](#)
- [EEM Policy Tcl Command Extension Reference, page 62](#)
- [Feature Information for Writing Embedded Event Manager Policies Using Tcl, page 255](#)

## Prerequisites for Writing Embedded Event Manager Policies Using Tcl

- Before writing EEM policies, you should be familiar with the “[Embedded Event Manager Overview](#)” module.
- If you want to write EEM policies using the command-line interface (CLI) commands, you should be familiar with the “[Writing Embedded Event Manager Policies Using the Cisco IOS CLI](#)” module.

## Information About Writing Embedded Event Manager Policies Using Tcl

To write EEM policies using Tcl, you should understand the following concepts:

- [EEM Policies, page 2](#)
- [EEM Policy Tcl Command Extension Categories, page 3](#)
- [General Flow of EEM Event Detection and Recovery, page 4](#)
- [Safe-Tcl, page 5](#)
- [Bytecode Support for EEM 2.4, page 7](#)
- [Registration Substitution, page 7](#)
- [Cisco File Naming Convention for EEM, page 8](#)

## EEM Policies

EEM offers the ability to monitor events and take informational or corrective action when the monitored events occur or reach a threshold. An EEM policy is an entity that defines an event and the actions to be taken when that event occurs. There are two types of EEM policies: an applet or a script. An applet is a simple form of policy that is defined within the command-line interface (CLI) configuration. A script is a form of policy that is written in Tool Command Language (Tcl).

### EEM Applet

An EEM applet is a concise method for defining event screening criteria and the actions to be taken when that event occurs. In EEM applet configuration mode, three types of configuration statements are supported. The event commands are used to specify the event criteria to trigger the applet to run, the action commands are used to specify an action to perform when the EEM applet is triggered, and the **set** command is used to set the value of an EEM applet variable. Currently only the `_exit_status` variable is supported for the **set** command.

Only one event configuration command is allowed within an applet configuration. When applet configuration submode is exited and no event command is present, a warning is displayed stating that no event is associated with the applet. If no event is specified, the applet is not considered registered. When no action is associated with the applet, events are still triggered but no actions are performed. Multiple action configuration commands are allowed within an applet configuration. Use the **show event manager policy registered** command to display a list of registered applets.

Before modifying an EEM applet, be aware that the existing applet is not replaced until you exit applet configuration mode. While you are in applet configuration mode modifying the applet, the existing applet may be executing. It is safe to modify the applet without unregistering it, because changes are written to a temporary file. When you exit applet configuration mode, the old applet is unregistered and the new version is registered.

Action configuration commands within an applet are uniquely identified using the *label* argument, which can be any string value. Actions are sorted within an applet in ascending alphanumeric key sequence using the *label* argument as the sort key, and they are run using this sequence. The same *label* argument can be used in different applets; the labels must be unique only within one applet.

The Embedded Event Manager schedules and runs policies on the basis of an event specification that is contained within the policy itself. When applet configuration mode is exited, EEM examines the event and action commands that are entered and registers the applet to be run when a specified event occurs.

For more details about writing EEM policies using the Cisco IOS CLI, see the [“Writing Embedded Event Manager Policies Using the Cisco IOS CLI”](#) module.

### EEM Script

All Embedded Event Manager scripts are written in Tcl. Tcl is a string-based command language that is interpreted at run time. The version of Tcl supported is Tcl version 8.3.4 plus added script support. Scripts are defined using an ASCII editor on another device, not on the networking device. The script is then copied to the networking device and registered with EEM. Tcl scripts are supported by EEM. As an enforced rule, Embedded Event Manager policies are short-lived run time routines that must be interpreted and executed in less than 20 seconds of elapsed time. If more than 20 seconds of elapsed time are required, the *maxrun* parameter may be specified in the *event\_register* statement to specify any desired value.

EEM policies use the full range of the Tcl language’s capabilities. However, Cisco provides enhancements to the Tcl language in the form of Tcl command extensions that facilitate the writing of EEM policies. The main categories of Tcl command extensions identify the detected event, the subsequent action, utility information, counter values, and system information.

EEM allows you to write and implement your own policies using Tcl. Writing an EEM script involves:

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

## EEM Policy Tcl Command Extension Categories

There are different categories of EEM policy Tcl command extensions.



### Note

The Tcl command extensions available in each of these categories for use in all EEM policies are described in later sections in this document.

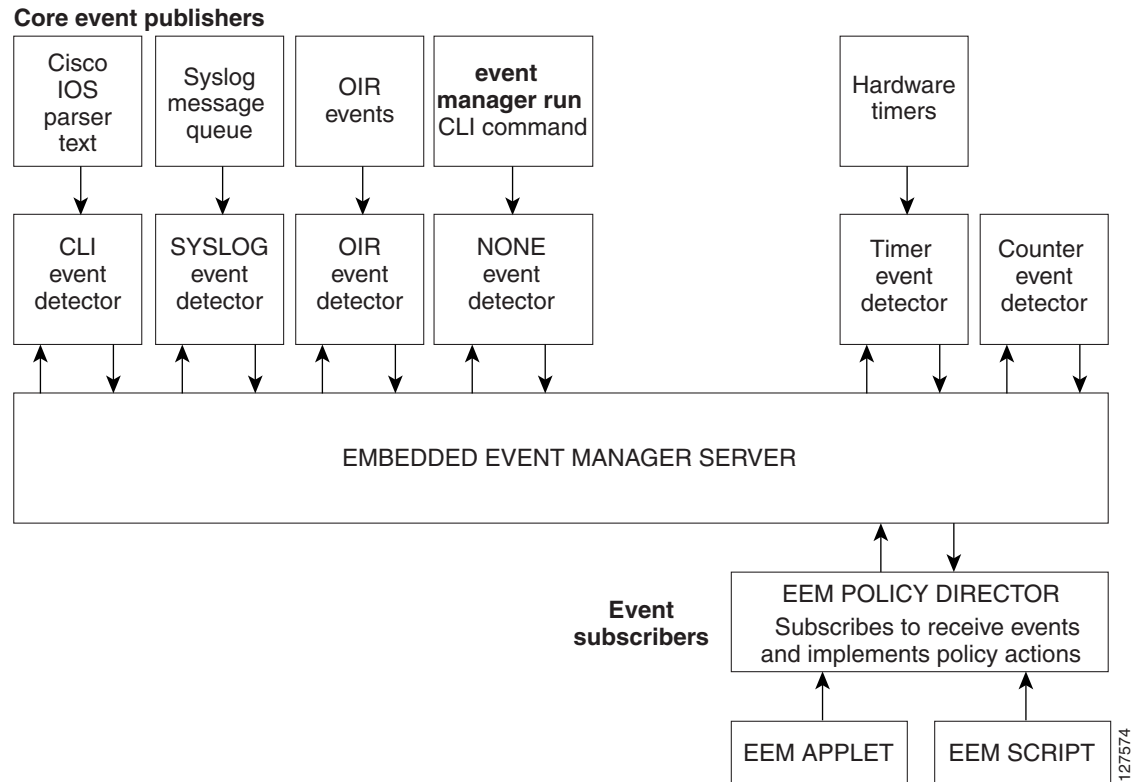
**Table 1** *EEM Policy Tcl Command Extension Categories*

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

## General Flow of EEM Event Detection and Recovery

EEM is a flexible, policy-driven framework that supports in-box monitoring of different components of the system with the help of software agents known as event detectors. [Figure 1](#) shows the relationship between the EEM server, the core event publishers (event detectors), and the event subscribers (policies). Basically, event publishers screen events and publish them when there is a match on an event specification that is provided by the event subscriber. Event detectors notify the EEM server when an event of interest occurs.

When an event or fault is detected, Embedded Event Manager determines from the event publishers—an example would be the OIR events publisher in [Figure 1](#)—if a registration for the encountered fault or event has occurred. EEM matches the event registration information with the event data itself. A policy registers for the detected event with the Tcl command extension **event\_register\_xxx**. The event information Tcl command extension **event\_reqinfo** is used in the policy to query the Embedded Event Manager for information about the detected event.

**Figure 1** Embedded Event Manager Core Event Detectors

## Safe-Tcl

Safe-Tcl is a safety mechanism that allows untrusted Tcl scripts to run in an interpreter that was created in the safe mode. The safe interpreter has a restricted set of commands that prevent accessing some system resources and harming the host and other applications. For example, it does not allow commands to access critical Cisco IOS file system directories.

Cisco-defined scripts run in full Tcl mode, but user-defined scripts run in Safe-Tcl mode. Safe-Tcl allows Cisco to disable or customize individual Tcl commands. For more details about Tcl commands, go to <http://www.tcl.tk/man/>.

The following list of Tcl commands are restricted with a few exceptions. Restrictions are noted against each command or command keyword:

- **cd**—Change directory is not allowed to one of the restricted Cisco directory names.
- **encoding**—The commands **encoding names**, **encoding convertfrom**, and **encoding convertto** are permitted. The **encoding system** command with no arguments is permitted, but the **encoding system** command with the **?encoding?** keyword is not permitted.
- **exec**—Not permitted.
- **fconfigure**—Permitted.
- **file**—The following are permitted:
  - **file dirname**
  - **file exists**

- **file extension**
- **file isdirectory**
- **file join**
- **file pathtype**
- **file rootname**
- **file split**
- **file stat**
- **file tail**
- **file**—The following are not permitted:
  - **file atime**
  - **file attributes**
  - **file channels**
  - **file copy**
  - **file delete**
  - **file executable**
  - **file isfile**
  - **file link**
  - **file lstat**
  - **file mkdir**
  - **file mtime**
  - **file nativename**
  - **file normalize**
  - **file owned**
  - **file readable**
  - **file readlink**
  - **file rename**
  - **file rootname**
  - **file separator**
  - **file size**
  - **file system**
  - **file type**
  - **file volumes**
  - **file writable**
- **glob**—The **glob** command is not permitted when searching in one of the restricted Cisco directories. Otherwise, it is permitted.
- **load**—Only files that are in the user policy directory or the user library directory are permitted to be loaded. Static packages (for example, libraries that consist of C code) are not permitted to be loaded with the **load** command.

- **open**—The **open** command is not allowed for a file that is located in one of the restricted Cisco directories.
- **pwd**—The **pwd** command is not permitted.
- **socket**—The **socket** command is permitted.
- **source**—The **source** command is permitted for files that are in the user policy directory or the user library directory.

## Bytecode Support for EEM 2.4

In Cisco IOS Release 12.4(20)T, EEM 2.4 introduces bytecode language (BCL) support by accepting files with the standard bytecode script extension `.tbc`. Tcl version 8.3.4 defines a BCL and includes a compiler that translates Tcl scripts into BCL. Valid EEM policy file extensions in EEM 2.4 for user and system policies are `.tcl` (Tcl Text files) and `.tbc` (Tcl bytecode files).

Storing Tcl scripts in bytecode improves the execution speed of the policy because the code is precompiled, creates a smaller policy size, and obscures the policy code. Obfuscation makes it a little more difficult to modify scripts and hides logic to preserve intellectual property rights.

Support for bytecode is being added to provide another option for release of supported and trusted code. We recommend that you only run well understood, or trusted and supported software on network devices. To generate Tcl bytecode for IOS EEM support, use TclPro versions 1.4 or 1.5.

To translate a Tcl script to bytecode you can use `procomp`, part of Free TclPro Compiler, or Active State Tcl Development Kit. When a Tcl script is compiled using `procomp`, the code is scrambled and a `.tbc` file is generated. The bytecode files are platform-independent and can be generated on any operating system on which TclPro is available, including Windows, Linux, and UNIX. `Procomp` is part of TclPro and available from <http://www.tcl.tk/software/tclpro>.

## Registration Substitution

In addition to regular Tcl substitution, EEM 2.3 (in Cisco IOS Releases 12.2(33)SXH and 12.2(33)SB, and later releases) permits the substitution of an individual parameter in an EEM event registration statement line with an environment variable.

EEM 2.4 in Cisco IOS Release 12.4(20)T introduces the ability to replace multiple parameters in event registration statement lines with a single environment variable.



### Note

---

Only the first environment variable supports multiple parameter substitution. Individual parameters can still be specified with additional environment variables after the initial variable.

---

To illustrate the substitution, a single environment variable, `$_eem_syslog_statement` is configured as:

```
::cisco::eem::event_register_syslog pattern COUNT
```

Using the registration substitution, the `$_eem_syslog_statement` environment variable is used in the following EEM user policy:

```
$_eem_syslog_statement occurs $_eem_occurs_val  
action_syslog "this is test 3"
```

Environment variables must be defined before a policy using them is registered. To define the `$_eem_syslog_statement` environment variable:

```
Router(config)# event manager environment eem_syslog_statement
::cisco::eem::event_register_syslog pattern COUNT
Router(config)# event manager environment eem_occurs_val 2
```

## Cisco File Naming Convention for EEM

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

- An optional prefix—Mandatory.—indicating, if present, that this is a system policy that should be registered automatically at boot time if it is not already registered. For example: `Mandatory.sl_text.tcl`.
- A filename body part containing a two-character abbreviation (see [Table 2](#)) for the first event specified; an underscore part; and a descriptive field part that further identifies the policy.
- A filename suffix part defined as `.tcl`.

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

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

**Table 2** Two-Character Abbreviation Specification

ap	event_register_appl
cl	event_register_cli
ct	event_register_counter
go	event_register_gold
if	event_register_interface
io	event_register_ioswdsysmon
la	event_register_ipsla
nf	event_register_nf
no	event_register_none
oi	event_register_oir
pr	event_register_process
rf	event_register_rf
rs	event_register_resource
rt	event_register_routing
rp	event_register_rpc
sl	event_register_syslog
sn	event_register_snmp
st	event_register_snmp_notification
so	event_register_snmp_object



**Table 2** Two-Character Abbreviation Specification (continued)

tm	event_register_timer
tr	event_register_track
ts	event_register_timer_subscriber
wd	event_register_wdsysmon

## How to Write Embedded Event Manager Policies Using Tcl

This section contains the following tasks:

- [Registering and Defining an EEM Tcl Script, page 9](#)
- [Displaying EEM Registered Policies, page 11](#)
- [Unregistering EEM Policies, page 12](#)
- [Suspending EEM Policy Execution, page 14](#)
- [Managing EEM Policies, page 16](#)
- [Modifying History Table Size and Displaying EEM History Data, page 17](#)
- [Displaying Software Modularity Process Reliability Metrics Using EEM, page 18](#)
- [Modifying the Sample EEM Policies, page 20](#)
- [Programming EEM Policies with Tcl, page 22](#)
- [Creating an EEM User Tcl Library Index, page 31](#)
- [Creating an EEM User Tcl Package Index, page 34](#)

### Registering and Defining an EEM Tcl Script

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

#### Prerequisites

You must have a policy available that is written in the Tcl scripting language. Sample policies are provided—see the details in the “[Sample EEM Policies](#)” section on [page 20](#) to see which policies are available for the Cisco IOS release image that you are using—and these sample policies are stored in the system policy directory.

#### SUMMARY STEPS

1. **enable**
2. **show event manager environment** [**all** | *variable-name*]
3. **configure terminal**
4. **event manager environment** *variable-name string*

5. Repeat [Step 4](#) to configure all the environment variables required by the policy to be registered in [Step 6](#).
6. **event manager policy** *policy-filename* [**type** {**system** | **user**}] [**trap**]
7. **exit**

## DETAILED STEPS

	Command or Action	Purpose
Step 1	<b>enable</b>  <b>Example:</b> Router> enable	Enables privileged EXEC mode. <ul style="list-style-type: none"> <li>Enter your password if prompted.</li> </ul>
Step 2	<b>show event manager environment</b> [ <b>all</b>   <i>variable-name</i> ]  <b>Example:</b> Router# show event manager environment all	(Optional) Displays the name and value of EEM environment variables. <ul style="list-style-type: none"> <li>The optional <b>all</b> keyword displays all the EEM environment variables.</li> <li>The optional <i>variable-name</i> argument displays information about the specified environment variable.</li> </ul>
Step 3	<b>configure terminal</b>  <b>Example:</b> Router# configure terminal	Enters global configuration mode.
Step 4	<b>event manager environment</b> <i>variable-name string</i>  <b>Example:</b> Router(config)# event manager environment _cron_entry 0-59/2 0-23/1 * * 0-6	Configures the value of the specified EEM environment variable. <ul style="list-style-type: none"> <li>In this example, the software assigns a CRON timer environment variable to be set to the second minute of every hour of every day.</li> </ul>
Step 5	Repeat <a href="#">Step 4</a> to configure all the environment variables required by the policy to be registered in <a href="#">Step 6</a> .	—
Step 6	<b>event manager policy</b> <i>policy-filename</i> [ <b>type</b> { <b>system</b>   <b>user</b> }] [ <b>trap</b> ]  <b>Example:</b> 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. <ul style="list-style-type: none"> <li>Use the <b>system</b> keyword to register a Cisco-defined system policy.</li> <li>Use the <b>user</b> keyword to register a user-defined system policy.</li> <li>Use the <b>trap</b> keyword to generate an SNMP trap when the policy is triggered.</li> <li>In this example, the sample EEM policy named <i>tm_cli_cmd.tcl</i> is registered as a system policy.</li> </ul>
Step 7	<b>exit</b>  <b>Example:</b> Router(config)# exit	Exits global configuration mode and returns to privileged EXEC mode.

## Examples

In the following example, the **show event manager environment** privileged EXEC command is used to display the name and value of all EEM environment variables.

```
Router# show event manager environment all

No.  Name                               Value
1    _cron_entry                          0-59/2 0-23/1 * * 0-6
2    _show_cmd                            show ver
3    _syslog_pattern                      .*UPDOWN.*Ethernet1/0.*
4    _config_cmd1                        interface Ethernet1/0
5    _config_cmd2                        no shut
```

## Displaying EEM Registered Policies

Perform this optional task to display EEM registered policies.

### SUMMARY STEPS

1. **enable**
2. **show event manager policy registered** [*event-type event-name*] [**time-ordered** | **name-ordered**] [**detailed** *policy-filename*]

### DETAILED STEPS

#### Step 1 enable

Enables privileged EXEC mode. Enter your password if prompted.

```
Router> enable
```

#### Step 2 show event manager policy registered [*event-type event-name*] [**time-ordered** | **name-ordered**] [**detailed** *policy-filename*]

Use this command with the **time-ordered** keyword to display information about currently registered policies sorted by time, for example:

```
Router# show event manager policy registered time-ordered

No.  Type      Event Type          Trap  Time Registered      Name
1    system   timer cron           Off   Wed May11 01:43:18 2005 tm_cli_cmd.tcl
    name {crontimer2} cron entry {0-59/1 0-23/1 * * 0-7}
    nice 0 priority normal maxrun 240
2    system   syslog              Off   Wed May11 01:43:28 2005 sl_intf_down.tcl
    occurs 1 pattern {.*UPDOWN.*Ethernet1/0.*}
    nice 0 priority normal maxrun 90
3    system   proc abort          Off   Wed May11 01:43:38 2005 pr_cdp_abort.tcl
    instance 1 path {cdp2.iosproc}
    nice 0 priority normal maxrun 20
```

Use this command with the **name-ordered** keyword to display information about currently registered policies sorted by name, for example:

```
Router# show event manager policy registered name-ordered

No.  Type      Event Type          Trap  Time Registered      Name
```

```

1  system  proc  abort                Off   Wed May11  01:43:38 2005  pr_cdp_abort.tcl
   instance 1 path {cdp2.iosproc}
   nice 0 priority normal maxrun 20

2  system  syslog                    Off   Wed May11  01:43:28 2005  sl_intf_down.tcl
   occurs 1 pattern {.*UPDOWN.*Ethernet1/0.*}
   nice 0 priority normal maxrun 90

3  system  timer cron                 Off   Wed May11  01:43:18 2005  tm_cli_cmd.tcl
   name {crontimer2} cron entry {0-59/1 0-23/1 * * 0-7}
   nice 0 priority normal maxrun 240

```

Use this command with the **event-type** keyword to display information about currently registered policies for the event type specified in the *event-name* argument, for example:

```
Router# show event manager policy registered event-type syslog
```

```

No.  Type   Event Type           Time Registered           Name
1    system syslog          Wed May11 01:43:28 2005  sl_intf_down.tcl
   occurs 1 pattern {.*UPDOWN.*Ethernet1/0.*}
   nice 0 priority normal maxrun 90

```

---

## Unregistering EEM Policies

Perform this task to remove an EEM policy from the running configuration file. Execution of the policy is canceled.

### SUMMARY STEPS

1. **enable**
2. **show event manager policy registered** [**event-type** *event-name*] [**system** | **user**] [**time-ordered** | **name-ordered**] [**detailed** *policy-filename*]
3. **configure terminal**
4. **no event manager policy** *policy-filename*
5. **exit**
6. Repeat [Step 2](#) to ensure that the policy has been removed.

## DETAILED STEPS

	Command or Action	Purpose
Step 1	<b>enable</b>  <b>Example:</b> Router> enable	Enables privileged EXEC mode. <ul style="list-style-type: none"> <li>Enter your password if prompted.</li> </ul>
Step 2	<b>show event manager policy registered</b> [ <i>event-type event-name</i> ] [ <i>system   user</i> ] [ <i>time-ordered   name-ordered</i> ] [ <i>detailed policy-filename</i> ]  <b>Example:</b> Router# show event manager policy registered	(Optional) Displays the EEM policies that are currently registered. <ul style="list-style-type: none"> <li>The optional <b>system</b> or <b>user</b> keyword displays the registered system or user policies.</li> <li>If no keywords are specified, EEM registered policies for all event types are displayed in time order.</li> </ul>
Step 3	<b>configure terminal</b>  <b>Example:</b> Router# configure terminal	Enters global configuration mode.
Step 4	<b>no event manager policy</b> <i>policy-filename</i>  <b>Example:</b> Router(config)# no event manager policy pr_cdp_abort.tcl	Removes the EEM policy from the configuration, causing the policy to be unregistered. <ul style="list-style-type: none"> <li>In this example, the <b>no</b> form of the command is used to unregister a specified policy.</li> </ul>
Step 5	<b>exit</b>  <b>Example:</b> Router(config)# exit	Exits global configuration mode and returns to privileged EXEC mode.
Step 6	Repeat <a href="#">Step 2</a> to ensure that the policy has been removed.  <b>Example:</b> Router# show event manager policy registered	—

## Examples

In the following example, the **show event manager policy registered** privileged EXEC command is used to display the three EEM policies that are currently registered:

```
Router# show event manager policy registered
```

```
No.  Type    Event Type      Trap  Time Registered      Name
1   system  timer cron        Off   Tue Oct11 01:43:18 2005 tm_cli_cmd.tcl
    name {crontimer2} cron entry {0-59/1 0-23/1 * * 0-7}
    nice 0 priority normal maxrun 240.000

2   system  syslog          Off   Tue Oct11 01:43:28 2005 sl_intf_down.tcl
    occurs 1 pattern {.*UPDOWN.*Ethernet1/0.*}
    nice 0 priority normal maxrun 90.000
```

```

3    system  proc  abort                Off   Tue Oct11  01:43:38 2005 pr_cdp_abort.tcl
instance 1 path {cdp2.iosproc}
nice 0 priority normal maxrun 20.000

```

After the current policies are displayed, it is decided to delete the `pr_cdp_abort.tcl` policy using the **no** form of the **event manager policy** command:

```

Router# configure terminal
Router(config)# no event manager policy pr_cdp_abort.tcl
Router(config)# exit

```

The **show event manager policy registered** privileged EXEC command is entered again to display the EEM policies that are currently registered. The policy `pr_cdp_abort.tcl` is no longer registered.

```

Router# show event manager policy registered

No.  Type      Event Type          Trap  Time Registered      Name
1    system  timer  cron                Off   Tue Oct11  01:45:17 2005 tm_cli_cmd.tcl
name {crontimer2} cron entry {0-59/1 0-23/1 * * 0-7}
nice 0 priority normal maxrun 240.000

2    system  syslog              Off   Tue Oct11  01:45:27 2005 sl_intf_down.tcl
occurs 1 pattern {.*UPDOWN.*Ethernet1/0.*}
nice 0 priority normal maxrun 90.000

```

## Suspending EEM Policy Execution

Perform this task to immediately suspend the execution of all EEM policies. Suspending policies, instead of unregistering them, might be necessary for reasons of temporary performance or security.

### SUMMARY STEPS

1. **enable**
2. **show event manager policy registered** [**event-type** *event-name*] [**system** | **user**] [**time-ordered** | **name-ordered**] [**detailed** *policy-filename*]
3. **configure terminal**
4. **event manager scheduler suspend**
5. **exit**

## DETAILED STEPS

	Command or Action	Purpose
Step 1	<b>enable</b>  <b>Example:</b> Router> enable	Enables privileged EXEC mode. <ul style="list-style-type: none"> <li>Enter your password if prompted.</li> </ul>
Step 2	<b>show event manager policy registered</b> [ <b>event-type</b> <i>event-name</i> ] [ <b>system</b>   <b>user</b> ] [ <b>time-ordered</b>   <b>name-ordered</b> ] [ <b>detailed</b> <i>policy-filename</i> ]  <b>Example:</b> Router# show event manager policy registered	(Optional) Displays the EEM policies that are currently registered. <ul style="list-style-type: none"> <li>The optional <b>system</b> or <b>user</b> keyword displays the registered system or user policies.</li> <li>If no keywords are specified, EEM registered policies for all event types are displayed in time order.</li> </ul>
Step 3	<b>configure terminal</b>  <b>Example:</b> Router# configure terminal	Enters global configuration mode.
Step 4	<b>event manager scheduler suspend</b>  <b>Example:</b> Router(config)# event manager scheduler suspend	Immediately suspends the execution of all EEM policies.
Step 5	<b>exit</b>  <b>Example:</b> Router(config)# exit	Exits global configuration mode and returns to privileged EXEC mode.

## Examples

In the following example, the **show event manager policy registered** privileged EXEC command is used to display all the EEM registered policies:

```
Router# show event manager policy registered

No.  Type    Event Type      Trap  Time Registered      Name
1    system  timer cron        Off   Sat Oct11 01:43:18 2003  tm_cli_cmd.tcl
    name {crontimer2} cron entry {0-59/1 0-23/1 * * 0-7}
    nice 0 priority normal maxrun 240.000

2    system  syslog          Off   Sat Oct11 01:43:28 2003  sl_intf_down.tcl
    occurs 1 pattern {.*UPDOWN.*Ethernet1/0.*}
    nice 0 priority normal maxrun 90.000

3    system  proc abort      Off   Sat Oct11 01:43:38 2003  pr_cdp_abort.tcl
    instance 1 path {cdp2.iosproc}
    nice 0 priority normal maxrun 20.000
```

The **event manager scheduler suspend** command is entered to immediately suspend the execution of all EEM policies:

```
Router# configure terminal
Router(config)# event manager scheduler suspend
```

```
*Nov 2 15:34:39.000: %HA_EM-6-FMS_POLICY_EXEC: fh_io_msg: Policy execution has been
suspended
```

## Managing EEM Policies

Perform this task to specify a directory to use for storing user library files or user-defined EEM policies.



### Note

This task applies only to EEM policies that are written using Tcl scripts.

### SUMMARY STEPS

1. **enable**
2. **show event manager directory user [library | policy]**
3. **configure terminal**
4. **event manager directory user {library path | policy path}**
5. **exit**

### DETAILED STEPS

	Command or Action	Purpose
Step 1	<b>enable</b>  <b>Example:</b> Router> enable	Enables privileged EXEC mode. <ul style="list-style-type: none"> <li>• Enter your password if prompted.</li> </ul>
Step 2	<b>show event manager directory user [library   policy]</b>  <b>Example:</b> Router# show event manager directory user library	(Optional) Displays the directory to use for storing EEM user library or policy files. <ul style="list-style-type: none"> <li>• The optional <b>library</b> keyword displays the directory to use for user library files.</li> <li>• The optional <b>policy</b> keyword displays the directory to use for user-defined EEM policies.</li> </ul>
Step 3	<b>configure terminal</b>  <b>Example:</b> Router# configure terminal	Enters global configuration mode.



	Command or Action	Purpose
Step 4	<pre>event manager directory user {library path   policy path}</pre> <p><b>Example:</b> Router(config)# event manager directory user library disk0:/usr/lib/tcl </p>	<p>Specifies a directory to use for storing user library files or user-defined EEM policies.</p> <ul style="list-style-type: none"> <li>Use the <i>path</i> argument to specify the absolute pathname to the user directory.</li> </ul>
Step 5	<pre>exit</pre> <p><b>Example:</b> Router(config)# exit </p>	<p>Exits global configuration mode and returns to privileged EXEC mode.</p>

## Examples

In the following example, the **show event manager directory user** privileged EXEC command is used to display the directory, if it exists, to use for storing EEM user library files:

```
Router# show event manager directory user library

disk0:/usr/lib/tcl
```

## Modifying History Table Size and Displaying EEM History Data

Perform this optional task to change the size of the history tables and to display EEM history data.

### SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **event manager history size {events | traps} [size]**
4. **exit**
5. **show event manager history events [detailed] [maximum number]**
6. **show event manager history traps {server | policy}**

### DETAILED STEPS

- 
- Step 1** **enable**  
Enables privileged EXEC mode. Enter your password if prompted.  
Router> **enable**
- Step 2** **configure terminal**  
Enters global configuration mode.  
Router# **configure terminal**

**Step 3 event manager history size {events | traps} [size]**

Use this command to change the size of the EEM event history table or the size of the EEM SNMP trap history table. In the following example, the size of the EEM event history table is changed to 30 entries:

```
Router(config)# event manager history size events 30
```

**Step 4 exit**

Exits global configuration mode and returns to privileged EXEC mode.

```
Router(config)# exit
```

**Step 5 show event manager history events [detailed] [maximum number]**

Use this command to display information about each EEM event that has been triggered.

```
Router# show event manager history events
```

No.	Time of Event	Event Type	Name
1	Fri Sep 9 13:48:40 2005	syslog	applet: one
2	Fri Sep 9 13:48:40 2005	syslog	applet: two
3	Fri Sep 9 13:48:40 2005	syslog	applet: three
4	Fri Sep 9 13:50:00 2005	timer cron	script: tm_cli_cmd.tcl
5	Fri Sep 9 13:51:00 2005	timer cron	script: tm_cli_cmd.tcl

**Step 6 show event manager history traps [server | policy]**

Use this command to display the EEM SNMP traps that have been sent either from the EEM server or from an EEM policy.

```
Router# show event manager history traps
```

No.	Time	Trap Type	Name
1	Fri Sep 9 13:48:40 2005	server	applet: four
2	Fri Sep 9 13:57:03 2005	policy	script: no_snmp_test.tcl

## Displaying Software Modularity Process Reliability Metrics Using EEM

Perform this optional task to display reliability metrics for Cisco IOS Software Modularity processes. The **show event manager metric processes** command was introduced in Cisco IOS Release 12.2(18)SXF4 and later releases and is supported only in Software Modularity images.

### SUMMARY STEPS

1. **enable**
2. **show event manager metric processes {all | process-name}**

### DETAILED STEPS

**Step 1 enable**

Enables privileged EXEC mode. Enter your password if prompted.

```
Router> enable
```

**Step 2** `show event manager metric process {all | process-name}`

Use this command to display the reliability metric data for processes. The system keeps a record of when processes start and end, and this data is used as the basis for reliability analysis. In this partial example, the first and last entries showing the metric data for the processes on all the cards inserted in the system are displayed.

```
Router# show event manager metric process all

=====
process name: devc-pty, instance: 1
sub_system id: 0, version: 00.00.0000
-----
last event type: process start
recent start time: Fri Oct10 20:34:40 2005
recent normal end time: n/a
recent abnormal end time: n/a
number of times started: 1
number of times ended normally: 0
number of times ended abnormally: 0
most recent 10 process start times:
-----
Fri Oct10 20:34:40 2005
-----

most recent 10 process end times and types:

cumulative process available time: 6 hours 30 minutes 7 seconds 378 milliseconds
cumulative process unavailable time: 0 hours 0 minutes 0 seconds 0 milliseconds
process availability: 0.100000000
number of abnormal ends within the past 60 minutes (since reload): 0
number of abnormal ends within the past 24 hours (since reload): 0
number of abnormal ends within the past 30 days (since reload): 0
.
.
.
=====
process name: cdp2.iosproc, instance: 1
sub_system id: 0, version: 00.00.0000
-----
last event type: process start
recent start time: Fri Oct10 20:35:02 2005
recent normal end time: n/a
recent abnormal end time: n/a
number of times started: 1
number of times ended normally: 0
number of times ended abnormally: 0
most recent 10 process start times:
-----
Fri Oct10 20:35:02 2005
-----

most recent 10 process end times and types:

cumulative process available time: 6 hours 29 minutes 45 seconds 506 milliseconds
cumulative process unavailable time: 0 hours 0 minutes 0 seconds 0 milliseconds
process availability: 0.100000000
number of abnormal ends within the past 60 minutes (since reload): 0
number of abnormal ends within the past 24 hours (since reload): 0
number of abnormal ends within the past 30 days (since reload): 0
```

## Troubleshooting Tips

Use the **debug event manager** command in privileged EXEC mode to troubleshoot EEM command operations. Use any debugging command with caution because the volume of output generated can slow or stop the router operations. We recommend that this command be used only under the supervision of a Cisco engineer.

## Modifying the Sample EEM Policies

Perform this task to modify one of the sample policies. Cisco IOS software contains some sample policies in the images that contain the Embedded Event Manager. Developers of EEM policies may modify these policies by customizing the event for which the policy is to be run and the options associated with logging and responding to the event. In addition, developers may select the actions to be implemented when the policy runs.

## Sample EEM Policies

Cisco includes a set of sample policies shown in [Table 3](#). You can copy the sample policies to a user directory and then modify the policies, or you can write your own policies. Tcl is currently the only Cisco-supported scripting language for policy creation. Tcl policies can be modified using a text editor such as Emacs. Policies must execute within a defined number of seconds of elapsed time, and the time variable can be configured within a policy. The default is currently 20 seconds.

[Table 3](#) describes the sample EEM policies.

**Table 3** *Sample EEM Policy Descriptions*

Name of Policy	Description
pr_cdp_abort.tcl	Introduced in Cisco IOS Release 12.2(18)SXF4 Software Modularity images. This policy monitors for cdp2.iosproc process abort events. It will log a message to SYSLOG and send an e-mail with the details of the abort.
pr_crash_reporter.tcl	Introduced in Cisco IOS Release 12.2(18)SXF4 Software Modularity images. This policy monitors for all process abort events. When an event occurs, the policy will send crash information, including the crashdump file, to the specified URL where a CGI script processes the data.
pr_iprouting_abort.tcl	Introduced in Cisco IOS Release 12.2(18)SXF4 Software Modularity images. This policy monitors for iprouting.iosproc process abort events. It will log a message to SYSLOG and send an e-mail with the details of the abort.
sl_intf_down.tcl	This policy runs when a configurable syslog message is logged. It will execute a configurable CLI command and e-mail the results.
tm_cli_cmd.tcl	This policy runs using a configurable CRON entry. It will execute a configurable CLI command and e-mail the results.

**Table 3** Sample EEM Policy Descriptions (continued)

Name of Policy	Description
tm_crash_history.tcl	Introduced in Cisco IOS Release 12.2(18)SXF4 Software Modularity images. This policy runs at midnight every day and e-mails a process crash history report to a specified e-mail address.
tm_crash_reporter.tcl	Introduced in Cisco IOS Release 12.4(2)T. This policy runs 5 seconds after it is registered. If the policy is saved in the configuration, it will also run each time that the router is reloaded. The policy will prompt for the reload reason. If the reload was due to a crash, the policy will search for the latest crashinfo file and send this information to a specified URL location.
tm_fsyst_usage.tcl	Introduced in Cisco IOS Release 12.2(18)SXF4 Software Modularity images. This policy runs using a configurable CRON entry and monitors disk space usage. A syslog message will be displayed if disk space usage crosses configurable thresholds.
wd_mem_reporter.tcl	Introduced in Cisco IOS Release 12.2(18)SXF4 Software Modularity images. This policy reports on low system memory conditions when the amount of memory available falls below 20 percent of the initial available system memory. A syslog message will be displayed and, optionally, an e-mail will be sent.

For more details about the sample policies available and how to run them, see the [“EEM Event Detector Demo: Examples”](#) section on page 37.

## SUMMARY STEPS

1. **enable**
2. **show event manager policy available detailed** *policy-filename*
3. Cut and paste the contents of the sample policy displayed on the screen to a text editor.
4. Edit the policy and save it with a new filename.
5. Copy the new file back to the router flash memory.
6. **configure terminal**
7. **event manager directory user** {*library path* | **policy path**}
8. **event manager policy** *policy-filename* [**type** {*system* | *user*}] [**trap**]

## DETAILED STEPS

### Step 1

#### **enable**

Enables privileged EXEC mode. Enter your password if prompted.

```
Router> enable
```

**Step 2** `show event manager policy available detailed policy-filename`

Displays the actual specified sample policy including details about the environment variables used by the policy and instructions for running the policy. In Cisco IOS 12.2(18)SXF4, the **detailed** keyword was introduced for the **show event manager policy available** and the **show event manager policy registered** commands. In Cisco IOS releases prior to 12.2(18)SXF4, you must copy one of the two Tcl scripts from the configuration examples section in this document (see the “[Programming Policies with Tcl: Sample Scripts Example](#)” section on page 45). In the following example, details about the sample policy `tm_cli_cmd.tcl` are displayed on the screen.

```
Router# show event manager policy available detailed tm_cli_cmd.tcl
```

**Step 3** Cut and paste the contents of the sample policy displayed on the screen to a text editor.

Use the edit and copy functions to move the contents from the router to a text editor on another device.

**Step 4** Edit the policy and save it with a new filename.

Use the text editor to modify the policy as a Tcl script. For file naming conventions, see the “[Cisco File Naming Convention for EEM](#)” section on page 8.

**Step 5** Copy the new file back to the router flash memory.

Copy the file to the flash file system on the router—typically `disk0:`. For more details about copying files, see the “[Using the Cisco IOS File System](#)” chapter in the *Cisco IOS Configuration Fundamentals Configuration Guide*.

**Step 6** `configure terminal`

Enters global configuration mode.

```
Router# configure terminal
```

**Step 7** `event manager directory user {library path | policy path}`

Specifies a directory to use for storing user library files or user-defined EEM policies. In the following example, the `user_library` directory on `disk0` is specified as the directory for storing user library files.

```
Router(config)# event manager directory user library disk0:/user_library
```

**Step 8** `event manager policy policy-filename [type {system | user}] [trap]`

Registers the EEM policy to be run when the specified event defined within the policy occurs. In the following example, the new EEM policy named `test.tcl` is registered as a user-defined policy.

```
Router(config)# event manager policy test.tcl type user
```

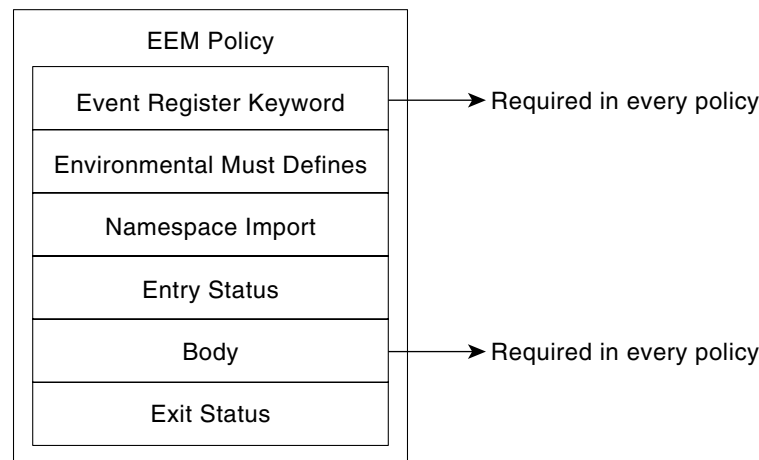
## Programming EEM Policies with Tcl

Perform this task to help you program a policy using Tcl command extensions. We recommend that you copy an existing policy and modify it. There are two required parts that must exist in an EEM Tcl policy: the **event\_register** Tcl command extension and the body. All other sections shown in the “[Tcl Policy Structure and Requirements](#)” concept are optional.

## Tcl Policy Structure and Requirements

All EEM policies share the same structure, shown in [Figure 2](#). There are two parts of an EEM policy that are required: the **event\_register** Tcl command extension and the body. The remaining parts of the policy are optional: environment must defines, namespace import, entry status, and exit status.

**Figure 2** *Tcl Policy Structure and Requirements*



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

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

The environment must defines section is optional and includes the definition of environment variables. The following example Tcl code shows how to check for, and define, some environment variables.

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

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

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

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

- The **event\_reqinfo** event information Tcl command extension that is used to query the EEM for information about the detected event.
- The action Tcl command extensions, such as **action\_syslog**, that are used to specify EEM specific actions.
- The system information Tcl command extensions, such as **sys\_reqinfo\_routename**, 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 **context\_retrieve** Tcl command extensions that are used to save Tcl variables for use by other policies.

The following example Tcl code shows the code to query an event and log a message as part of the body section.

```
# Query the event info and log a message.
array set arr_einfo [event_reqinfo]

if {$_cerrno != 0} {
    set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
        $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}

global timer_type timer_time_sec
set timer_type $arr_einfo(timer_type)
set timer_time_sec $arr_einfo(timer_time_sec)

# Log a message.
set msg [format "timer event: timer type %s, time expired %s" \
    $timer_type [clock format $timer_time_sec]]

action_syslog priority info msg $msg
if {$_cerrno != 0} {
    set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
        $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}
```

## EEM Entry Status

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

Entry status designations may use one of three possible values: 0 (previous policy was successful), Not=0 (previous policy failed), and Undefined (no previous policy was executed).

## EEM Exit Status

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



## EEM Policies and Cisco Error Number

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

For example, the **action\_syslog** command in the example below sets these global variables as a side effect of the command execution:

```
action_syslog priority warning msg "A sample message generated by action_syslog"
if {$_cerrno != 0} {
    set result [format "component=%s; subsystem err=%s; posix err=%s;\n%s" \
        $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}
```

### `_cerrno`: 32-Bit Error Return Values

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

```
XYSSSSSSSSSSSSSEEEEEEEEEPPPPPPPP
```

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

```
862439AE
```

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

```
10000110001001000011100110101110
```

This 32-bit integer is divided up into the five variables shown in [Table 4](#).

**Table 4** `_cerrno`: 32-Bit Error Return Value Variables

Variable	Description
XY	The error class (indicates the severity of the error). This variable corresponds to the first two bits in the 32-bit error return value; 10 in the case above, which indicates CERR_CLASS_WARNING:  See <a href="#">Table 5</a> 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 <code>\$_cerr_sub_num</code> .
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 <code>\$_cerr_sub_err</code> .
PPPPPPPP	The pass-through POSIX error code (9 bits = 512 values). This represents the last of the 32-bit sequence, and the string corresponding to this error code is contained in <code>\$_cerr_posix_err</code> .

### Error Class Encodings for XY

The first variable, XY, references the possible error class encodings shown in [Table 5](#).

**Table 5**      **Error Class Encodings**

00	CERR_CLASS_SUCCESS
01	CERR_CLASS_INFO
10	CERR_CLASS_WARNING
11	CERR_CLASS_FATAL

An error return value of zero means SUCCESS.

## SUMMARY STEPS

1. **enable**
2. **show event manager policy available detailed** *policy-filename*
3. Cut and paste the contents of the sample policy displayed on the screen to a text editor.
4. Define the required **event\_register** Tcl command extension.
5. Add the appropriate namespace under the `::cisco` hierarchy.
6. Program the `must defines` section to check for each environment variable that is used in this policy.
7. Program the body of the script.
8. Check the entry status to determine if a policy has previously run for this event.
9. Check the exit status to determine whether or not to apply the default action for this event, if a default action exists.
10. Set Cisco Error Number (`_cerno`) Tcl global variables.
11. Save the Tcl script with a new filename, and copy the Tcl script to the router.
12. **configure terminal**
13. **event manager directory user** {*library path* | *policy path*}
14. **event manager policy** *policy-filename* [**type** {**system** | **user**}] [**trap**]
15. Cause the policy to execute, and observe the policy.
16. Use debugging techniques if the policy does not execute correctly.

## DETAILED STEPS

---

### Step 1 **enable**

Enables privileged EXEC mode. Enter your password if prompted.

```
Router> enable
```

### Step 2 **show event manager policy available detailed** *policy-filename*

Displays the actual specified sample policy including details about the environment variables used by the policy and instructions for running the policy. In Cisco IOS 12.2(18)SXF4, the **detailed** keyword was introduced for the **show event manager policy available** and the **show event manager policy registered** commands. In Cisco IOS releases prior to 12.2(18)SXF4, you must copy one of the two Tcl scripts from the configuration examples section in this document (see the [“Programming Policies with Tcl: Sample Scripts Example”](#) section on page 45). In the following example, details about the sample policy `tm_cli_cmd.tcl` are displayed on the screen.

```
Router# show event manager policy available detailed tm_cli_cmd.tcl
```

- Step 3** Cut and paste the contents of the sample policy displayed on the screen to a text editor. Use the edit and copy functions to move the contents from the router to a text editor on another device. Use the text editor to edit the policy as a Tcl script.
- Step 4** Define the required **event\_register** Tcl command extension. Choose the appropriate **event\_register** Tcl command extension from [Table 6](#) for the event that you want to detect, and add it to the policy.

**Table 6** EEM Event Registration Tcl Command Extensions

---

**Event Registration Tcl Command Extensions**

---

event_register_appl
event_register_cli
event_register_counter
event_register_gold
event_register_interface
event_register_ioswdsysmon
event_register_ipsla
event_register_nf
event_register_none
event_register_oir
event_register_process
event_register_resource
event_register_rf
event_register_routing
event_register_rpc
event_register_snmp
event_register_snmp_notification
event_register_snmp_object
event_register_syslog
event_register_timer
event_register_timer_subscriber
event_register_track
event_register_wdsysmon

---

- Step 5** Add the appropriate namespace under the `::cisco` hierarchy. Policy developers can use the new namespace `::cisco` in Tcl policies in order to group all the extensions used by Cisco IOS EEM. There are two namespaces under the `::cisco` hierarchy, and [Table 7](#) shows which category of EEM Tcl command extension belongs under each namespace.

**Table 7** Cisco IOS EEM Namespace Groupings

Namespace	Category of Tcl Command Extension
::cisco::eem	EEM event registration
	EEM event information
	EEM event publish
	EEM action
	EEM utility
	EEM context library
	EEM system information
	CLI library
::cisco::lib	SMTP library



**Note** Make sure that you import the appropriate namespaces or use the qualified command names when using the above commands.

- Step 6** Program the `must defines` section to check for each environment variable that is used in this policy. This is an optional step. `Must defines` are a section of the policy that tests whether any EEM environment variables that are required by the policy are defined before the recovery actions are taken. The `must defines` section is not required if the policy does not use any EEM environment variables. EEM environment variables for EEM scripts are Tcl global variables that are defined external to the policy before the policy is run. To define an EEM environment variable, use the Embedded Event Manager configuration command **event manager environment** CLI command. By convention all Cisco EEM environment variables begin with “\_” (an underscore). In order to avoid future conflict, customers are urged not to define new variables that start with “\_”.



**Note** You can display the Embedded Event Manager environment variables set on your system by using the **show event manager environment** privileged EXEC command.

For example, Embedded Event Manager environment variables defined by the sample policies include e-mail variables. The sample policies that send e-mail must have the variables shown in [Table 8](#) set in order to function properly.

Table 8 describes the e-mail-specific environment variables used in the sample EEM policies.

**Table 8 E-mail-Specific Environmental Variables Used by the Sample Policies**

Environment Variable	Description	Example
_email_server	A Simple Mail Transfer Protocol (SMTP) mail server used to send e-mail.	The e-mail server name can be in any one of the following template formats: <ul style="list-style-type: none"> <li>• username:password@host</li> <li>• username@host</li> <li>• host</li> </ul>
_email_to	The address to which e-mail is sent.	engineering@example.com
_email_from	The address from which e-mail is sent.	devtest@example.com
_email_cc	The address to which the e-mail must be copied.	manager@example.com

The following example of a must define section shows how to program a check for e-mail-specific environment variables.

**Example 1 Example of Must Defines**

```

if {[info exists _email_server]} {
    set result \
        "Policy cannot be run: variable _email_server has not been set"
    error $result $errorMsg
}
if {[info exists _email_from]} {
    set result \
        "Policy cannot be run: variable _email_from has not been set"
    error $result $errorMsg
}
if {[info exists _email_to]} {
    set result \
        "Policy cannot be run: variable _email_to has not been set"
    error $result $errorMsg
}
if {[info exists _email_cc]} {
    set result \
        "Policy cannot be run: variable _email_cc has not been set"
    error $result $errorMsg
}
    
```

**Step 7** Program the body of the script.

In this section of the script, you can define any of the following:

- The **event\_reqinfo** event information Tcl command extension that is used to query the EEM for information about the detected event.
- The action Tcl command extensions, such as **action\_syslog**, that are used to specify EEM specific actions.
- The system information Tcl command extensions, such as **sys\_reqinfo\_routename**, that are used to obtain general system information.

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

**Step 8** Check the entry status to determine if a policy has previously run for this event.

If the prior policy is successful, the current policy may or may not require execution. Entry status designations may use one of three possible values: 0 (previous policy was successful), Not=0 (previous policy failed), and Undefined (no previous policy was executed).

**Step 9** Check the exit status to determine whether or not to apply the default action for this event, if a default action exists.

A value of zero means do not perform the default action. A value of nonzero means perform the default action. The exit status will be passed to subsequent policies that are run for the same event.

**Step 10** Set Cisco Error Number (`_cerrno`) Tcl global variables.

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

For example, the **action\_syslog** command in the example below sets these global variables as a side effect of the command execution:

```
action_syslog priority warning msg "A sample message generated by action_syslog
if {$_cerrno != 0} {
    set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
        $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}
```

**Step 11** Save the Tcl script with a new filename, and copy the Tcl script to the router.

Embedded Event Manager policy filenames adhere to the following specification:

- An optional prefix—Mandatory.—indicating, if present, that this is a system policy that should be registered automatically at boot time if it is not already registered. For example: Mandatory.sl\_text.tcl.
- A filename body part containing a two-character abbreviation (see [Table 2 on page 8](#)) for the first event specified; an underscore character part; and a descriptive field part further identifying the policy.
- A filename suffix part defined as .tcl.

For more details, see the “Cisco File Naming Convention for EEM” section on page 8.

Copy the file to the flash file system on the router—typically `disk0:`. For more details about copying files, see the “Using the Cisco IOS File System” chapter in the *Cisco IOS Configuration Fundamentals Configuration Guide*.

**Step 12** **configure terminal**

Enters global configuration mode.

```
Router# configure terminal
```

**Step 13** **event manager directory user {library path | policy path}**

Specifies a directory to use for storing user library files or user-defined EEM policies. In the following example, the `user_library` directory on `disk0` is specified as the directory for storing user library files.

```
Router(config)# event manager directory user library disk0:/user_library
```

**Step 14** `event manager policy policy-filename [type {system | user}] [trap]`

Registers the EEM policy to be run when the specified event defined within the policy occurs. In the following example, the new EEM policy named `cl_mytest.tcl` is registered as a user-defined policy.

```
Router(config)# event manager policy cl_mytest.tcl type user
```

**Step 15** Cause the policy to execute, and observe the policy.

To test that the policy runs, generate the conditions that will cause the policy to execute and observe that the policy runs as expected.

**Step 16** Use debugging techniques if the policy does not execute correctly.

Use the Cisco IOS **debug event manager** CLI command with its various keywords to debug issues. Refer to the [“Troubleshooting Tips” section on page 31](#) for details about using Tcl-specific keywords.

---

## Troubleshooting Tips

- Use the **debug event manager tcl commands** CLI command to debug issues with Tcl extension commands. When enabled, this command displays all data that is passed in and read back from the TTY session that handles the CLI interactions. This data helps ensure users that the commands they are passing to the CLI are valid.
- The CLI library allows users to run CLI commands and obtain the output of commands in Tcl. Use the **debug event manager tcl cli-library** CLI command to debug issues with the CLI library.
- The SMTP library allows users to send e-mail messages to an SMTP e-mail server. Use the **debug event manager tcl smtp\_library** CLI command to debug issues with the SMTP library. When enabled, this command displays all data that is passed in and read back from the SMTP library routines. This data helps ensure users that the commands they are passing to the SMTP library are valid.
- Tcl is a flexible language that allows you to override commands. For example, you can modify the **set** command and create a version of the **set** command that displays a message when a scalar variable is set. When the **set** command is entered in a policy, a message is displayed anytime a scalar variable is set, and this provides a way to debug scalar variables. To view an example of this debugging technique, see the [“Tracing Tcl set Command Operations: Example” section on page 57](#).

To view examples of the some of these debugging techniques, see the [“Debugging Embedded Event Manager Policies: Examples” section on page 55](#).

## Creating an EEM User Tcl Library Index

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

### SUMMARY STEPS

1. On your workstation (UNIX, Linux, PC, or Mac) create a library directory and copy the Tcl library files into the directory.

2. **tclsh**
3. **auto\_mkindex** *directory\_name \*.tcl*
4. Copy the Tcl library files from [Step 1](#) and the tclIndex file from [Step 3](#) to the directory used for storing user library files on the target router.
5. Copy a user-defined EEM policy file written in Tcl to the directory used for storing user-defined EEM policies on the target router.
6. **enable**
7. **configure terminal**
8. **event manager directory user library** *path*
9. **event manager directory user policy** *path*
10. **event manager policy** *policy-filename* [**type** {system | user}] [**trap**]
11. **event manager run** *policy-filename*

## DETAILED STEPS

**Step 1** On your workstation (UNIX, Linux, PC, or Mac) create a library directory and copy the Tcl library files into the directory.

The following example files can be used to create a tclIndex on a workstation running the Tcl shell:

### lib1.tcl

```
proc test1 {} {
    puts "In procedure test1"
}

proc test2 {} {
    puts "In procedure test2"
}
```

### lib2.tcl

```
proc test3 {} {
    puts "In procedure test3"
}
```

**Step 2** **tclsh**

Use this command to enter the Tcl shell.

```
workstation% tclsh
```

**Step 3** **auto\_mkindex** *directory\_name \*.tcl*

Use the **auto\_mkindex** command to create the tclIndex file. The tclIndex file that contains a directory of all the procedures contained in the Tcl library files. We recommend that you run **auto\_mkindex** inside a directory because there can only be a single tclIndex file in any directory and you may have other Tcl files to be grouped together. Running **auto\_mkindex** in a directory determines which tcl source file or files are indexed using a specific tclIndex.

```
workstation% auto_mkindex eem_library *.tcl
```

The following example TclIndex is created when the lib1.tcl and lib2.tcl files are in a library file directory and the **auto\_mkindex** command is run.



**tclIndex**

```
# Tcl autoload index file, version 2.0
# This file is generated by the "auto_mkindex" command
# and sourced to set up indexing information for one or
# more commands. Typically each line is a command that
# sets an element in the auto_index array, where the
# element name is the name of a command and the value is
# a script that loads the command.

set auto_index(test1) [list source [file join $dir lib1.tcl]]
set auto_index(test2) [list source [file join $dir lib1.tcl]]
set auto_index(test3) [list source [file join $dir lib2.tcl]]
```

- Step 4** Copy the Tcl library files from [Step 1](#) and the tclIndex file from [Step 3](#) to the directory used for storing user library files on the target router.
- Step 5** Copy a user-defined EEM policy file written in Tcl to the directory used for storing user-defined EEM policies on the target router. The directory can be the same directory used in [Step 4](#).

The directory for storing user-defined EEM policies can be the same directory used in [Step 4](#). The following example user-defined EEM policy can be used to test the Tcl library support in EEM.

**libtest.tcl**

```
::cisco::eem::event_register_none

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

global auto_index auto_path

puts [array names auto_index]

if { [catch {test1} result]} {
    puts "calling test1 failed result = $result $auto_path"
}

if { [catch {test2} result]} {
    puts "calling test2 failed result = $result $auto_path"
}

if { [catch {test3} result]} {
    puts "calling test3 failed result = $result $auto_path"
}
```

- Step 6** **enable**  
Enables privileged EXEC mode. Enter your password if prompted.

```
Router> enable
```

- Step 7** **configure terminal**  
Enables global configuration mode.

```
Router# configure terminal
```

- Step 8** **event manager directory user library path**

Use this command to specify the EEM user library directory; this is the directory to which the files in [Step 4](#) were copied.

```
router(config)# event manager directory user library disk2:/eem_library
```

**Step 9** event manager directory user policy *path*

Use this command to specify the EEM user policy directory; this is the directory to which the file in [Step 5](#) was copied.

```
router(config)# event manager directory user policy disk2:/eem_policies
```

**Step 10** event manager policy *policy-name* [type {system | user}] [trap]

Use this command to register a user-defined EEM policy. In this example, the policy named libtest.tcl is registered.

```
router(config)# event manager policy libtest.tcl
```

**Step 11** event manager run *policy-name*

Use this command to manually run an EEM policy. In this example, the policy named libtest.tcl is run to test the Tcl support in EEM. The example output shows that the test for Tcl support in EEM was successful.

```
router(config)# event manager run libtest.tcl
```

The following output is displayed:

```
01:24:37: %HA_EM-6-LOG: libtest.tcl: In procedure test1
01:24:37: %HA_EM-6-LOG: libtest.tcl: In procedure test2
01:24:37: %HA_EM-6-LOG: libtest.tcl: In procedure test3
```

## Creating an EEM User Tcl Package Index

Perform this task to create a Tcl package index file that contains a directory of all the Tcl packages and version information contained in a library of Tcl package files. Tcl packages are supported using the Tcl **package** keyword, and this support was introduced in Cisco IOS Release 12.4(11)T.

Tcl packages are located in either the EEM system library directory or the EEM user library directory. When a **package require** Tcl command is executed, the user library directory is searched first for a pkgIndex.tcl file. If the pkgIndex.tcl file is not found in the user directory, the system library directory is searched. In this task, a Tcl package directory—the pkgIndex.tcl file—is created in the appropriate library directory using the **pkg\_mkIndex** command to contain information about all of the Tcl packages contained in the directory along with version information. If the index is not created, the Tcl packages will not be found when an EEM policy is run that contains a **package require** Tcl command.

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

**Note**

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

### SUMMARY STEPS

1. On your workstation (UNIX, Linux, PC, or Mac) create a library directory and copy the Tcl package files into the directory.
2. **tclsh**
3. **pkg\_mkIndex** *directory\_name* \*.tcl

4. Copy the Tcl package files from [Step 1](#) and the pkgIndex file from [Step 3](#) to the directory used for storing user library files on the target router.
5. Copy a user-defined EEM policy file written in Tcl to the directory used for storing user-defined EEM policies on the target router.
6. **enable**
7. **configure terminal**
8. **event manager directory user library *path***
9. **event manager directory user policy *path***
10. **event manager policy *policy-filename* [type {system | user}] [trap]**
11. **event manager run *policy-filename***

## DETAILED STEPS

**Step 1** On your workstation (UNIX, Linux, PC, or Mac) create a library directory and copy the Tcl package files into the directory.

**Step 2** **tclsh**

Use this command to enter the Tcl shell.

```
workstation% tclsh
```

**Step 3** **pkg\_mkindex *directory\_name* \*.tcl**

Use the **pkg\_mkindex** command to create the pkgIndex file. The pkgIndex file contains a directory of all the packages contained in the Tcl library files. We recommend that you run pkg\_mkindex inside a directory because there can only be a single pkgIndex file in any directory and you may have other Tcl files to be grouped together. Running pkg\_mkindex in a directory determines which Tcl package file or files are indexed using a specific pkgIndex.

```
workstation% pkg_mkindex eem_library *.tcl
```

The following example pkgIndex is created when some Tcl package files are in a library file directory and the **pkg\_mkindex** command is run.

### pkgIndex

```
# Tcl package index file, version 1.1
# This file is generated by the "pkg_mkIndex" command
# and sourced either when an application starts up or
# by a "package unknown" script. It invokes the
# "package ifneeded" command to set up package-related
# information so that packages will be loaded automatically
# in response to "package require" commands. When this
# script is sourced, the variable $dir must contain the
# full path name of this file's directory.

package ifneeded xmlrpc 0.3 [list source [file join $dir xmlrpc.tcl]]
```

**Step 4** Copy the Tcl library files from [Step 1](#) and the pkgIndex file from [Step 3](#) to the directory used for storing user library files on the target router.

**Step 5** Copy a user-defined EEM policy file written in Tcl to the directory used for storing user-defined EEM policies on the target router. The directory can be the same directory used in [Step 4](#).

The directory for storing user-defined EEM policies can be the same directory used in [Step 4](#). The following example user-defined EEM policy can be used to test the Tcl package support in EEM.

**packagetest.tcl**

```
::cisco::eem::event_register_none maxrun 1000000.000
#
# test if xmlrpc available
#
#
# Namespace imports
#
namespace import ::cisco::eem::*
namespace import ::cisco::lib::*
#
package require xmlrpc
puts "Did you get an error?"
```

**Step 6** **enable**

Enables privileged EXEC mode. Enter your password if prompted.

```
Router> enable
```

**Step 7** **configure terminal**

Enables global configuration mode.

```
Router# configure terminal
```

**Step 8** **event manager directory user library path**

Use this command to specify the EEM user library directory; this is the directory to which the files in [Step 4](#) were copied.

```
router(config)# event manager directory user library disk2:/eem_library
```

**Step 9** **event manager directory user policy path**

Use this command to specify the EEM user policy directory; this is the directory to which the file in [Step 5](#) was copied.

```
router(config)# event manager directory user policy disk2:/eem_policies
```

**Step 10** **event manager policy policy-name [type {system | user} [trap]**

Use this command to register a user-defined EEM policy. In this example, the policy named packagetest.tcl is registered.

```
router(config)# event manager policy packagetest.tcl
```

**Step 11** **event manager run policy-name**

Use this command to manually run an EEM policy. In this example, the policy named packagetest.tcl is run to test the Tcl package support in EEM.

```
router(config)# event manager run packagetest.tcl
```

# Configuration Examples for Writing Embedded Event Manager Policies Using Tcl

This section contains the following configuration examples:

- [Assigning a Username for a Tcl Session: Examples, page 37](#)
- [EEM Event Detector Demo: Examples, page 37](#)
- [Programming Policies with Tcl: Sample Scripts Example, page 45](#)
- [Debugging Embedded Event Manager Policies: Examples, page 55](#)
- [Tracing Tcl set Command Operations: Example, page 57](#)
- [RPC Event Detector: Example, page 57](#)

## Assigning a Username for a Tcl Session: Examples

The following example shows how to set a username to be associated with a Tcl session. If you are using authentication, authorization, and accounting (AAA) security and implement authorization on a command basis, you should use the **event manager session cli username** command to set a username to be associated with a Tcl session. The username is used when a Tcl policy executes a CLI command. TACACS+ verifies each CLI command using the username associated with the Tcl session that is running the policy. Commands from Tcl policies are not usually verified because the router must be in privileged EXEC mode to register the policy. In the example, the username is yourname, and this is the username that is used whenever a CLI command session is initiated from within an EEM policy.

```
configure terminal
event manager session cli username yourname
end
```

## EEM Event Detector Demo: Examples

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

- [EEM Sample Policy Descriptions](#)
- [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 some of the sample EEM policies:

- `ap_perf_test_base_cpu.tcl`—Is run to measure the CPU performance of EEM policies.
- `no_perf_test_init.tcl`—Is run to measure the CPU performance of EEM policies.
- `sl_intf_down.tcl`—Is run when a configurable syslog message is logged. It executes up to two configurable CLI commands and e-mails the results.

- `tm_cli_cmd.tcl`—Is run using a configurable CRON entry. It executes a configurable CLI command and e-mails the results.
- `tm_crash_reporter.tcl`—Introduced in Cisco IOS Release 12.4(2)T, 12.2(31)SB3, and 12.2(33)SRB. Is run 5 seconds after it is registered and 5 seconds after the router boots up. When triggered, the script attempts to find the reload reason. If the reload reason was due to a crash, the policy searches for the related crashinfo file and sends this information to a URL location specified by the user in the environment variable `_crash_reporter_url`.
- `tm_fsys_usage.tcl`—Introduced in Cisco IOS Release 12.2(18)SXF4 Cisco IOS Software Modularity images. This policy runs using a configurable CRON entry and monitors disk space usage. A syslog message is displayed if disk space usage crosses configurable thresholds.

### Event Manager Environment Variables for the Sample Policies

Event manager environment variables are Tcl global variables that are defined external to the EEM policy before the policy is registered and run. The sample policies require three of the e-mail environment variables to be set (see [Table 8 on page 29](#) for a list of the e-mail variables); only `_email_cc` is optional. Other required and optional variable settings are outlined in the following tables.

[Table 9](#) describes the EEM environment variables that must be set before the `ap_perf_test_base_cpu.tcl` sample policy is run.

**Table 9 Environment Variables Used in the `ap_perf_test_base_cpu.tcl` Policy**

Environment Variable	Description	Example
<code>_perf_iterations</code>	The number of iterations over which to run the measurement.	<b>100</b>
<code>_perf_cmd1</code>	The first non interactive CLI command that is executed as part of the measurement test. This variable is optional and need not be specified.	<b>enable</b>
<code>_perf_cmd2</code>	The second non interactive CLI command that is as part of the measurement test. To use <code>_perf_cmd2</code> , <code>_perf_cmd1</code> must be defined. This variable is optional and need not be specified.	<b>show version</b>
<code>_perf_cmd3</code>	The third non interactive CLI command that is as part of the measurement test. To use <code>_perf_cmd3</code> , <code>_perf_cmd1</code> must be defined. This variable is optional and need not be specified.	<b>show interface counters protocol status</b>

Table 10 describes the EEM environment variables that must be set before the `no_perf_test_init.tcl` sample policy is run.

**Table 10** Environment Variables Used in the `no_perf_test_init.tcl` Policy

Environment Variable	Description	Example
<code>_perf_iterations</code>	The number of iterations over which to run the measurement.	<b>100</b>
<code>_perf_cmd1</code>	The first non interactive CLI command that is executed as part of the measurement test. This variable is optional and need not be specified.	<b>enable</b>
<code>_perf_cmd2</code>	The second non interactive CLI command that is as part of the measurement test. To use <code>_perf_cmd2</code> , <code>_perf_cmd1</code> must be defined. This variable is optional and need not be specified.	<b>show version</b>
<code>_perf_cmd3</code>	The third non interactive CLI command that is as part of the measurement test. To use <code>_perf_cmd3</code> , <code>_perf_cmd1</code> must be defined. This variable is optional and need not be specified.	<b>show interface counters protocol status</b>

Table 11 describes the EEM environment variables that must be set before the `sl_intf_down.tcl` sample policy is run.

**Table 11** Environment Variables Used in the `sl_intf_down.tcl` Policy

Environment Variable	Description	Example
<code>_config_cmd1</code>	The first configuration command that is executed.	<b>interface Ethernet1/0</b>
<code>_config_cmd2</code>	The second configuration command that is executed. This variable is optional and need not be specified.	<b>no shutdown</b>
<code>_syslog_pattern</code>	A regular expression pattern match string that is used to compare syslog messages to determine when the policy runs.	<b>. *UPDOWN.*FastEthernet0/0.*</b>

Table 12 describes the EEM environment variables that must be set before the `tm_cli_cmd.tcl` sample policy is run.

**Table 12** Environment Variables Used in the `tm_cli_cmd.tcl` Policy

Environment Variable	Description	Example
<code>_cron_entry</code>	A CRON specification that determines when the policy will run.	<b>0-59/1 0-23/1 * * 0-7</b>
<code>_show_cmd</code>	The CLI command to be executed when the policy is run.	<b>show version</b>

Table 13 describes the EEM environment variables that must be set before the `tm_crash_reporter.tcl` sample policy is run.

**Table 13** *Environment Variables Used in the `tm_crash_reporter.tcl` Policy*

Environment Variable	Description	Example
<code>_crash_reporter_debug</code>	A value that identifies whether debug information for <code>tm_crash_reporter.tcl</code> will be enabled. This variable is optional and need not be specified.	1
<code>_crash_reporter_url</code>	The URL location to which the crash report is sent.	<code>http://www.example.com/fm/interface_tm.cgi</code>

Table 14 describes the EEM environment variables that must be set before the `tm_fsys_usage.tcl` sample policy is run.

**Table 14** *Environment Variables Used in the `tm_fsys_usage.tcl` Policy*

Environment Variable	Description	Example
<code>_tm_fsys_usage_cron</code>	A CRON specification that is used in the <code>event_register</code> Tcl command extension. If unspecified, the <code>tm_fsys_usage.tcl</code> policy is triggered once per minute. This variable is optional and need not be specified.	<code>0-59/1 0-23/1 * * 0-7</code>
<code>_tm_fsys_usage_debug</code>	When this variable is set to a value of 1, disk usage information is displayed for all entries in the system. This variable is optional and need not be specified.	1
<code>_tm_fsys_usage_freebytes</code>	Free byte threshold for systems or specific prefixes. If free space falls below a given value, a warning is displayed. This variable is optional and need not be specified.	<code>disk2:98000000</code>
<code>_tm_fsys_usage_percent</code>	Disk usage percentage thresholds for systems or specific prefixes. If the disk usage percentage exceeds a given percentage, a warning is displayed. If unspecified, the default disk usage percentage is 80 percent for all systems. This variable is optional and need not be specified.	<code>nvrnm:25 disk2:5</code>

### Registration of Some EEM Policies

Some EEM policies must be unregistered and then reregistered if an EEM environment variable is modified after the policy is registered. The `event_register_xxx` statement that appears at the start of the policy contains some of the EEM environment variables, and this statement is used to establish the conditions under which the policy is run. If the environment variables are modified after the policy has been registered, the conditions may become invalid. To avoid any errors, the policy must be unregistered and then reregistered. The following variables are affected:



- `_cron_entry` in the `tm_cli_cmd.tcl` policy
- `_syslog_pattern` in the `sl_intf_down.tcl` policy

### Basic Configuration Details for All Sample Policies

To allow e-mail to be sent from the Embedded Event Manager, the `hostname` and `ip domain-name` commands must be configured. The EEM environment variables must also be set. After a Cisco IOS image has been booted, use the following initial configuration, substituting appropriate values for your network. The environment variables for the `tm_fsyz_usage` sample policy (see [Table 14 on page 40](#)) are all optional and are not listed here:

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

### Using the Sample Policies

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

- [Running the Mandatory.go\\_\\*.tcl Sample Policy, page 41](#)
- [Running the ap\\_perf\\_test\\_base\\_cpu.tcl and no\\_perf\\_test\\_init.tcl Sample Policies, page 42](#)
- [Running the sl\\_intf\\_down.tcl Sample Policy, page 43](#)
- [Running the tm\\_cli\\_cmd.tcl Sample Policy, page 44](#)
- [Running the tm\\_crash\\_reporter.tcl Sample Policy, page 44](#)
- [Running the tm\\_fsyz\\_usage.tcl Sample Policy, page 45](#)

### Running the Mandatory.go\_\*.tcl Sample Policy

There are GOLD TCL scripts for each test which runs as a part of GOLD EEM Policy. You can modify the TCL script for the test, specify the consecutive failure count, and also change the default corrective action. For example, one could chose to power down a linecard card, instead of reset or other CLI based actions.

For each registered test, a default TCL script is available, which can be registered with the system, and matches with the default action. This can be then overridden by modifying these scripts.

The following table shows a list of the mandatory polices that GOLD installed into EEM. Each of the policies performs some sort of action such as resetting the card or disabling the port.

GOLD Tcl Scripts	Test
Mandatory.go_asicsync.tcl	TestAsicSync
Mandatory.go_bootup.tcl	Common for all bootup tests.
Mandatory.go_fabric.tcl	TestFabricHealth

GOLD Tcl Scripts	Test
Mandatory.go_fabrich0.tcl	TestFabricCh0Health
Mandatory.go_fabrich1.tcl	TestFabricCh1Health
Mandatory.go_ipsec.tcl	TestIPSecEncrypDecrypPkt
Mandatory.go_mac.tcl	TestMacNotification
Mandatory.go_nondislp.tcl	TestNonDisruptiveLoopback
Mandatory.go_scratchreg.tcl	TestScratchRegister
Mandatory.go_sprping.tcl	TestSPRPInbandPing

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 mandatory.go\_\*.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.

```
enable
show event manager policy registered
show event manager policy available
configure terminal
  event manager policy Mandatory.go_spuriousisr.tcl
end
show event manager policy registered
show event manager environment
```

#### Running the ap\_perf\_test\_base\_cpu.tcl and no\_perf\_test\_init.tcl Sample Policies

These sample policies measure the CPU performance of EEM policies. The policies help find the average execution time of each EEM policy and use the CLI library to execute the configuration commands specified in the EEM environment variables `_perf_cmd1` and, optionally, `_perf_cmd2` and `_perf_cmd3`.

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, enter the **service timestamps debug datetime msec** command and then you can register the `ap_perf_test_base_cpu.tcl` and `no_perf_test_init.tcl` policies 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 policies `ap_perf_test_base_cpu.tcl` and `no_perf_test_init.tcl` need to be registered together, as they run as a test suite. You can run the `no_perf_test_init.tcl` policy to start the tests. Analyze the results using the syslog messages from each iteration. The total number of iterations is specified by the variable `_perf_iterations`. Take the time difference and divide it by the total number of iterations to get the average execution time of each EEM policy.

```
enable
show event manager policy registered
show event manager policy available
show event manager environment
configure terminal
```

```
service timestamps debug datetime msec
event manager environment _perf_iterations 100
event manager policy ap_perf_test_base_cpu.tcl
event manager policy no_perf_test_init.tcl
end
show event manager policy registered
show event manager policy available
show event manager environment
event manager run no_perf_test_init.tcl
```

### Running the no\_perf\_test\_init.tcl Sample Policy

This sample policy measures the the cpu performance of EEM policies. The policy helps to find the average execution time of each EEM policy and uses the CLI library to execute the configuration commands specified in the EEM environment variables `_perf_cmd1` and, optionally, `_perf_cmd2` and `_perf_cmd3`.

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 `no_perf_test_init.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.

Analyze the results using the syslog messages from each iteration. The total number of iteration is specified by the variable `_perf_iterations`. Take the time difference and divide it by the total number of iterations to get the average execution time of each EEM policy.

```
enable
show event manager policy registered
show event manager policy available
configure terminal
 event manager policy no_perf_test_init.tcl
end
show event manager policy registered
show event manager environment
```

### Running the sl\_intf\_down.tcl Sample Policy

This sample policy demonstrates the ability to modify the configuration when a syslog message with a specific pattern is logged. The policy gathers detailed information about the event and uses the CLI library to execute the configuration commands specified in the EEM environment variables `_config_cmd1` and, optionally, `_config_cmd2`. An e-mail message is sent with the results of the CLI command.

The following sample configuration demonstrates how to use this policy. Starting in user EXEC mode, enter the **enable** command at the router prompt. The router enters privileged EXEC mode, where you can enter the **show event manager policy registered** command to verify that no policies are currently registered. The next command is the **show event manager policy available** command to display which policies are available to be installed. After you enter the **configure terminal** command to reach global configuration mode, you can register the `sl_intf_down.tcl` policy with EEM using the **event manager policy** command. Exit from global configuration mode and enter the **show event manager policy registered** command again to verify that the policy has been registered.

The policy runs when an interface goes down. Enter the **show event manager environment** command to display the current environment variable values. Unplug the cable (or configure a shutdown) for the interface specified in the `_syslog_pattern` EEM environment variable. The interface goes down, prompting the syslog daemon to log a syslog message about the interface being down, and the syslog event detector is called.

The syslog event detector reviews the outstanding event specifications and finds a match for interface status change. The EEM server is notified, and the server runs the policy that is registered to handle this event—`sl_intf_down.tcl`.

```
enable
show event manager policy registered
show event manager policy available
configure terminal
  event manager policy sl_intf_down.tcl
end
show event manager policy registered
show event manager environment
```

### Running the `tm_cli_cmd.tcl` Sample Policy

This sample policy demonstrates the ability to periodically execute a CLI command and to e-mail the results. The CRON specification “0-59/2 0-23/1 \* \* 0-7” causes this policy to be run on the second minute of each hour. The policy gathers detailed information about the event and uses the CLI library to execute the configuration commands specified in the EEM environment variable `_show_cmd`. An e-mail message is sent with the results of the CLI command.

The following sample configuration demonstrates how to use this policy. Starting in user EXEC mode, enter the **enable** command at the router prompt. The router enters privileged EXEC mode where you can enter the **show event manager policy registered** command to verify that no policies are currently registered. The next command is the **show event manager policy available** command to display which policies are available to be installed. After you enter the **configure terminal** command to reach global configuration mode, you can register the `tm_cli_cmd.tcl` policy with EEM using the **event manager policy** command. Exit from global configuration mode and enter the **show event manager policy registered** command to verify that the policy has been registered.

The timer event detector triggers an event for this case periodically according to the CRON string set in the EEM environment variable `_cron_entry`. The EEM server is notified, and the server runs the policy that is registered to handle this event—`tm_cli_cmd.tcl`.

```
enable
show event manager policy registered
show event manager policy available
configure terminal
  event manager policy tm_cli_cmd.tcl
end
show event manager policy registered
```

### Running the `tm_crash_reporter.tcl` Sample Policy

This sample policy demonstrates the ability to send an HTTP-formatted crash report to a URL location. If the policy registration is saved in the startup configuration file, the policy is triggered 5 seconds after bootup. When triggered, the script attempts to find the reload reason. If the reload reason was due to a crash, the policy searches for the related crashinfo file and sends this information to a URL location specified by the user in the environment variable `_crash_reporter_url`. A CGI script, `interface_tm.cgi`, has been created to receive the URL from the `tm_crash_reporter.tcl` policy and save the crash information in a local database on the target URL machine.

A Perl CGI script, `interface_tm.cgi`, has been created and is designed to run on a machine that contains an HTTP server and is accessible by the router that runs the `tm_crash_reporter.tcl` policy. The `interface_tm.cgi` script parses the data passed into it from `tm_crash_reporter.tcl` and appends the crash information to a text file, creating a history of all crashes in the system. Additionally, detailed information on each crash is stored in three files in a crash database directory that is specified by the user. Another Perl CGI script, `crash_report_display.cgi`, has been created to display the information stored in the database created by the `interface_tm.cgi` script. The `crash_report_display.cgi` script should be placed on the same machine that contains `interface_tm.cgi`. The machine should be running a web browser such as Internet Explorer or Netscape. When the `crash_report_display.cgi` script is run, it displays the crash information in a readable format.

The following sample configuration demonstrates how to use this policy. Starting in user EXEC mode, enter the **enable** command at the router prompt. The router enters privileged EXEC mode where you can enter the **show event manager policy registered** command to verify that no policies are currently registered. The next command is the **show event manager policy available** command to display which policies are available to be installed. After you enter the **configure terminal** command to reach global configuration mode, you can register the `tm_crash_reporter.tcl` policy with EEM using the **event manager policy** command. Exit from global configuration mode and enter the **show event manager policy registered** command to verify that the policy has been registered.

```
enable
show event manager policy registered
show event manager policy available
configure terminal
  event manager policy tm_crash_reporter.tcl
end
show event manager policy registered
```

### Running the `tm_fsys_usage.tcl` Sample Policy

This sample policy demonstrates the ability to periodically monitor disk space usage and report through syslog when configurable thresholds have been crossed.

The following sample configuration demonstrates how to use this policy. Starting in user EXEC mode, enter the **enable** command at the router prompt. The router enters privileged EXEC mode, where you can enter the **show event manager policy registered** command to verify that no policies are currently registered. The next command is the **show event manager policy available** command to display which policies are available to be installed. After you enter the **configure terminal** command to reach global configuration mode, you can register the `tm_fsys_usage.tcl` policy with EEM using the **event manager policy** command. Exit from global configuration mode and enter the **show event manager policy registered** command again to verify that the policy has been registered. If you had configured any of the optional environment variables that are used in the `tm_fsys_usage.tcl` policy, the **show event manager environment** command displays the configured variables.

```
enable
show event manager policy registered
show event manager policy available
configure terminal
  event manager policy tm_fsys_usage.tcl
end
show event manager policy registered
show event manager environment
```

## Programming Policies with Tcl: Sample Scripts Example

This section contains some of the sample policies that are included as EEM system policies. For more details about these policies, see the [“EEM Event Detector Demo: Examples”](#) section on page 37.

- [Mandatory.go\\_ipsec.tcl Sample Policy, page 46](#)
- [ap\\_perf\\_test\\_base\\_cpu.tcl Sample Policy, page 47](#)
- [tm\\_cli\\_cmd.tcl Sample Policy, page 50](#)
- [sl\\_intf\\_down.tcl Sample Policy, page 53](#)

### Mandatory.go\_ipsec.tcl Sample Policy

The following sample policy for the TestIPSecEncrypDecrypPkt Test.

```
::cisco::eem::event_register_gold card all testing_type monitoring test_name TestIPSecEncrypDecrypPkt consecutive_failure 6 platform_action 0 queue_priority last
#
# GOLD TestIPSecEncrypDecrypPkt Test TCL script
#
# March 2005, Hai Qiu
#
# Copyright (c) 2005-2007 by cisco Systems, Inc.
# All rights reserved.
#
#
# Register for TestIPSecEncrypDecrypPkt test even
# the elements for register the event
# card [all | card #]
# sub_card [all | sub_card #]
# severity_major | severity_minor | severity_normal default : severity_normal
# new_failure [true | false] default: dont_care
# testing_type [bootup | ondemand | schedule | monitoring]
# test_name [ test name ]
# test_id [ test # ]
# consecutive_failure [ consecutive_failure # ]
# platform_action [action_flag]
# action_flag [ 0 | 1 | 2 ]
# queue_priority [ normal | low | high | last] default: normal
#
# Note:
# 1: "card" element is required. If other elements are not specified,
#    treat them as dont care, or default.
#
# 2: action_flag is platform specific. It is up to platform to
#    determine what action need to be taken based on the value
#    For Cat6k platform
#    action_flag 0 : TCL script take action to reset card
#    action_flag 1 : TCL script doesn't take action to reset card
#    action_flag 2 : TCL script takes action to reset card for bootup diag
#                   when there is major error
#    action_flag 3 : TCL script doesn't take action to reset card for
#                   bootup diag when there is major error
#
# 3: "queue_priority last" would guarantee this policy will be executed last
#    if there are other EEM events in queue with queue priority other
#    than "last"
namespace import ::cisco::eem::*
namespace import ::cisco::lib::*
# 1. query the information of latest triggered eem event
array set arr_einfo [event_reinfo]
if {$_cerrno != 0} {
    set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
        $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}
puts "GOLD EEM TCL policy for TestIPSecEncrypDecrypPkt"
```

```

#set msg [format "array=%s", array names arr_einfo]
#puts "msg $msg"
#set msg $arr_einfo(msg)
set card $arr_einfo(card)
set sub_card $arr_einfo(sub_card)
#set overall_result $arr_einfo(overall_result)
#puts "GOLD event msg recieved: $card/$sub_card overall_result= $overall_result"
# 2. execute the user-defined config commands
if [catch {cli_open} result] {
    error $result $errorInfo
} else {
    array set cli1 $result
}
if [catch {cli_exec $cli1(fd) "en"} result] {
    error $result $errorInfo
}
# Use "diagn action mod mod# test testname default" command
# for default platform action
if [catch {cli_exec $cli1(fd) "diagnostic action mod $card test TestIPSecEncrypD
ecrypPkt default"} result] {
    error $result $errorInfo
} else {
    set cmd_output $result
}
if [catch {cli_close $cli1(fd) $cli1(tty_id)} result] {
    error $result $errorInfo
}

```

#### ap\_perf\_test\_base\_cpu.tcl Sample Policy

The following sample policy measures the CPU performance of EEM policies.

```

::cisco::eem::event_register_appl sub_system 798 type 9999
#-----
# EEM policy used for measuring the cpu performance of EEM policies.
#
# July 2005, Cisco EEM team
#
# Copyright (c) 2005, 2006 by cisco Systems, Inc.
# All rights reserved.
#-----
###
### Input arguments:
###
### arg1 $iter          - current iteration count
###
### The following EEM environment variables are used:
###
### _perf_iterations (mandatory) - number of iterations over which we
###                               will run our measurement.
### Example:
### event manager environment _perf_iterations 100
###
### _perf_cmd1 (optional) - optional non interactive cli command
###                       to be executed as part of the
###                       measurement test.
### Example:
### event manager environment _perf_cmd1 enable
###
### _perf_cmd2 (optional) - optional non interactive cli command
###                       to be executed as part of the
###                       measurement test.
###                               To use _perf_cmd2, _perf_cmd1 MUST
###                               be defined.

```

```

### Example:
### event manager environment _perf_cmd2 show ver
###
### _perf_cmd3 (optional)      - optional non interactive cli command
###                           to be executed as part of the
###                           measurement test.
###                           To use _perf_cmd3, _perf_cmd1 MUST
###                           be defined.
### Example:
### event manager environment _perf_cmd3 show int counters protocol status
###
### Description:
### Iterate through _perf_iterations of this policy.
### It is up to the user to calculate the average
### execution time based on the system timestamps.
### Optional commands _perf_cmd1,
### _perf_cmd2 and _perf_cmd3 are executed if defined.
###
### A value of 100 is a good starting point.
###
### Outputs:
### Console output.
###
### Usage example:
### >conf t
### >service timestamps debug datetime msec
### >event manager environment _perf_iterations 100
### >event manager policy ap_perf_base_cpu.tcl
### >event manager policy no_perf_test_init.tcl
### >end
### 2d19h: %SYS-5-CONFIG_I: Configured from console by console
### >event manager run no_perf_test_init.tcl
###
### Oct 16 14:57:17.284: %SYS-5-CONFIG_I: Configured from console by console
### >event manager run no_perf_test_init.tcl
###
### Oct 16 19:32:02.772: %HA_EM-6-LOG:
### eem_policy/no_perf_test_init.tcl: EEM performance test start
### Oct 16 19:32:03.115: %HA_EM-6-LOG:
### eem_policy/ap_perf_test_base_cpu.tcl: EEM performance test iteration 1
### Oct 16 19:32:03.467: %HA_EM-6-LOG:
### eem_policy/ap_perf_test_base_cpu.tcl: EEM performance test iteration 2
### ...
### Oct 16 19:32:36.936: %HA_EM-6-LOG:
### eem_policy/ap_perf_test_base_cpu.tcl: EEM performance test iteration 100
### Oct 16 19:32:36.936: %HA_EM-6-LOG:
### eem_policy/ap_perf_test_base_cpu.tcl: EEM performance test end
###
### The user must calculate execution time and average time of execution.
### In this example, total time = 19:32:36.936 - 19:32:02.772 = 34.164
### Average script execution time = 341.64 milliseconds
###
# check if all the env variables we need exist
# If any of them doesn't exist, print out an error msg and quit
if {[info exists _perf_iterations]} {
    set result \
        "Policy cannot be run: variable _perf_iterations has not been set"
    error $result $errorMsg
}
# ensure our target iteration count > 0
if {$_perf_iterations <= 0} {
    set result \
        "Policy cannot be run: variable _perf_iterations <= 0"
    error $result $errorMsg
}

```



```

}
namespace import ::cisco::eem::*
namespace import ::cisco::lib::*
# query the event info
array set arr_einfo [event_reqinfo]
if {$_cerrno != 0} {
    set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
        $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}
set iter $arr_einfo(data1)
set iter [expr $iter + 1]
# if _perf_cmd1 is defined
if {[info exists _perf_cmd1]} {
    # open the cli library
    if [catch {cli_open} result] {
        error $result $errorInfo
    } else {
        array set cli1 $result
    }
    # execute the comamnd defined in _perf_cmd1
    if [catch {cli_exec $cli1(fd) $_perf_cmd1} result] {
        error $result $errorInfo
    }
    # if _perf_cmd2 is defined
    if {[info exists _perf_cmd2]} {
        # execute the comamnd defined in _perf_cmd2
        if [catch {cli_exec $cli1(fd) $_perf_cmd2} result] {
            error $result $errorInfo
        } else {
            set cmd_output $result
        }
    }
    # if _perf_cmd3 is defined
    if {[info exists _perf_cmd3]} {
        # execute the comamnd defined in _perf_cmd3
        if [catch {cli_exec $cli1(fd) $_perf_cmd3} result] {
            error $result $errorInfo
        } else {
            set cmd_output $result
        }
    }
    # close the cli library
    if [catch {cli_close $cli1(fd) $cli1(tty_id)} result] {
        error $result $errorInfo
    }
}

# log a message
set msg [format "EEM performance test iteration %s" $iter]
action_syslog priority info msg $msg
if {$_cerrno != 0} {
    set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
        $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}

# use the context info from the previous run to determine when to end
if {$iter >= $_perf_iterations} {
    #log the final messages
    action_syslog priority info msg "EEM performance test end"
    if {$_cerrno != 0} {
        set result [format \
            "component=%s; subsys err=%s; posix err=%s;\n%s" \
            $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    }
}

```

```

        error $result
    }
    exit 0
}
# cause the next iteration to run
event_publish sub_system 798 type 9999 arg1 $iter
if {$_cerrno != 0} {
    set result [format \
        "component=%s; subsys err=%s; posix err=%s;\n%s" \
        $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}
}

```

### tm\_cli\_cmd.tcl Sample Policy

The following sample policy runs a configurable CRON entry. The policy executes a configurable Cisco IOS CLI command and e-mails the results. An optional log file can be defined to which the output is appended with a timestamp.

```

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

#-----
# EEM policy that will periodically execute a cli command and email the
# results to a user.
#
# July 2005, Cisco EEM team
#
# Copyright (c) 2005 by cisco Systems, Inc.
# All rights reserved.
#-----

### The following EEM environment variables are used:
###
### _cron_entry (mandatory)           - A CRON specification that determines
###                                   when the policy will run. See the
###                                   IOS Embedded Event Manager
###                                   documentation for more information
###                                   on how to specify a cron entry.
### Example: _cron_entry              0-59/1 0-23/1 * * 0-7
###
### _log_file (mandatory without _email_....)
###                                   - A filename to append the output to.
###                                   If this variable is defined, the
###                                   output is appended to the specified
###                                   file with a timestamp added.
### Example: _log_file                disk0:/my_file.log
###
### _email_server (mandatory without _log_file)
###                                   - A Simple Mail Transfer Protocol (SMTP)
###                                   mail server used to send e-mail.
### Example: _email_server            mailserver.example.com
###
### _email_from (mandatory without _log_file)
###                                   - The address from which e-mail is sent.
### Example: _email_from              devtest@example.com
###
### _email_to (mandatory without _log_file)
###                                   - The address to which e-mail is sent.
### Example: _email_to                engineering@example.com
###
### _email_cc (optional)              - The address to which the e-mail must
###                                   be copied.
### Example: _email_cc                manager@example.com
###

```

```

### _show_cmd (mandatory)           - The CLI command to be executed when
###                                 the policy is run.
### Example: _show_cmd             show version
###

# check if all required environment variables exist
# If any required environment variable does not exist, print out an error msg and quit
if {[info exists _log_file]} {
    if {[info exists _email_server]} {
        set result \
            "Policy cannot be run: variable _log_file or _email_server has not been set"
        error $result $errorInfo
    }
    if {[info exists _email_from]} {
        set result \
            "Policy cannot be run: variable _log_file or _email_from has not been set"
        error $result $errorInfo
    }
    if {[info exists _email_to]} {
        set result \
            "Policy cannot be run: variable _log_file ore _email_to has not been set"
        error $result $errorInfo
    }
    if {[info exists _email_cc]} {
        #_email_cc is an option, must set to empty string if not set.
        set _email_cc ""
    }
}

if {[info exists _show_cmd]} {
    set result \
        "Policy cannot be run: variable _show_cmd has not been set"
    error $result $errorInfo
}

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

# query the event info and log a message
array set arr_einfo [event_reqinfo]

if {$_cerrno != 0} {
    set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
        $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}

global timer_type timer_time_sec
set timer_type $arr_einfo(timer_type)
set timer_time_sec $arr_einfo(timer_time_sec)

# log a message
set msg [format "timer event: timer type %s, time expired %s" \
    $timer_type [clock format $timer_time_sec]]

action_syslog priority info msg $msg
if {$_cerrno != 0} {
    set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
        $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}

# 1. execute the command

```

```

if [catch {cli_open} result] {
    error $result $errorInfo
} else {
    array set cli1 $result
}
if [catch {cli_exec $cli1(fd) "en"} result] {
    error $result $errorInfo
}
# save exact execution time for command
set time_now [clock seconds]
# execute command
if [catch {cli_exec $cli1(fd) $_show_cmd} result] {
    error $result $errorInfo
} else {
    set cmd_output $result
    # format output: remove trailing router prompt
    regexp {\n*(.*\n)(^\n)*}$ $result dummy cmd_output
}
if [catch {cli_close $cli1(fd) $cli1(tty_id)} result] {
    error $result $errorInfo
}

# 2. log the success of the CLI command
set msg [format "Command \"%s\" executed successfully" $_show_cmd]
action_syslog priority info msg $msg
if {$_cerrno != 0} {
    set result [format "component=%s; subsystem err=%s; posix err=%s;\n%s" \
        $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}

# 3. if _log_file is defined, then attach it to the file
if {[info exists _log_file]} {
    # attach output to file
    if [catch {open $_log_file a+} result] {
        error $result
    }
    set fileD $result
    # save timestamp of command execution
    # (Format = 00:53:44 PDT Mon May 02 2005)
    set time_now [clock format $time_now -format "%T %Z %a %b %d %Y"]
    puts $fileD "%% Timestamp = $time_now"
    puts $fileD $cmd_output
    close $fileD
}

# 4. if _email_server is defined send the email out
if {[info exists _email_server]} {
    set routername [info hostname]
    if {[string match "" $routername]} {
        error "Host name is not configured"
    }

    if [catch {smtp_subst [file join $tcl_library email_template_cmd.tm]} \
        result] {
        error $result $errorInfo
    }

    if [catch {smtp_send_email $result} result] {
        error $result $errorInfo
    }
}

```

**sl\_intf\_down.tcl Sample Policy**

The following sample policy runs when a configurable syslog message is logged. The policy executes a configurable CLI command and e-mails the results.

```
::cisco::eem::event_register_syslog occurs 1 pattern $_syslog_pattern maxrun 90

#-----
# EEM policy to monitor for a specified syslog message.
# Designed to be used for syslog interface-down messages.
# When event is triggered, the given config commands will be run.
#
# July 2005, Cisco EEM team
#
# Copyright (c) 2005 by cisco Systems, Inc.
# All rights reserved.
#-----

### The following EEM environment variables are used:
###
### _syslog_pattern (mandatory)           - A regular expression pattern match string
###                                     that is used to compare syslog messages
###                                     to determine when policy runs
### Example: _syslog_pattern             .*UPDOWN.*FastEthernet0/0.*
###
### _email_server (mandatory)            - A Simple Mail Transfer Protocol (SMTP)
###                                     mail server used to send e-mail.
### Example: _email_server               mailserver.example.com
###
### _email_from (mandatory)              - The address from which e-mail is sent.
### Example: _email_from                 devtest@example.com
###
### _email_to (mandatory)                - The address to which e-mail is sent.
### Example: _email_to                   engineering@example.com
###
### _email_cc (optional)                 - The address to which the e-mail must
###                                     be copied.
### Example: _email_cc                   manager@example.com
###
### _config_cmd1 (optional)              - The first configuration command that
###                                     is executed.
### Example: _config_cmd1                interface Ethernet1/0
###
### _config_cmd2 (optional)              - The second configuration command that
###                                     is executed.
### Example: _config_cmd2                no shutdown
###

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

```

}
if {[info exists _email_cc]} {
    #_email_cc is an option, must set to empty string if not set.
    set _email_cc ""
}

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

# 1. query the information of latest triggered eem event
array set arr_einfo [event_reqinfo]

if {$_cerrno != 0} {
    set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
        $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}

set msg $arr_einfo(msg)
set config_cmds ""

# 2. execute the user-defined config commands
if [catch {cli_open} result] {
    error $result $errorInfo
} else {
    array set cli1 $result
}
if [catch {cli_exec $cli1(fd) "en"} result] {
    error $result $errorInfo
}
if [catch {cli_exec $cli1(fd) "config t"} result] {
    error $result $errorInfo
}

if {[info exists _config_cmd1]} {
    if [catch {cli_exec $cli1(fd) $_config_cmd1} result] {
        error $result $errorInfo
    }

    append config_cmds $_config_cmd1
}

if {[info exists _config_cmd2]} {
    if [catch {cli_exec $cli1(fd) $_config_cmd2} result] {
        error $result $errorInfo
    }
    append config_cmds "\n"
    append config_cmds $_config_cmd2
}

if [catch {cli_exec $cli1(fd) "end"} result] {
    error $result $errorInfo
}
if [catch {cli_close $cli1(fd) $cli1(tty_id)} result] {
    error $result $errorInfo
}

after 60000
# 3. send the notification email
set routename [info hostname]
if {[string match "" $routename]} {
    error "Host name is not configured"
}

```

```

if [catch {smtp_subst [file join $tcl_library email_template_cfg.tm]} result] {
    error $result $errorInfo
}
if [catch {smtp_send_email $result} result] {
    error $result $errorInfo
}

```

The following e-mail template file is used with the EEM sample policy above:

```

email_template_cfg.tm

Mailservername: $_email_server
From: $_email_from
To: $_email_to
Cc: $_email_cc
Subject: From router $routername: Periodic $_show_cmd Output

$cmd_output

```

## Debugging Embedded Event Manager Policies: Examples

The following examples show how to debug the CLI library and the SMTP library.

### Debugging the CLI Library

The CLI library allows users to run CLI commands and obtain the output of commands in Tcl. An Embedded Event Manager **debug** command has been provided for users of this library. The command to enable CLI library debugging is **debug event manager tcl cli\_library**. When enabled, this command displays all data that is passed in and read back from the TTY session that handles the CLI interactions. This data helps ensure users that the commands that they are passing to the CLI are valid.

### Example of the debug event manager tcl cli\_library Command

This example uses the sample policy `sl_intf_down.tcl`. When triggered, `sl_intf_down.tcl` passes a configuration command to the CLI through the CLI library. The command passed in below is **show event manager environment**. This command is not a valid command in configuration mode. Without the **debug** command enabled, the output is shown below:

```

00:00:57:sl_intf_down.tcl[0]:config_cmds are show eve man env
00:00:57:%SYS-5-CONFIG_I:Configured from console by vty0

```

Notice that with the output above the user would not know whether or not the command succeeded in the CLI. With the **debug event manager tcl cli\_library** command enabled, the user sees the following:

```

01:17:07: sl_intf_down.tcl[0]: DEBUG(cli_lib) : CTL : cli_open called.
01:17:07: sl_intf_down.tcl[0]: DEBUG(cli_lib) : OUT : nelson>
01:17:07: sl_intf_down.tcl[0]: DEBUG(cli_lib) : IN : nelson>enable
01:17:07: sl_intf_down.tcl[0]: DEBUG(cli_lib) : OUT : nelson#
01:17:07: sl_intf_down.tcl[0]: DEBUG(cli_lib) : IN : nelson#configure terminal
01:17:07: sl_intf_down.tcl[0]: DEBUG(cli_lib) : OUT : Enter configuration commands, one
per line. End with CNTL/Z.
01:17:07: sl_intf_down.tcl[0]: DEBUG(cli_lib) : OUT : nelson(config)#
01:17:07: sl_intf_down.tcl[0]: DEBUG(cli_lib) : IN : nelson(config)#show event manager
environment
01:17:07: sl_intf_down.tcl[0]: DEBUG(cli_lib) : OUT :
01:17:07: sl_intf_down.tcl[0]: DEBUG(cli_lib) : OUT : % Invalid input detected at '^'
marker.
01:17:07: sl_intf_down.tcl[0]: DEBUG(cli_lib) : OUT : nelson(config)#
01:17:07: sl_intf_down.tcl[0]: DEBUG(cli_lib) : IN : nelson(config)#end
01:17:07: sl_intf_down.tcl[0]: DEBUG(cli_lib) : OUT : nelson#
01:17:07: sl_intf_down.tcl[0]: DEBUG(cli_lib) : CTL : cli_close called.

```

```
01:17:07: sl_intf_down.tcl[0]: DEBUG(cli_lib) : IN : nelson#exit
01:17:07: sl_intf_down.tcl[0]: config_cmds are show event manager environment
01:17:07: %SYS-5-CONFIG-I: Configured from console by vty0
```

The output above shows that **show event manager environment** is an invalid command in configuration mode. The IN keyword signifies all data passed in to the TTY through the CLI library. The OUT keyword signifies all data read back from the TTY through the CLI library. The CTL keyword signifies helper functions used in the CLI library. These helper functions are used to set up and remove connections to the CLI.

### Debugging the SMTP Library

The SMTP library allows users to send e-mail messages to an SMTP e-mail server. An Embedded Event Manager **debug** command has been provided for users of this library. The command to enable SMTP library debugging is **debug event manager tcl smtp\_library**. When enabled, this command displays all data that is passed in and read back from the SMTP library routines. This data helps ensure users that the commands that they are passing to the SMTP library are valid.

### Example of the debug event manager tcl smtp\_library Command

This example uses the sample policy tm\_cli\_cmd.tcl. When triggered, tm\_cli\_cmd.tcl runs the command **show event manager policy available system** through the CLI library. The result is then mailed to a user through the SMTP library. The output will help debug any issues related to using the SMTP library.

With the **debug event manager tcl smtp\_library** command enabled, the users see the following on the console:

```
00:39:46: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_read : 220 XXXX.example.com ESMT
XXXX 1.1.0; Tue, 25 Jun 2002 14:20:39 -0700 (PDT)
00:39:46: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : HELO XXXX.example.com
00:39:46: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_read : 250 XXXX.example.com Hello
XXXX.example.com [XXXX], pleased to meet you
00:39:46: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : MAIL FROM:<XX@example.com>
00:39:46: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_read : 250 <XX@example.com>... Sender
ok
00:39:46: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : RCPT TO:<XX@example.com>
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_read : 250 <XX@example.com>...
Recipient ok
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : RCPT TO:<XX@example.com>
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_read : 250 <XX@example.com>...
Recipient ok
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : DATA
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_read : 354 Enter mail, end with "."
on a line by itself
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : Date: 25 Jun 2002 14:35:00 UTC
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : Message-ID:
<20020625143500.2387058729877@XXXX.example.com>
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : From: XX@example.com
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : To: XX@example.com
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : Cc: XX@example.com
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : Subject: From router nelson:
Periodic show eve man po ava system Output
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : No. Type Time Created
Name
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : 1 system Fri May3
20:42:34 2002 pr_cdp_abort.tcl
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : 2 system Fri May3
20:42:54 2002 pr_iprouting_abort.tcl
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : 3 system Wed Apr3
02:16:33 2002 sl_intf_down.tcl
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : 4 system Mon Jun24
23:34:16 2002 tm_cli_cmd.tcl
```



```

00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : 5      system  Wed Mar27
05:53:15 2002      tm_crash_hist.tcl
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : nelson#
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write :
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : .
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_read  : 250 ADE90179 Message accepted
for delivery
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : QUIT
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_read  : 221 XXXX.example.com closing
connection

```

## Tracing Tcl set Command Operations: Example

Tcl is a flexible language. One of the flexible aspects of Tcl is that you can override commands. In this example, the Tcl `set` command is renamed as `_set` and a new version of the `set` command is created that displays a message containing the text “setting” and appends the scalar variable that is being set. This example can be used to trace all instances of scalar variables being set.

```

rename set _set
proc set {var args} {
    puts [list setting $var $args]
    uplevel _set $var $args
};

```

When this is placed in a policy, a message is displayed anytime a scalar variable is set, for example:

```

02:17:58: sl_intf_down.tcl[0]: setting test_var 1

```

## RPC Event Detector: Example

```

TCL script (rpccli.tcl):

::cisco::eem::event_register_rpc

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

proc run_cli { clist } {
    set rbuf ""

    if {[llength $clist] < 1} {
        return -code ok $rbuf
    }

    if {[catch {cli_open} result]} {
        return -code error $result
    } else {
        array set cliarr $result
    }

    if {[catch {cli_exec $cliarr(fd) "enable"} result]} {
        return -code error $result
    }

    if {[catch {cli_exec $cliarr(fd) "term length 0"} result]} {
        return -code error $result
    }

    foreach cmd $clist {

```

```

    if {[catch {cli_exec $cliarr(fd) $cmd} result]} {
        return -code error $result
    }

    append rbuf $result
}

if {[catch {cli_close $cliarr(fd) $cliarr(tty_id)} result]} {
    puts "WARNING: $result"
}

return -code ok $rbuf
}

proc run_cli_interactive { clist } {
    set rbuf ""

    if {[length $clist] < 1} {
        return -code ok $rbuf
    }

    if {[catch {cli_open} result]} {
        return -code error $result
    } else {
        array set cliarr $result
    }

    if {[catch {cli_exec $cliarr(fd) "enable"} result]} {
        return -code error $result
    }

    if {[catch {cli_exec $cliarr(fd) "term length 0"} result]} {
        return -code error $result
    }

    foreach cmd $clist {
        array set sendexp $cmd

        if {[catch {cli_write $cliarr(fd) $sendexp(send)} result]} {
            return -code error $result
        }

        foreach response $sendexp(responses) {
            array set resp $response

            if {[catch {cli_read_pattern $cliarr(fd) $resp(expect)} result]} {
                return -code error $result
            }

            if {[catch {cli_write $cliarr(fd) $resp(reply)} result]} {
                return -code error $result
            }
        }

        if {[catch {cli_read $cliarr(fd)} result]} {
            return -code error $result
        }

        append rbuf $result
    }

    if {[catch {cli_close $cliarr(fd) $cliarr(tty_id)} result]} {
        puts "WARNING: $result"
    }
}

```

```

        return -code ok $rbuf
    }

    array set arr_einfo [event_reqinfo]

    set args $arr_einfo(args)

    set cmds [list]

    for { set i 0 } { $i < $args } { incr i } {
        set arg "arg${i}"
        # Split each argument on the '^' character. The first element is
        # the command, and each subsequent element is a prompt followed by
        # a response to that prompt.
        set cmdlist [split $arr_einfo($arg) "^"]
        set cmdarr(send) [lindex $cmdlist 0]
        set cmdarr(responses) [list]
        if { [expr ([llength $cmdlist] - 1) % 2] != 0 } {
            return -code 88
        }
        set cmdarr(responses) [list]
        for { set j 1 } { $j < [llength $cmdlist] } { incr j 2 } {
            set resps(expect) [lindex $cmdlist $j]
            set resps(reply) [lindex $cmdlist [expr $j + 1]]
            lappend cmdarr(responses) [array get resps]
        }
        lappend cmds [array get cmdarr]
    }

    set rc [catch {run_cli_interactive $cmds} output]

    if { $rc != 0 } {
        error $output $errorInfo
        return -code 88
    }

    puts $output

```

## Where to Go Next

- For information about EEM overview, go to [“Embedded Event Manager Overview”](#) module.
- For information about writing EEM policies using the Cisco IOS CLI, go to the [“Writing Embedded Event Manager Policies Using the Cisco IOS CLI”](#) module.

## Additional References

The following sections provide references related to writing Embedded Event Manager policies using Tcl.

### Related Documents

Related Topic	Document Title
Cisco IOS commands	<a href="#">Cisco IOS Master Commands List, All Releases</a>
Network Management commands (including EEM commands): complete command syntax, defaults, command mode, command history, usage guidelines, and examples	<a href="#">Cisco IOS Network Management Command Reference</a>
Embedded Event Manager overview	<a href="#">Embedded Event Manager Overview</a> module.
Embedded Event Manager policy writing using the CLI	<a href="#">Writing Embedded Event Manager Policies Using the Cisco IOS CLI</a> module
Embedded Resource Manager	<a href="#">Embedded Resource Manager</a> module

### MIBs

MIB	MIBs Link
CISCO-EMBEDDED-EVENT-MGR-MIB	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: <a href="http://www.cisco.com/go/mibs">http://www.cisco.com/go/mibs</a>

### 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
<p>The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.</p>	<p><a href="http://www.cisco.com/cisco/web/support/index.html">http://www.cisco.com/cisco/web/support/index.html</a></p>

# EEM Policy Tcl Command Extension Reference

This section documents the following EEM policy Tcl command extension categories:

- [EEM Event Registration Tcl Command Extensions, page 63](#)
- [EEM Event Information Tcl Command Extension, page 124](#)
- [EEM Event Tcl Command Extension, page 149](#)
- [EEM Action Tcl Command Extensions, page 163](#)
- [EEM Utility Tcl Command Extensions, page 175](#)
- [EEM System Information Tcl Command Extensions, page 190](#)
- [EEM Library Debug Command Extensions, page 205](#)
- [SMTP Library Command Extensions, page 208](#)
- [CLI Library Command Extensions, page 211](#)
- [Tcl Context Library Command Extensions, page 239](#)
- [Event Registration Tcl Command Extensions for EEM 3.2, page 245](#)




---

**Note**

For all EEM Tcl command extensions, if there is an error, the returned Tcl result string contains the error information.

---




---

**Note**

Arguments for which no numeric range is specified take an integer from `-2147483648` to `2147483647`, inclusive.

---

The following conventions are used for the syntax documented on the Tcl command extension pages:

- An optional argument is shown within square brackets, for example:  
`[type ?]`
- A question mark `?` represents a variable to be entered.
- Choices between arguments are represented by pipes, for example:  
`priority low|normal|high`

## EEM Event Registration Tcl Command Extensions

- [event\\_register\\_appl](#), page 64
- [event\\_register\\_cli](#), page 66
- [event\\_register\\_counter](#), page 69
- [event\\_register\\_gold](#), page 71
- [event\\_register\\_interface](#), page 75
- [event\\_register\\_ipsla](#), page 82
- [event\\_register\\_nf](#), page 85
- [event\\_register\\_ioswdsysmon](#), page 80
- [event\\_register\\_none](#), page 88
- [event\\_register\\_oir](#), page 89
- [event\\_register\\_process](#), page 91
- [event\\_register\\_resource](#), page 93
- [event\\_register\\_rf](#), page 95
- [event\\_register\\_routing](#), page 97
- [event\\_register\\_rpc](#), page 99
- [event\\_register\\_snmp](#), page 101
- [event\\_register\\_snmp\\_notification](#), page 104
- [event\\_register\\_snmp\\_object](#), page 106
- [event\\_register\\_syslog](#), page 108
- [event\\_register\\_timer](#), page 111
- [event\\_register\\_timer\\_subscriber](#), page 115
- [event\\_register\\_track](#), page 117
- [event\\_register\\_wdsysmon](#), page 119

## event\_register\_appl

Registers for an application event. Use this Tcl command extension to run a policy when an application event is triggered following another policy's execution of an **event\_publish** Tcl command extension; the **event\_publish** command extension publishes an application event.

In order to register for an application event, a subsystem must be specified. Either a Tcl policy or the internal Embedded Event Manager (EEM) API can publish an application event. If the event is being published by a policy, the `sub_system` argument that is reserved for a policy is 798.

### Syntax

```
event_register_appl [tag ?] sub_system ? type ? [queue_priority low|normal|high|last]
[maxrun ?] [nice 0|1]
```

### Arguments

tag	(Optional) String identifying a tag that can be used with the trigger Tcl command extension to support multiple event statements within a Tcl script.
sub_system	(Mandatory) Number assigned to the EEM policy that published the application event. The number is set to 798 because all other numbers are reserved for Cisco use. If this argument is not specified, all components are matched.
type	(Mandatory) Event subtype within the specified event. The <code>sub_system</code> and <code>type</code> arguments uniquely identify an application event. If this argument is not specified, all types are matched. If you specify this argument, you must choose an integer between 1 and 4294967295, inclusive.  There must be a match of component and type between the <b>event_publish</b> command extension and the <b>event_register_appl</b> command extension in order for the publishing and registration to work.



queue_priority	<p>(Optional) Priority level at which the script will be queued:</p> <ul style="list-style-type: none"> <li>• queue_priority low—Specifies that the script is to be queued at the lowest of the three priority levels.</li> <li>• queue_priority normal—Specifies that the script is to be queued at a priority level greater than low priority but less than high priority.</li> <li>• queue_priority high—Specifies that the script is to be queued at the highest of the three priority levels.</li> <li>• queue_priority last—Specifies that the script is to be queued at the lowest priority level.</li> </ul> <p>If more than one script is registered with the “queue_priority_last” argument set, these scripts will execute in the order in which the events are published.</p> <p><b>Note</b> The queue_priority argument specifies the queuing priority, but not the execution priority, of the script being registered.</p> <p>If this argument is not specified, the default queuing priority is normal.</p>
maxrun	<p>(Optional) Maximum run time of the script (specified in SSSSSSSSS[.MMM] format, where SSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the default 20-second run-time limit is used.</p>
nice	<p>(Optional) Policy run-time priority setting. When the nice argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.</p>

If multiple conditions exist, the application event will be raised when all the conditions are satisfied.

**Result String**

None

**Set\_cerrno**

No

## event\_register\_cli

Registers for a CLI event. Use this Tcl command extension to run a policy when a CLI command of a specific pattern is entered based on pattern matching performed against an expanded CLI command.


**Note**

The user can enter an abbreviated CLI command, such as **sh mem summary**, and the parser will expand the command to **show memory summary** to perform the matching.



**Note**

The functionality provided in the CLI event detector only allows a regular expression pattern match on a valid IOS CLI command itself. This does not include text after a pipe character when redirection is used.

### Syntax

```
event_register_cli [tag ?] sync yes|no skip yes|no
[occurs ?] [period ?] pattern ? [default ?] [enter] [questionmark] [tab] [mode]
[queue_priority low|normal|high|last] [maxrun ?] [nice 0|1]
```

### Arguments

tag	(Optional) String identifying a tag that can be used with the trigger Tcl command extension to support multiple event statements within a Tcl script.
sync	(Mandatory) A “yes” means that the policy (the event publish) will run synchronously with the CLI command; a “no” means that the event publish will be performed asynchronously with the CLI command. The event detector will be notified when the policy completes running. The exit status of the policy indicates whether or not the CLI command should be executed: if the exit status is zero, which means that the policy is executed successfully, the CLI command will not be executed; otherwise, the CLI command will be executed.
skip	Mandatory if the sync argument is “no” and should not exist if the sync argument is “yes.” If the skip argument is “yes,” it means that the CLI command should not be executed. If the skip argument is “no,” it means that the CLI command should be executed.   <b>Caution</b> When the skip argument is “yes,” unintended results may be produced if the pattern match is made for configuration commands because the CLI command that matches the regular expression will not be executed.
occurs	(Optional) The number of occurrences before the event is raised. If this argument is not specified, the event is raised on the first occurrence. If this argument is specified, it must be an integer between 1 and 4294967295, inclusive.

period	(Optional) Specifies a backward looking time window in which all CLI events must occur (the occurs clause must be satisfied) in order for an event to be published (specified in SSSSSSSSS[.MMM] format, where SSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the most recent event is used.
pattern	(Mandatory) Specifies the regular expression used to perform the CLI command pattern match.
default	(Optional) The time period during which the CLI event detector waits for the policy to exit (specified in SSSSSSSSS[.MMM] format, where SSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If the default time period expires before the policy exits, the default action will be executed. The default action is to run the command. If this argument is not specified, the default time period is set to 30 seconds.
queue_priority	<p>(Optional) Priority level at which the script will be queued:</p> <ul style="list-style-type: none"> <li>• queue_priority low—Specifies that the script is to be queued at the lowest of the three priority levels.</li> <li>• queue_priority normal—Specifies that the script is to be queued at a priority level greater than low priority but less than high priority.</li> <li>• queue_priority high—Specifies that the script is to be queued at the highest of the three priority levels.</li> <li>• queue_priority last—Specifies that the script is to be queued at the lowest priority level.</li> </ul> <p>If more than one script is registered with the “queue_priority_last” argument set, these scripts will execute in the order in which the events are published.</p> <p><b>Note</b> The queue_priority argument specifies the queuing priority, but not the execution priority, of the script being registered.</p> <p>If this argument is not specified, the default queuing priority is normal.</p>
enter	(Optional) Specifies to perform the event match when the user presses the Enter key. When this parameter is used, the input string will not be expanded before matching.
questionmark	(Optional) Specifies to perform the event match when the user presses the ? key. When this parameter is used, the input string will not be expanded before matching.
tab	(Optional) Specifies to perform the event match when the user presses the Tab key. When this parameter is used, the input string will not be expanded before matching.

mode	(Optional) Events will only be generated when the parser is in the specified parser mode. The available modes can be listed using the <b>show parser dump</b> CLI command. The mode parameter is checked when any one of the optional parameters—enter, questionmark, or tab— is specified.
maxrun	(Optional) Maximum run time of the script (specified in SSSSSSSSS[.MMM] format, where SSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the default 20-second run-time limit is used.
nice	(Optional) Policy run-time priority setting. When the nice argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.

If multiple conditions are specified, the CLI event will be raised when all the conditions are matched.

#### Result String

None

#### Set\_cerrno

No



#### Note

This policy runs before the CLI command is executed. For example, suppose policy\_CLI is registered to run when the **copy** command is entered. When the **copy** command is entered, the CLI event detector finds a pattern match and triggers this policy to run. When the policy execution ends, the CLI event detector determines if the **copy** command needs to be executed according to “sync”, “skip” (set in the policy), and the exit status of the policy execution if needed.

## event\_register\_counter

Registers for a counter event as both a publisher and a subscriber. Use this Tcl command extension to run a policy on the basis of a named counter crossing a threshold. This event counter, as a subscriber, identifies the name of the counter to which it wants to subscribe and depends on another policy or another process to actually manipulate the counter. For example, let policyB act as a counter policy, whereas policyA (although it does not need to be a counter policy) uses **register\_counter**, **counter\_modify**, or **unregister\_counter** Tcl command extensions to manipulate the counter defined in policyB.

### Syntax

```
event_register_counter [tag ?] name ? entry_op gt|ge|eq|ne|lt|le entry_val ?
exit_op gt|ge|eq|ne|lt|le exit_val ? [queue_priority low|normal|high|last]
[maxrun ?] [nice 0|1]
```

### Arguments

tag	(Optional) String identifying a tag that can be used with the trigger Tcl command extension to support multiple event statements within a Tcl script.
name	(Mandatory) Name of the counter.
entry_op	(Mandatory) Entry comparison operator used to compare the current counter value with the entry value; if true, an event will be raised and event monitoring will be disabled until exit criteria are met.
entry_val	(Mandatory) Value with which the current counter value should be compared to decide if the counter event should be raised.
exit_op	(Mandatory) Exit comparison operator used to compare the current counter value with the exit value; if true, event monitoring for this event will be reenabled.
exit_val	(Mandatory) Value with which the current counter value should be compared to decide if the exit criteria are met.

queue_priority	<p>(Optional) Priority level at which the script will be queued:</p> <ul style="list-style-type: none"> <li>• queue_priority low—Specifies that the script is to be queued at the lowest of the three priority levels.</li> <li>• queue_priority normal—Specifies that the script is to be queued at a priority level greater than low priority but less than high priority.</li> <li>• queue_priority high—Specifies that the script is to be queued at the highest of the three priority levels.</li> <li>• queue_priority last—Specifies that the script is to be queued at the lowest priority level.</li> </ul> <p>If more than one script is registered with the “queue_priority_last” argument set, these scripts will execute in the order in which the events are published.</p> <p><b>Note</b> The queue_priority argument specifies the queuing priority, but not the execution priority, of the script being registered.</p> <p>If this argument is not specified, the default queuing priority is normal.</p>
maxrun	<p>(Optional) Maximum run time of the script (specified in SSSSSSSSS[.MMM] format, where SSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the default 20-second run-time limit is used.</p>
nice	<p>(Optional) Policy run-time priority setting. When the nice argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.</p>

**Result String**

None

**Set\_cerrno**

No

# event\_register\_gold

Registers for a Generic Online Diagnostic (GOLD) failure event. Use this Tcl command extension to run a policy on the basis of a Generic Online Diagnostic (GOLD) failure event for the specified card and subcard.

## Syntax

```
event_register_gold card all|card_number
[subcard all|subcard_number]
[new_failure TRUE|FALSE]
[severity_major TRUE]
[severity_minor TRUE]
[severity_normal TRUE]
[action_notify TRUE|FALSE]
[testing_type [bootup|ondemand|schedule|monitoring]]
[test_name [testname]]
[test_id [testnumber]]
[consecutive_failure consecutive_failure_number]
[platform_action [action_flag]]
[maxrun ?]
[queue_priority low|normal|high|last]
[nice 0|1]
```

## Arguments

card	<p>(Mandatory) Specifies whether all cards or one card is to be monitored:</p> <ul style="list-style-type: none"> <li>card all—Specifies that all cards are to be monitored. This is the default.</li> <li>card-number—Specifies that the card identified by the number card-number is to be monitored.</li> </ul> <p>This argument must be specified to complete the <b>event_register_gold</b> Tcl command extension.</p>
subcard	<p>(Optional) Specifies that one or more subcards are to be monitored:</p> <ul style="list-style-type: none"> <li>subcard all—Specifies that all subcards are to be monitored.</li> <li>subcard-number—Specifies that the subcard identified by the number subcard-number is to be monitored.</li> </ul> <p>If this argument is not specified, all subcards are monitored by default.</p>
new_failure	<p>(Optional) Specifies event criteria based on the new test failure information from GOLD:</p> <ul style="list-style-type: none"> <li>new_failure TRUE—Specifies that the event criterion for the new test failure is true from GOLD.</li> <li>new_failure FALSE—Specifies that the event criterion for the new test failure is false from GOLD.</li> </ul> <p>If this argument is not specified, the new test failure information from GOLD is not considered in the event criteria.</p>
severity_major	<p>(Optional) Specifies that the event criteria for diagnostic result matches with the diagnostic major error from GOLD.</p>

severity_minor	(Optional) Specifies that the event criteria for diagnostic result matches with diagnostic minor error from GOLD.
severity_normal	(Optional) Specifies that the event criteria for diagnostic result matches with diagnostic normal from GOLD. This is the default.
action_notify	<p>(Optional) Specifies the event criteria based on the action notify information from GOLD:</p> <ul style="list-style-type: none"> <li>• action_notify TRUE—Specifies that the event criterion for the action notify is true from GOLD.</li> <li>• action_notify FALSE—Specifies that the event criterion for the action notify is false from GOLD.</li> </ul> <p>If this argument is not specified, the action notify information from GOLD is not considered in the event criteria.</p>
testing_type	<p>(Optional) Specifies the event criteria based on the testing types of the diagnostic from GOLD:</p> <ul style="list-style-type: none"> <li>• testing_type bootup—Specifies the diagnostic tests that are running on system bootup.</li> <li>• testing_type ondemand—Specifies the diagnostic tests that are running from CLI after the card is online.</li> <li>• testing_type schedule—Specifies the scheduled diagnostic tests.</li> <li>• testing_type monitoring—Specifies the diagnostic tests that are running periodically in the background to monitor the health of the system.</li> </ul> <p>If this argument is not specified, the testing type information from GOLD is not considered in the event criteria and the policy applies to all the diagnostic testing types.</p>
test_name	<p>(Optional) Specifies the event criteria based on the test name:</p> <ul style="list-style-type: none"> <li>• test_name test-name—Specifies the event criteria based on the test with the name test-name.</li> </ul> <p>If this argument is not specified, the test name information from GOLD is not considered in the event criteria.</p>
test_id	<p>(Optional) Specifies the event criteria based on test ID:</p> <ul style="list-style-type: none"> <li>• test_id test-id—Specifies the event criteria based on the test with the ID number test-id. The maximum value of test-id is 65535.</li> </ul> <p><b>Note</b> Because the test ID can be different for the same test on different line cards, usually the test_name keyword should be used instead. If the test ID is specified and conflicts with the specified test name, the test name overwrites the test ID.</p> <p>If this argument is not specified, test ID information from GOLD is not considered in the event criteria.</p>



consecutive_failure	<p>(Optional) Specifies the event criteria based on consecutive test failure information from GOLD:</p> <ul style="list-style-type: none"> <li>consecutive_failure consecutive-failure-number—Specifies that the event criterion is based on the occurrence of consecutive-failure-number consecutive test failures.</li> </ul> <p>If this argument is not specified, consecutive test failure information from GOLD is not considered in the event criteria.</p>
platform_action	<p>(Optional) Specifies whether callback to the platform is needed when all the event criteria are matched. When callback is needed, the platform needs to register a callback function through the provided registry.</p> <ul style="list-style-type: none"> <li>platform_action action-flag-number—Specifies that, when callback to the platform is needed, specific information is specified by the platform-specific action-flag-number value. The maximum value of action-flag-number is 65535.</li> </ul> <p><b>Note</b> It is up to the platform to determine what action needs to be taken based on the flag.</p> <p>If this argument is not specified, there is no callback.</p>
maxrun	<p>(Optional) Specifies the maximum runt time of the script.</p> <ul style="list-style-type: none"> <li>maxrun max-run-time-number—Specifies that the maximum run time of the script is max-run-time-number seconds. The maximum value of max-run-time-number is 4294967295 seconds.</li> </ul> <p>If this argument is not specified, the default run time is 20 seconds.</p>

queue_priority	<p>(Optional) Priority level at which the script will be queued:</p> <ul style="list-style-type: none"> <li>• queue_priority low—Specifies that the script is to be queued at the lowest of the three priority levels.</li> <li>• queue_priority normal—Specifies that the script is to be queued at a priority level greater than low priority but less than high priority.</li> <li>• queue_priority high—Specifies that the script is to be queued at the highest of the three priority levels.</li> <li>• queue_priority last—Specifies that the script is to be queued at the lowest priority level.</li> </ul> <p>If more than one script is registered with the “queue_priority_last” argument set, these scripts will execute in the order in which the events are published.</p> <p><b>Note</b> The queue_priority argument specifies the queuing priority, but not the execution priority, of the script being registered.</p> <p>If this argument is not specified, the default queuing priority is normal.</p>
nice	<p>(Optional) Policy run-time priority setting:</p> <ul style="list-style-type: none"> <li>• nice 0—Specifies that the policy is run at the default run-time priority level.</li> <li>• nice 1—Specifies that the policy is run at a run-time priority that is less than the default priority.</li> </ul> <p>If this argument is not specified, the default run-time priority is used.</p>

**Result String**

None

**Set\_cerrno**

No

# event\_register\_interface

Registers for an interface counter event. Use this Tcl command extension to generate an event when specified interface counters exceed specified thresholds.

## Syntax

```
event_register_interface [tag ?] name ?
parameter ? entry_op gt|ge|eq|ne|lt|le
entry_val ? entry_val_is_increment TRUE|FALSE
entry_type value|increment|rate
[exit_comb or|and]
[exit_op gt|ge|eq|ne|lt|le]
[exit_val ?] [exit_val_is_increment TRUE|FALSE]
[exit_type value|increment|rate]
[exit_time ?] [poll_interval ?]
[average_factor ?] [queue_priority low|normal|high|last]
[maxrun ?] [nice 0|1]
```

## Arguments

tag	(Optional) String identifying a tag that can be used with the trigger Tcl command extension to support multiple event statements within a Tcl script.
name	(Mandatory) The name of the interface being monitored, for example, Ethernet 0/0. Abbreviations and spaces are not allowed.
parameter	(Mandatory) The name of the counter being compared as follows: <ul style="list-style-type: none"> <li>input_errors—Includes runts, giants, no buffer, CRC, frame, overrun, and ignored counts. Other input-related errors can also cause the input errors count to be increased, and some datagrams may have more than one error; therefore, this sum may not balance with the sum of enumerated input error counts.</li> <li>input_errors_crc—Cyclic redundancy checksum generated by the originating LAN station or far-end device does not match the checksum calculated from the data received.</li> <li>input_errors_frame—Number of packets received incorrectly having a CRC error and a noninteger number of octets.</li> <li>input_errors_overrun—Number of times the receiver hardware was unable to hand received data to a hardware buffer because the input rate exceeded the receiver’s ability to handle the data.</li> <li>input_packets_dropped—Number of packets dropped because of a full input queue.</li> <li>interface_resets—Number of times that an interface has been completely reset.</li> <li>output_buffer_failures—Number of failed buffers and number of buffers swapped out.</li> <li>output_buffer_swappedout—Number of packets swapped to DRAM.</li> </ul>

parameter (continued)	<ul style="list-style-type: none"> <li>• output_errors—Sum of all errors that prevented the final transmission of datagrams out of the interface being examined. Note that this may not balance with the sum of the enumerated output errors, because some datagrams may have more than one error, and others may have errors that do not fall into any of the specifically tabulated categories.</li> <li>• output_errors_underrun—Number of times that the transmitter has been running faster than the router can handle.</li> <li>• output_packets_dropped—Number of packets dropped because of a full output queue.</li> <li>• receive_broadcasts—Number of broadcast or multicast packets received by the interface.</li> <li>• receive_giants—Number of packets that are discarded because they exceed the maximum packet size of the medium.</li> <li>• receive_rate_bps—Interface receive rate in bytes per second.</li> <li>• receive_rate_pps—Interface receive rate in packets per second.</li> <li>• receive_runts—Number of packets that are discarded because they are smaller than the minimum packet size of the medium.</li> <li>• receive_throttle—Number of times that the receiver on the port was disabled, possibly because of buffer or processor overload.</li> <li>• reliability—Reliability of the interface as a fraction of 255 (255/255 is 100 percent reliability), calculated as an exponential average over 5 minutes.</li> <li>• rxload—Receive rate of the interface as a fraction of 255 (255/255 is 100 percent).</li> <li>• transmit_rate_bps—Interface transmit rate in bytes per second.</li> <li>• transmit_rate_pps—Interface transmit rate in packets per second.</li> <li>• txload—Transmit rate of the interface as a fraction of 255 (255/255 is 100 percent).</li> </ul>
entry_op	(Mandatory) The comparison operator used to compare the current interface value with the entry value; if true, an event will be raised and event monitoring will be disabled until exit criteria are met.
entry_val	(Mandatory) The value at which the event will be triggered.
entry_val_is_increment	<p>(Mandatory) If TRUE, the entry_val field is treated as an incremental difference and is compared with the difference between the current counter value and the value when the event was last true (the first polled sample if this is a new event). A negative value checks the incremental difference for a counter that is decreasing. If FALSE, the entry_val field is compared against the current counter value.</p> <p><b>Note</b> In Cisco IOS Release 12.4(20)T, this keyword is deprecated, and if specified, the syntax is converted into equivalent entry-type keyword syntax.</p>

entry-type	<p>Specifies a type of operation to be applied to the object ID specified by the entry-val argument.</p> <p>Value is defined as the actual value of the entry-val argument.</p> <p>Increment uses the entry-val field as an incremental difference and the entry-val is compared with the difference between the current counter value and the value when the event was last triggered (or the first polled sample if this is a new event). A negative value checks the incremental difference for a counter that is decreasing.</p> <p>Rate is defined as the average rate of change over a period of time. The time period is the average-factor value multiplied by the poll-interval value. At each poll interval the difference between the current sample and the previous sample is taken and recorded as an absolute value. An average of the previous average-factor value samples is taken to be the rate of change.</p>
exit_comb	<p>(Optional) Used to indicate the combination of exit condition tests required to rearm the event trigger; if the and operator is specified, both exit value and exit time tests must be true to cause rearm; if the or operator is specified, either exit value or exit time tests can be true to cause event monitoring to be rearmed.</p>
exit_op	<p>(Optional) The comparison operator used to compare the current interface value with the exit value; if true, event monitoring for this event will be reenabled.</p>
exit_val	<p>(Optional) The value at which the event is rearmed to be monitored again.</p>
exit_val_is_increment	<p>(Optional) If TRUE, the exit_val field is treated as an incremental difference and is compared with the difference between the current counter value and the value when the event was last true. A negative value checks the incremental difference for a counter that is decreasing. If FALSE, the exit_val field is compared against the current counter value.</p> <p><b>Note</b> In Cisco IOS Release 12.4(20)T, this keyword is deprecated, and if specified, the syntax is converted into equivalent exit-type keyword syntax.</p>
exit-type	<p>(Optional) Specifies a type of operation to be applied to the object ID specified by the exit-val argument. If not specified, the value is assumed.</p> <p>Value is defined as the actual value of the exit-val argument.</p> <p>Increment uses the exit-val field as an incremental difference and the exit-val is compared with the difference between the current counter value and the value when the event was last triggered (or the first polled sample if this is a new event). A negative value checks the incremental difference for a counter that is decreasing.</p> <p>Rate is defined as the average rate of change over a period of time. The time period is the average-factor value multiplied by the poll-interval value. At each poll interval the difference between the current sample and the previous sample is taken and recorded as an absolute value. An average of the previous average-factor value samples is taken to be the rate of change.</p>

exit_time	(Optional) The time period at which the event is rearmed to be monitored again (specified in SSSSSSSSSS[.MMM] format, where SSSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999).
poll_interval	(Optional) The frequency used to collect the samples (specified in SSSSSSSSSS[.MMM] format, where SSSSSSSSSS must be an integer representing seconds between 60 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). The poll interval value must not be less than 1 second. The default is 1 second.
average-factor	(Optional) Number in the range from 1 to 64 used to calculate the period used for rate-based calculations. The average-factor value is multiplied by the poll-interval value to derive the period in milliseconds. The minimum average factor value is 1.
queue_priority	<p>(Optional) Priority level at which the script will be queued:</p> <ul style="list-style-type: none"> <li>• queue_priority low—Specifies that the script is to be queued at the lowest of the three priority levels.</li> <li>• queue_priority normal—Specifies that the script is to be queued at a priority level greater than low priority but less than high priority.</li> <li>• queue_priority high—Specifies that the script is to be queued at the highest of the three priority levels.</li> <li>• queue_priority last—Specifies that the script is to be queued at the lowest priority level.</li> </ul> <p>If more than one script is registered with the “queue_priority_last” argument set, these scripts will execute in the order in which the events are published.</p> <p><b>Note</b> The queue_priority argument specifies the queuing priority, but not the execution priority, of the script being registered.</p> <p>If this argument is not specified, the default queuing priority is normal.</p>
maxrun	(Optional) Maximum run time of the script (specified in SSSSSSSSSS[.MMM] format, where SSSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the default 20-second run-time limit is used.
nice	(Optional) Policy run-time priority setting. When the nice argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.

**Result String**

None

**Set\_cerrno**

No

## event\_register\_ioswdsysmon

Registers for an IOSWDSysMon event. Use this Tcl command extension to generate an event when a Cisco IOS task exceeds specific CPU utilization or memory thresholds. A Cisco IOS task is called a Cisco IOS process in native Cisco IOS.

### Syntax

```
event_register_ioswdsysmon [tag ?] [timewin ?] [sub12op and|or] [sub1 ?] [sub2 ?]
[queue_priority low|normal|high|last] [maxrun ?] [nice 0|1]
```

### Arguments

tag	(Optional) String identifying a tag that can be used with the trigger Tcl command extension to support multiple event statements within a Tcl script.
timewin	(Optional) Defines the time window within which all of the subevents must occur in order for an event to be generated (specified in SSSSSSSSS[.MMM] format, where SSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999).
sub12_op	(Optional) The combination operator for comparison between subevent 1 and subevent 2.
sub1	(Optional) The subevent 1 specification.
sub2	(Optional) The subevent 2 specification.
queue_priority	<p>(Optional) Priority level at which the script will be queued:</p> <ul style="list-style-type: none"> <li>queue_priority low—Specifies that the script is to be queued at the lowest of the three priority levels.</li> <li>queue_priority normal—Specifies that the script is to be queued at a priority level greater than low priority but less than high priority.</li> <li>queue_priority high—Specifies that the script is to be queued at the highest of the three priority levels.</li> <li>queue_priority last—Specifies that the script is to be queued at the lowest priority level.</li> </ul> <p>If more than one script is registered with the “queue_priority_last” argument set, these scripts will execute in the order in which the events are published.</p> <p><b>Note</b> The queue_priority argument specifies the queuing priority, but not the execution priority, of the script being registered.</p> <p>If this argument is not specified, the default queuing priority is normal.</p>



maxrun	(Optional) Maximum run time of the script (specified in SSSSSSSSSS[.MMM] format, where SSSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the default 20-second run-time limit is used.
nice	(Optional) Policy run-time priority setting. When the nice argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.

**Subevent Syntax**

cpu\_proc path ? taskname ? op gt|ge|eq|ne|lt|le val ? [period ?]

mem\_proc path ? taskname ? op gt|ge|eq|ne|lt|le val ? [is\_percent TRUE|FALSE] [period ?]

**Subevent Arguments**

cpu_proc	(Mandatory) Specifies the use of a sample collection of CPU statistics.
path	(Mandatory) Software Modularity images only. The pathname of the POSIX process that contains the Cisco IOS scheduler to be monitored. For example, /sbin/cdp2.iosproc.
taskname	(Mandatory) The name of the Cisco IOS task to be monitored.
op	(Mandatory) The comparison operator used to compare the collected usage sample with the specified value; if true, an event will be raised.
val	(Mandatory) The value to be compared.
period	(Optional) The elapsed time period for the collection samples to be averaged (specified in SSSSSSSSSS[.MMM] format, where SSSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the most recent sample is used.
mem_proc	(Mandatory) Specifies the use of a sample collection of memory statistics.
is_percent	(Optional) Whether the specified value is a percentage.

**Result String**

None

**Set\_cerrno**

No

## event\_register\_ipsla

Registers for an event that is triggered by the **event ipsla** command. Use this Tcl command to publish an event when an IPSLA reaction is triggered. The group ID or the operation ID is required to register the event.

### Syntax

```
event_register_ipsla [tag ?] group_name ? operation_id ? [reaction_type ?]
[dest_ip_addr ?][queue_priority low|normal|high|last] [maxrun ?] [nice 0|1]
```

### Arguments

tag	(Optional) String identifying a tag that can be used with the trigger Tcl command extension to support multiple event statements within a Tcl script.
group_name	(Mandatory) Specifies the IP SLAs group name.
operation_id	(Mandatory) Specifies the IP SLA operation ID. Number must be in the range from 1 to 2147483647.

reaction_type	<p>(Optional) Specifies the reaction to be taken for the specified IP SLAs operation.</p> <p>Type of IP SLAs reaction—One of the following keywords can be specified: <b>connectionLoss</b>, <b>icpif</b>, <b>jitterAvg</b>, <b>jitterDSAvg</b>, <b>jitterSDAvg</b>, <b>maxOfNegativeDS</b>, <b>maxOfNegativeSD</b>, <b>maxOfPositiveDS</b>, <b>maxOfPositiveSD</b>, <b>mos</b>, <b>packetLateArrival</b>, <b>packetLossDS</b>, <b>packetLossSD</b>, <b>packetMIA</b>, <b>packetOutOfSequence</b>, <b>rtt</b>, <b>timeout</b> or <b>verifyError</b> can be specified.</p> <p>Type of IP SLAs reaction. One of the following keywords can be specified:</p> <ul style="list-style-type: none"> <li>• connectionLoss</li> <li>• icpif</li> <li>• jitterAvg</li> <li>• jitterDSAvg</li> <li>• jitterSDAvg</li> <li>• maxOfNegativeDS</li> <li>• maxOfNegativeSD</li> <li>• maxOfPositiveDS</li> <li>• maxOfPositiveSD</li> <li>• mos</li> <li>• packetLateArrival</li> <li>• packetLossDS</li> <li>• packetLossSD</li> <li>• packetMIA</li> <li>• packetOutOfSequence</li> <li>• rtt</li> <li>• timeout</li> <li>• verifyError</li> </ul>
dest_ip_address	<p>(Optional) Specifies the destination IP address of the destination port for which the IP SLAs events are monitored.</p>

queue_priority	<p>(Optional) Priority level at which the script will be queued:</p> <ul style="list-style-type: none"> <li>• queue_priority low—Specifies that the script is to be queued at the lowest of the three priority levels.</li> <li>• queue_priority normal—Specifies that the script is to be queued at a priority level greater than low priority but less than high priority.</li> <li>• queue_priority high—Specifies that the script is to be queued at the highest of the three priority levels.</li> <li>• queue_priority last—Specifies that the script is to be queued at the lowest priority level.</li> </ul> <p>If more than one script is registered with the “queue_priority_last” argument set, these scripts will execute in the order in which the events are published.</p> <p><b>Note</b> The queue_priority argument specifies the queuing priority, but not the execution priority, of the script being registered.</p> <p>If this argument is not specified, the default queuing priority is normal.</p>
maxrun	<p>(Optional) Maximum run time of the script (specified in SSSSSSSSS[.MMM] format, where SSSSSSSSS must be an integer representing seconds between 0 and 31536000, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the default 20-second run-time limit is used.</p>
nice	<p>(Optional) Policy run-time priority setting. When the nice argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.</p>

**Result String**

None

**Set\_cerrno**

No

# event\_register\_nf

Registers for an event when a NetFlow event is triggered by the **event nf** command. Use this Tcl command to publish an event when a NetFlow reaction is triggered..

### Syntax

```
event_register_nf [tag ?] monitor_name ? event_type create|update|delete
exit_event_type create|update|delete event1-event4 ? [maxrun ?] [nice 0|1]
```

### Arguments

tag	(Optional) String identifying a tag that can be used with the trigger Tcl command extension to support multiple event statements within a Tcl script.
monitor_name	(Mandatory) The name of the NetFlow monitor.
event_type	(Mandatory) The type of event to monitor for the create, update, and delete flow.
exit_event_type	(Mandatory) The event-type (create, delete, update) at which the event is rearmed to be monitored again.
event1- event4	(Mandatory) Specifies the event and its attributes to monitor. Valid values are <b>event1</b> , <b>event2</b> , <b>event3</b> , and <b>event4</b> .  The subevent keywords can be used alone, together, or in any combination with each other, but each keyword can be used only once.
maxrun	(Optional) Maximum run time of the script (specified in SSSSSSSSS[.MMM] format, where SSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the default 20-second run-time limit is used.
nice	(Optional) Policy run-time priority setting. When the nice argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.

### Subevent Syntax

```
field ? rate_interval ? event1 only entry_value ? entry_op eq|ge|gt|le|lt|wc
[exit_value ?] [exit_op eq|ge|gt|le|lt|wc] [exit_rate_interval ? event1 only]
```

## Subevent Arguments

field	<p>(Mandatory) Specifies the cache or field attribute to be monitored. One of the following attributes can be specified:</p> <ul style="list-style-type: none"> <li>• <b>counter</b> {<b>bytes</b>   <b>packets</b>}—Specifies the counter fields.</li> <li>• <b>datalink</b> {<b>dot1q</b>   <b>mac</b>}—Specifies the datalink (layer2) fields.</li> <li>• <b>flow</b> {<b>direction</b>   <b>sampler</b>}—Specifies the flow identifying fields.</li> <li>• <b>interface</b> {<b>input</b>   <b>output</b>}—Specifies the interface fields.</li> <li>• <b>ipv4</b> <i>field-type</i>—Specifies the IPv4 fields.</li> <li>• <b>ipv6</b> <i>field-type</i>—IPv6 fields</li> <li>• <b>routing</b> <i>routing-attribute</i>—Specifies the routing attributes.</li> <li>• <b>timestamp sysuptime</b> {<b>first</b>   <b>last</b>}—Specifies the timestamp fields.</li> <li>• <b>transport</b> <i>field-type</i>—Specifies the Transport layer fields.</li> </ul>
rate_interval	(Mandatory) Specifies the rate interval value in seconds used to calculate the rate. This field is only valid for event1.
entry_value	(Mandatory) Specifies the field or rate value.
entry_op	<p>(Mandatory) Specifies the field operator.</p> <p>The comparison operator valid values are:</p> <ul style="list-style-type: none"> <li>• <b>eq</b> - Equal to</li> <li>• <b>ge</b> - Greater than or equal to</li> <li>• <b>gt</b> - Greater than</li> <li>• <b>le</b> - Less than or equal to</li> <li>• <b>lt</b> - Less than</li> <li>• <b>wc</b> - Wildcard</li> </ul>
exit_value	(Optional) The value at which the event is rearmed to be monitored again.

exit_op	<p>(Optional) The comparison operator used to compare the current event field or rate value with the exit value; if true, event monitoring for this event is reenabled.</p> <p>The comparison operator valid values are:</p> <ul style="list-style-type: none"><li>• <b>eq</b> - Equal to</li><li>• <b>ge</b> - Greater than or equal to</li><li>• <b>gt</b> - Greater than</li><li>• <b>le</b> - Less than or equal to</li><li>• <b>lt</b> - Less than</li><li>• <b>wc</b> - Wildcard</li></ul>
exit_rate_interval	<p>(Optional) Specifies the exit rate interval value in seconds used to calculate the exit rate value. This field is only valid for event1.</p>

**Result String**

None

**Set\_cerrno**

No

## event\_register\_none

Registers for an event that is triggered by the **event manager run** command. These events are handled by the None event detector that screens for this event.

### Syntax

```
event_register_none [tag ?] [queue_priority low|normal|high|last] [maxrun ?] [nice 0|1]
```

### Arguments

tag	(Optional) String identifying a tag that can be used with the trigger Tcl command extension to support multiple event statements within a Tcl script.
queue_priority	<p>(Optional) Priority level at which the script will be queued:</p> <ul style="list-style-type: none"> <li>queue_priority low—Specifies that the script is to be queued at the lowest of the three priority levels.</li> <li>queue_priority normal—Specifies that the script is to be queued at a priority level greater than low priority but less than high priority.</li> <li>queue_priority high—Specifies that the script is to be queued at the highest of the three priority levels.</li> <li>queue_priority last—Specifies that the script is to be queued at the lowest priority level.</li> </ul> <p>If more than one script is registered with the “queue_priority_last” argument set, these scripts will execute in the order in which the events are published.</p> <p><b>Note</b> The queue_priority argument specifies the queuing priority, but not the execution priority, of the script being registered.</p> <p>If this argument is not specified, the default queuing priority is normal.</p>
maxrun	(Optional) Maximum run time of the script (specified in SSSSSSSSS[.MMM] format, where SSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the default 20-second run-time limit is used.
nice	(Optional) Policy run-time priority setting. When the nice argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.

### Result String

None

### Set\_cerrno

No



# event\_register\_oir

Registers for an online insertion and removal (OIR) event. Use this Tcl command extension to run a policy on the basis of an event raised when a hardware card OIR occurs. These events are handled by the OIR event detector that screens for this event.

### Syntax

```
event_register_oir [tag ?] [queue_priority low|normal|high|last] [maxrun ?] [nice 0|1]
```

### Arguments

tag	(Optional) String identifying a tag that can be used with the trigger Tcl command extension to support multiple event statements within a Tcl script.
queue_priority	<p>(Optional) Priority level at which the script will be queued:</p> <ul style="list-style-type: none"> <li>• queue_priority low—Specifies that the script is to be queued at the lowest of the three priority levels.</li> <li>• queue_priority normal—Specifies that the script is to be queued at a priority level greater than low priority but less than high priority.</li> <li>• queue_priority high—Specifies that the script is to be queued at the highest of the three priority levels.</li> <li>• queue_priority last—Specifies that the script is to be queued at the lowest priority level.</li> </ul> <p>If more than one script is registered with the “queue_priority_last” argument set, these scripts will execute in the order in which the events are published.</p> <p><b>Note</b> The queue_priority argument specifies the queuing priority, but not the execution priority, of the script being registered.</p> <p>If this argument is not specified, the default queuing priority is normal.</p>
maxrun	(Optional) Maximum run time of the script (specified in SSSSSSSSS[.MMM] format, where SSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the default 20-second run-time limit is used.
nice	(Optional) Policy run-time priority setting. When the nice argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.

### Result String

None

**Set\_cerrno**

No

## event\_register\_process

Registers for a process event. Use this Tcl command extension to run a policy on the basis of an event raised when a Cisco IOS Software Modularity process starts or stops. These events are handled by the System Manager event detector that screens for this event. This Tcl command extension is supported only in Software Modularity images.

### Syntax

```
event_register_process [tag ?] abort|term|start|user_restart|user_shutdown
[sub_system ?] [version ?] [instance ?] [path ?] [node ?]
[queue_priority low|normal|high|last] [maxrun ?] [nice 0|1]
```

### Arguments

tag	(Optional) String identifying a tag that can be used with the trigger Tcl command extension to support multiple event statements within a Tcl script.
abort	(Mandatory) Abnormal process termination. Process may abort because of exiting with a nonzero exit status, receiving a kernel-generated signal, or receiving a SIGTERM or SIGKILL signal that is not sent because of user request.
term	(Mandatory) Normal process termination.
start	(Mandatory) Process start.
user_restart	(Mandatory) Process termination due to the process restart request from the CLI command.
user_shutdown	(Mandatory) Process termination due to the process kill request from the CLI command.
sub_system	(Optional) Number assigned to the EEM policy that published the process event. Number is set to 798 because all other numbers are reserved for Cisco use.
version	(Optional) Version number of the process assigned by the version manager. Must be of the form major_number.minor_number.level. If specified, each component of the version number must be an integer between 1 and 4294967295, inclusive.
instance	(Optional) Process instance ID. If specified, this argument must be an integer between 1 and 4294967295, inclusive.
path	(Optional) Process pathname (a regular expression string). If the value of the process-name argument contains embedded blanks, enclose it in double quotation marks. Use path “.*” to match all processes.

node	<p>(Optional) The node name is a string that consists of the word “node” followed by two fields separated by a slash character using the following format:</p> <pre>node&lt;slot-number&gt;/&lt;cpu-number&gt;</pre> <p>The slot-number is the hardware slot number. The cpu-number is the hardware CPU number. For example, the SP CPU in a Supervisor card on a Cisco Catalyst 6500 series switch located in slot 0 would be specified as node0/0. The RP CPU in a Supervisor card on a Cisco Catalyst 6500 series switch located in slot 0 would be addressed as node0/1. If the node argument is not specified, the default node specification is always the regular expression pattern match of * representing all applicable nodes.</p>
queue_priority	<p>(Optional) Priority level at which the script will be queued:</p> <ul style="list-style-type: none"> <li>• queue_priority low—Specifies that the script is to be queued at the lowest of the three priority levels.</li> <li>• queue_priority normal—Specifies that the script is to be queued at a priority level greater than low priority but less than high priority.</li> <li>• queue_priority high—Specifies that the script is to be queued at the highest of the three priority levels.</li> <li>• queue_priority last—Specifies that the script is to be queued at the lowest priority level.</li> </ul> <p>If more than one script is registered with the “queue_priority_last” argument set, these scripts will execute in the order in which the events are published.</p> <p><b>Note</b> The queue_priority argument specifies the queuing priority, but not the execution priority, of the script being registered.</p> <p>If this argument is not specified, the default queuing priority is normal.</p>
maxrun	<p>(Optional) Maximum run time of the script (specified in SSSSSSSSS[.MMM] format, where SSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the default 20-second run-time limit is used.</p>
nice	<p>(Optional) Policy run-time priority setting. When the nice argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.</p>

If an optional argument is not specified, the event matches all possible values of the argument. If multiple arguments are specified, the process event will be raised when all the conditions are matched.

#### Result String

None

#### Set\_cerrno

No

# event\_register\_resource

Registers for an Embedded Resource Manager (ERM) event. Use this Tcl command extension to run a policy on the basis of an ERM event report for a specified policy. ERM events are screened by the EEM Resource event detector, allowing an EEM policy to be run when a match occurs for the specified ERM policy.

### Syntax

```
event_register_resource policy policy-name [queue_priority low|normal|high|last]
[maxrun ?] [nice 0|1]
```

### Arguments

policy	(Mandatory) Specifies the use of a policy.
policy-name	(Mandatory) Name of an ERM policy.
queue_priority	<p>(Optional) Priority level at which the script will be queued:</p> <ul style="list-style-type: none"> <li>queue_priority low—Specifies that the script is to be queued at the lowest of the three priority levels.</li> <li>queue_priority normal—Specifies that the script is to be queued at a priority level greater than low priority but less than high priority.</li> <li>queue_priority high—Specifies that the script is to be queued at the highest of the three priority levels.</li> <li>queue_priority last—Specifies that the script is to be queued at the lowest priority level.</li> </ul> <p>If more than one script is registered with the “queue_priority_last” argument set, these scripts will execute in the order in which the events are published.</p> <p><b>Note</b> The queue_priority argument specifies the queuing priority, but not the execution priority, of the script being registered.</p> <p>If this argument is not specified, the default queuing priority is normal.</p>
maxrun	(Optional) Maximum run time of the script (specified in SSSSSSSSS[.MMM] format, where SSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the default 20-second run-time limit is used.
nice	(Optional) Policy run-time priority setting. When the nice argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.

### Result String

None

**Set\_cerrno**

No

# event\_register\_rf

Registers for a Redundancy Facility (RF) event. Use this Tcl command extension to run a policy when an RF progression or status event notification occurs.

### Syntax

```
event_register_rf [tag ?] event ?
[queue_priority low|normal|high|last]
[maxrun ?] [nice 0|1]
```

### Arguments

tag	(Optional) String identifying a tag that can be used with the trigger Tcl command extension to support multiple event statements within a Tcl script.
event	(Mandatory) Name of the RF progression or status event. Valid values are: <ul style="list-style-type: none"> <li>• RF_PROG_ACTIVE</li> <li>• RF_PROG_ACTIVE_DRAIN</li> <li>• RF_PROG_ACTIVE_FAST = 200</li> <li>• RF_PROG_ACTIVE_PRECONFIG</li> <li>• RF_PROG_ACTIVE_POSTCONFIG</li> <li>• RF_PROG_EXTRALOAD</li> <li>• RF_PROG_HANDBACK</li> <li>• RF_PROG_INITIALIZATION</li> <li>• RF_PROG_PLATFORM_SYNC</li> <li>• RF_PROG_STANDBY_BULK</li> <li>• RF_PROG_STANDBY_COLD</li> <li>• RF_PROG_STANDBY_CONFIG</li> <li>• RF_PROG_STANDBY_FILESYS</li> <li>• RF_PROG_STANDBY_HOT</li> <li>• RF_PROG_STANDBY_OIR_SYNC_DONE</li> <li>• RF_REGISTRATION_STATUS</li> <li>• RF_STATUS_MAINTENANCE_ENABLE</li> <li>• RF_STATUS_MANUAL_SWACT</li> <li>• RF_STATUS_OPER_REDUNDANCY_MODE_CHANGE</li> <li>• RF_STATUS_PEER_COMM</li> <li>• RF_STATUS_PEER_PRESENCE</li> <li>• RF_STATUS_REDUNDANCY_MODE_CHANGE</li> <li>• RF_STATUS_SWACT_INHIBIT</li> </ul>

queue_priority	<p>(Optional) Priority level at which the script will be queued:</p> <ul style="list-style-type: none"> <li>• queue_priority low—Specifies that the script is to be queued at the lowest of the three priority levels.</li> <li>• queue_priority normal—Specifies that the script is to be queued at a priority level greater than low priority but less than high priority.</li> <li>• queue_priority high—Specifies that the script is to be queued at the highest of the three priority levels.</li> <li>• queue_priority last—Specifies that the script is to be queued at the lowest priority level.</li> </ul> <p>If more than one script is registered with the “queue_priority_last” argument set, these scripts will execute in the order in which the events are published.</p> <p><b>Note</b> The queue_priority argument specifies the queuing priority, but not the execution priority, of the script being registered.</p> <p>If this argument is not specified, the default queuing priority is normal.</p>
maxrun	<p>(Optional) Maximum run time of the script (specified in SSSSSSSSS[.MMM] format, where SSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the default 20-second run-time limit is used.</p>
nice	<p>(Optional) Policy run-time priority setting. When the nice argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.</p>

**Result String**

None

**Set\_cerrno**

No



## event\_register\_routing

Registers for an event that is triggered by the **event routing** command. These events are handled by the routing event detector to publish an event when route entries change in Routing Information Base (RIB) infrastructure. Use this Tcl command extension to run a routing policy for this script. The network IP address for the route to be monitored must be specified.

### Syntax

```
event_register_routing [tag ?] network ? length [ge|le|ne] [type add|remove|modify|all]
[protocol ?] [queue_priority normal|low|high|last] [maxrun ?] [nice {0 | 1}]
```

### Arguments

tag	(Optional) String identifying a tag that can be used with the trigger Tcl command extension to support multiple event statements within a Tcl script.
network	Specifies the network IP address. The network number can be any valid IP address or prefix.
length	<p>Specifies the length of the network mask in bits. The bit mask can be a number from 0 to 32.</p> <ul style="list-style-type: none"> <li>• <b>ge</b>—(Optional) Specifies the minimum prefix length to be matched. The <b>ge</b> keyword represents greater than or equal to operator.</li> <li>• <b>le</b>—(Optional) Specifies the maximum prefix length to be matched. The <b>le</b> keyword represents the less than or equal to operator.</li> <li>• <b>ne</b>—(Optional) Specifies the prefix length not to be matched. The <b>ne</b> keyword represents not equal to operator.</li> </ul> <p>When <b>ge</b>, <b>le</b> and <b>ne</b> keywords are not configured, an exact match of network length is processed.</p>
type	(Optional) Specifies the desired policy trigger. The type options are <b>add</b> , <b>remove</b> , <b>modify</b> , and <b>all</b> . The default is <b>all</b> .
protocol	<p>(Optional) Specifies the protocol value for the network being monitored.</p> <p>One of the following protocols can be used: <b>all</b>, <b>bgp</b>, <b>connected</b>, <b>eigrp</b>, <b>isis</b>, <b>iso-igrp</b>, <b>mobile</b>, <b>odr</b>, <b>ospf</b>, <b>rip</b>, and <b>static</b>. The default is <b>all</b>.</p>

queue_priority	<p>(Optional) Priority level at which the script will be queued:</p> <ul style="list-style-type: none"> <li>• queue_priority low—Specifies that the script is to be queued at the lowest of the three priority levels.</li> <li>• queue_priority normal—Specifies that the script is to be queued at a priority level greater than low priority but less than high priority.</li> <li>• queue_priority high—Specifies that the script is to be queued at the highest of the three priority levels.</li> <li>• queue_priority last—Specifies that the script is to be queued at the lowest priority level.</li> </ul> <p>If more than one script is registered with the “queue_priority_last” argument set, these scripts will execute in the order in which the events are published.</p> <p><b>Note</b> The queue_priority argument specifies the queuing priority, but not the execution priority, of the script being registered.</p> <p>If this argument is not specified, the default queuing priority is normal.</p>
maxrun	<p>(Optional) Maximum run time of the script (specified in SSSSSSSSS[.MMM] format, where SSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the default 20-second run-time limit is used.</p>
nice	<p>(Optional) Policy run-time priority setting. When the nice argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.</p>

**Result String**

None

**Set\_cerrno**

No

# event\_register\_rpc

Registers for an event that is triggered by the EEM SSH Remote Procedure Call (RPC) command. These events are handled by the RPC event detector that screens for this event. Use this Tcl command extension to run a RPC policy for this script.

### Syntax

```
event_register_rpc [queue_priority {normal | low | high | last}] [maxrun <sec.msec>] [nice {0 | 1}] [default <sec.msec>]
```

### Arguments

queue_priority	<p>(Optional) Priority level at which the script will be queued:</p> <ul style="list-style-type: none"> <li>• queue_priority low—Specifies that the script is to be queued at the lowest of the three priority levels.</li> <li>• queue_priority normal—Specifies that the script is to be queued at a priority level greater than low priority but less than high priority.</li> <li>• queue_priority high—Specifies that the script is to be queued at the highest of the three priority levels.</li> <li>• queue_priority last—Specifies that the script is to be queued at the lowest priority level.</li> </ul> <p>If more than one script is registered with the “queue_priority_last” argument set, these scripts will execute in the order in which the events are published.</p> <p><b>Note</b> The queue_priority argument specifies the queuing priority, but not the execution priority, of the script being registered.</p> <p>If this argument is not specified, the default queuing priority is normal.</p>
maxrun	<p>(Optional) Maximum run time of the script (specified in SSSSSSSSS[.MMM] format, where SSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the default 20-second run-time limit is used.</p>

nice	(Optional) Policy run-time priority setting. When the nice argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.
default	(Optional) The time period during which the CLI event detector waits for the policy to exit (specified in SSSSSSSSS[.MMM] format, where SSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If the default time period expires before the policy exits, the default action will be executed. The default action is to run the command. If this argument is not specified, the default time period is set to 30 seconds.

**Result String**

None

**Set\_cerrno**

No

## event\_register\_snmp

Registers for a Simple Network Management Protocol (SNMP) statistics event. Use this Tcl command extension to run a policy when a given counter specified by an SNMP object ID (oid) crosses a defined threshold.

### Syntax

```
event_register_snmp [tag ?] oid ? get_type exact|next
entry_op gt|ge|eq|ne|lt|le entry_val ?
entry_type value|increment|rate
[exit_comb or|and]
[exit_op gt|ge|eq|ne|lt|le] [exit_val ?]
[exit_type value|increment|rate]
[exit_time ?] poll_interval ? [average_factor ?]
[queue_priority low|normal|high|last]
[maxrun ?] [nice 0|1]
```

### Arguments

tag	(Optional) String identifying a tag that can be used with the trigger Tcl command extension to support multiple event statements within a Tcl script.
oid	(Mandatory) OID number of data element in SNMP dot notation (for example, 1.3.6.1.2.1.2.1.0). The types of OIDs allowed are: <ul style="list-style-type: none"> <li>COUNTER_TYPE</li> <li>COUNTER_64_TYPE</li> <li>GAUGE_TYPE</li> <li>INTEGER_TYPE</li> <li>OCTET_PRIM_TYPE</li> <li>OPAQUE_PRIM_TYPE</li> <li>TIME_TICKS_TYPE</li> </ul>
entry_op	(Mandatory) Entry comparison operator used to compare the current OID data value with the entry value; if true, an event will be raised and event monitoring will be disabled until exit criteria are met.
get_type	(Mandatory) Type of SNMP get operation that needs to be applied to the OID specified. If the get_type argument is “exact,” the value of the specified OID is retrieved; if the get_type argument is “next,” the value of the lexicographical successor to the specified OID is retrieved.
entry_val	(Mandatory) Value with which the current oid data value should be compared to decide if the SNMP event should be raised.

entry-type	<p>Specifies a type of operation to be applied to the object ID specified by the entry-val argument.</p> <p>Value is defined as the actual value of the entry-val argument.</p> <p>Increment uses the entry-val field as an incremental difference and the entry-val is compared with the difference between the current counter value and the value when the event was last triggered (or the first polled sample if this is a new event). A negative value checks the incremental difference for a counter that is decreasing.</p> <p>Rate is defined as the average rate of change over a period of time. The time period is the average-factor value multiplied by the poll-interval value. At each poll interval the difference between the current sample and the previous sample is taken and recorded as an absolute value. An average of the previous average-factor value samples is taken to be the rate of change.</p>
exit_comb	<p>(Optional) Exit combination operator used to indicate the combination of exit condition tests required to decide if the exit criteria are met so that the event monitoring can be reenabled. If it is “and,” both exit value and exit time tests must be passed to meet the exit criteria. If it is “or,” either exit value or exit time tests can be passed to meet the exit criteria.</p> <p>When exit_comb is “and,” exit_op, and exit_val (exit_time) must exist. When exit_comb is “or,” (exit_op and exit_val) or (exit_time) must exist.</p>
exit_op	<p>(Optional) Exit comparison operator used to compare the current oid data value with the exit value; if true, event monitoring for this event will be reenabled.</p>
exit_val	<p>(Optional) Value with which the current oid data value should be compared to decide if the exit criteria are met.</p>
exit-type	<p>(Optional) Specifies a type of operation to be applied to the object ID specified by the exit-val argument. If not specified, the value is assumed.</p> <p>Value is defined as the actual value of the exit-val argument.</p> <p>Increment uses the exit-val field as an incremental difference and the exit-val is compared with the difference between the current counter value and the value when the event was last triggered (or the first polled sample if this is a new event). A negative value checks the incremental difference for a counter that is decreasing.</p> <p>Rate is defined as the average rate of change over a period of time. The time period is the average-factor value multiplied by the poll-interval value. At each poll interval the difference between the current sample and the previous sample is taken and recorded as an absolute value. An average of the previous average-factor value samples is taken to be the rate of change.</p>
exit_time	<p>(Optional) Number of POSIX timer units after an event is raised when event monitoring will be enabled again. Specified in SSSSSSSSS[.MMM] format where SSSSSSSSS must be an integer number representing seconds between 0 and 4294967295, inclusive. MMM represents milliseconds and must be an integer number between 0 and 999.</p>

poll_interval	(Mandatory) Interval between consecutive polls in POSIX timer units. Currently the interval is forced to be at least 1 second (specified in SSSSSSSSS[.MMM] format, where SSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999).
average-factor	(Optional) Number in the range from 1 to 64 used to calculate the period used for rate-based calculations. The average-factor value is multiplied by the poll-interval value to derive the period in milliseconds. The minimum average factor value is 1.
queue_priority	<p>(Optional) Priority level at which the script will be queued:</p> <ul style="list-style-type: none"> <li>• queue_priority low—Specifies that the script is to be queued at the lowest of the three priority levels.</li> <li>• queue_priority normal—Specifies that the script is to be queued at a priority level greater than low priority but less than high priority.</li> <li>• queue_priority high—Specifies that the script is to be queued at the highest of the three priority levels.</li> <li>• queue_priority last—Specifies that the script is to be queued at the lowest priority level.</li> </ul> <p>If more than one script is registered with the “queue_priority_last” argument set, these scripts will execute in the order in which the events are published.</p> <p><b>Note</b> The queue_priority argument specifies the queuing priority, but not the execution priority, of the script being registered.</p> <p>If this argument is not specified, the default queuing priority is normal.</p>
maxrun	(Optional) Maximum run time of the script (specified in SSSSSSSSS[.MMM] format, where SSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the default 20-second run-time limit is used.
nice	(Optional) Policy run-time priority setting. When the nice argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.

**Result String**

None

**Set\_cerrno**

No

## event\_register\_snmp\_notification

Registers for a Simple Network Management Protocol (SNMP) notification trap event. Use this Tcl command extension to run a policy when an SNMP trap with the specified SNMP object ID (oid) is encountered on a specific interface or address. The **snmp-server manager** CLI command must be enabled for the SNMP notifications to work using Tcl policies.

### Syntax

```
event_register_snmp_notification [tag ?] oid ? oid_val ?
op {gt|ge|eq|ne|lt|le}
[maxrun ?]
[src_ip_address ?]
[dest_ip_address ?]
[queue_priority {normal|low|high|last}]
[maxrun ?]
[nice {0|1}]
[default ?]
[direction {incoming|outgoing}]
[msg_op {drop|send}]
```

### Arguments

tag	(Optional) String identifying a tag that can be used with the trigger Tcl command extension to support multiple event statements within a Tcl script.
oid	(Mandatory) OID number of the data element in SNMP dot notation (for example, 1.3.6.1.2.1.2.1.0). If the specified OID ends with a dot (.), then all OIDs that start with the OID number before the dot are matched. The types of OIDs allowed are: <ul style="list-style-type: none"> <li>• COUNTER_TYPE</li> <li>• COUNTER_64_TYPE</li> <li>• GAUGE_TYPE</li> <li>• INTEGER_TYPE</li> <li>• OCTET_PRIM_TYPE</li> <li>• OPAQUE_PRIM_TYPE</li> <li>• TIME_TICKS_TYPE</li> </ul>
oid_val	(Mandatory) OID value with which the current OID data value should be compared to decide if the SNMP event should be raised.
op	(Mandatory) Comparison operator used to compare the current OID data value with the SNMP Protocol Data Unit (PDU) OID data value; if this is true, an event is raised.
maxrun	(Optional) Maximum run time of the script (specified in sssssss[.mmm] format, where sssssss must be an integer representing seconds between 0 and 31536000, inclusive, and where mmm must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the default 20-second run-time limit is used.
src_ip_address	(Optional) Source IP address where the SNMP notification trap originates. The default is all; it is set to receive SNMP notification traps from all IP addresses.



dest_ip_address	(Optional) Destination IP address where the SNMP notification trap is sent. The default is all; it is set to receive SNMP traps from all destination IP addresses.
queue_priority	<p>(Optional) Priority level at which the script will be queued:</p> <ul style="list-style-type: none"> <li>• queue_priority low—Specifies that the script is to be queued at the lowest of the three priority levels.</li> <li>• queue_priority normal—Specifies that the script is to be queued at a priority level greater than low priority but less than high priority.</li> <li>• queue_priority high—Specifies that the script is to be queued at the highest of the three priority levels.</li> <li>• queue_priority last—Specifies that the script is to be queued at the lowest priority level.</li> </ul> <p>If more than one script is registered with the queue_priority_last argument set, these scripts will execute in the order in which the events are published.</p> <p><b>Note</b> The queue_priority argument specifies the queuing priority, but not the execution priority, of the script being registered.</p> <p>If this argument is not specified, the default queuing priority is normal.</p>
default	(Optional) Specifies the time period in seconds during which the snmp notification event detector waits for the policy to exit. The time period is specified in ssssssss[.mmm] format, where ssssssss must be an integer representing seconds between 0 and 4294967295 and mmm must be an integer representing milliseconds between 0 and 999.
nice	(Optional) Policy run-time priority setting. When the nice argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.
direction	(Optional) The direction of the incoming or outgoing SNMP trap or inform PDU to filter. The default value is incoming.
msg_op	(Optional) The action to be taken on the SNMP PDU (drop it or send it) once the event is triggered. The default value is send.

**Result String**

None

**Set\_cerrno**

No

## event\_register\_snmp\_object

Registers for a Simple Network Management Protocol (SNMP) object event. Use this Tcl command extension to replace the value when an SNMP with the specified SNMP-object ID (OID) is encountered on a specific interface or address.

### Syntax

```
event_register_snmp_object oid ?
type {int|uint|counter|counter64|gauge|ipv4||oid|string}
sync {yes|no}
skip {yes|no}
[istable {yes|no}]
[default ?]
[queue_priority {normal|low|high|last}]
[maxrun ?]
[nice {0|1}]
```

### Arguments

oid	(Mandatory) OID number of the data element in SNMP dot notation (for example, 1.3.6.1.2.1.2.1.0). If the specified OID ends with a dot (.), then all OIDs that start with the OID number before the dot are matched. The types of OIDs allowed are: <ul style="list-style-type: none"> <li>• COUNTER_TYPE</li> <li>• COUNTER_64_TYPE</li> <li>• GAUGE_TYPE</li> <li>• INTEGER_TYPE</li> <li>• OCTET_PRIM_TYPE</li> <li>• OPAQUE_PRIM_TYPE</li> <li>• TIME_TICKS_TYPE</li> </ul>
type	(Mandatory) OID value type.
sync	(Mandatory) A “yes” means that the EEM policy will be notified. If the applet set_exit_status or Tcl return value is 0, then SNMP will handle the request. If the return value is 1, SNMP will use the value provided by the policy for the get request and will not process the set request. A “no” means that EEM will not be notified and SNMP will handle the request.  Only one OID can be associated with a synchronous policy. However, multiple synchronous policies can be registered for the same OID.
skip	Mandatory if the sync argument is “no” and should not exist if the sync argument is “yes.” If the skip argument is “yes,” it means that SNMP will handle the request. If the skip argument is “no,” it means that SNMP will act as if the object does not exist.
istable	(Optional) A value of “no” means the OID is scalar object, and “yes” means the OID is table object.

default	(Optional) The time period during which the SNMP Object event detector waits for the policy to exit (specified in ssssssss[.mmm] format, where ssssssss must be an integer representing seconds between 0 and 4294967295, inclusive, and where mmm must be an integer representing milliseconds between 0 and 999). If the default time period expires before the policy exits, the default action will be executed. The default action is to process the set or get request normally by SNMP subsystem. If this argument is not specified, the default time period is set to 30 seconds.
maxrun	(Optional) Maximum run time of the script (specified in ssssssss[.mmm] format, where ssssssss must be an integer representing seconds between 0 and 31536000, inclusive, and where mmm must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the default 20-second run-time limit is used.
queue_priority	<p>(Optional) Priority level at which the script will be queued:</p> <ul style="list-style-type: none"> <li>• queue_priority low—Specifies that the script is to be queued at the lowest of the three priority levels.</li> <li>• queue_priority normal—Specifies that the script is to be queued at a priority level greater than low priority but less than high priority.</li> <li>• queue_priority high—Specifies that the script is to be queued at the highest of the three priority levels.</li> <li>• queue_priority last—Specifies that the script is to be queued at the lowest priority level.</li> </ul> <p>If more than one script is registered with the queue_priority_last argument set, these scripts will execute in the order in which the events are published.</p> <p><b>Note</b> The queue_priority argument specifies the queuing priority, but not the execution priority, of the script being registered.</p> <p>If this argument is not specified, the default queuing priority is normal.</p>
nice	(Optional) Policy run-time priority setting. When the nice argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.

**Result String**

None

**Set\_cerrno**

No

## event\_register\_syslog

Registers for a syslog event. Use this Tcl command extension to trigger a policy when a syslog message of a specific pattern is logged after a certain number of occurrences during a certain period of time.

### Syntax

```
event_register_syslog [tag ?] [occurs ?] [period ?] pattern ?
[priority all|emergencies|alerts|critical|errors|warnings|notifications|
informational|debugging|0|1|2|3|4|5|6|7]
[queue_priority low|normal|high|last]
[severity_fatal] [severity_critical] [severity_major]
[severity_minor] [severity_warning] [severity_notification]
[severity_normal] [severity_debugging]
[maxrun ?] [nice 0|1]
```

### Arguments

tag	(Optional) String identifying a tag that can be used with the trigger Tcl command extension to support multiple event statements within a Tcl script.
occurs	(Optional) Number of occurrences before the event is raised; if not specified, the event is raised on the first occurrence. If specified, the value must be greater than 0.
period	(Optional) Time interval, in seconds and milliseconds, during which the one or more occurrences must take place in order to raise an event (specified in SSSSSSSSS[.MMM] format where SSSSSSSSS must be an integer number representing seconds between 0 and 4294967295, inclusive, and where MMM represents milliseconds and must be an integer number between 0 and 999). If this argument is not specified, no period check is applied.
pattern	(Mandatory) A regular expression used to perform syslog message pattern match. This argument is what the policy uses to identify the logged syslog message.
priority	(Optional) The message priority to be screened. If this argument is specified, only messages that are at the specified logging priority level, or lower, are screened. If this argument is not specified, the default priority is 0.

queue_priority	<p>(Optional) Priority level at which the script will be queued:</p> <ul style="list-style-type: none"> <li>• queue_priority low—Specifies that the script is to be queued at the lowest of the three priority levels.</li> <li>• queue_priority normal—Specifies that the script is to be queued at a priority level greater than low priority but less than high priority.</li> <li>• queue_priority high—Specifies that the script is to be queued at the highest of the three priority levels.</li> <li>• queue_priority last—Specifies that the script is to be queued at the lowest priority level.</li> </ul> <p>If more than one script is registered with the “queue_priority_last” argument set, these scripts will execute in the order in which the events are published.</p> <p><b>Note</b> The queue_priority argument specifies the queuing priority, but not the execution priority, of the script being registered.</p> <p>If this argument is not specified, the default queuing priority is normal.</p>
maxrun	<p>(Optional) Maximum run time of the script (specified in SSSSSSSSS[.MMM] format, where SSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the default 20-second run-time limit is used.</p>
nice	<p>(Optional) Policy run-time priority setting. When the nice argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.</p>
severity_xxx	<p>(Optional) The event severity to be screened. If this argument is specified, only messages that are at the specified severity level are screened. See <a href="#">Table 15</a> for the severity level mapping for syslog events.</p>

If multiple conditions are specified, the syslog event will be raised when all the conditions are matched.

**Table 15** Severity Level Mapping For Syslog Events

Severity Keyword	Syslog Priority	Description
severity_fatal	LOG_EMERG (0)	System is unusable.
severity_critical	LOG_ALERT (1)	Critical conditions, immediate attention required.
severity_major	LOG_CRIT (2)	Major conditions.
severity_minor	LOG_ERR (3)	Minor conditions.
severity_warning	LOG_WARNING (4)	Warning conditions.
severity_notification	LOG_NOTICE (5)	Basic notification, informational messages.

**Table 15**      **Severity Level Mapping For Syslog Events**

severity_normal	LOG_INFO (6)	Normal event, indicates returning to a normal state.
severity_debugging	LOG_DEBUG (7)	Debugging messages.

**Result String**

None

**Set\_cerrno**

No

## event\_register\_timer

Creates a timer and registers for a timer event as both a publisher and a subscriber. Use this Tcl command extension when there is a need to trigger a policy that is time specific or timer based. This event timer is both an event publisher and a subscriber. The publisher part indicates the conditions under which the named timer is to go off. The subscriber part identifies the name of the timer to which the event is subscribing.



### Note

Both the CRON and absolute time specifications work on local time.

### Syntax

```
event_register_timer [tag ?] watchdog|countdown|absolute|cron
[name ?] [cron_entry ?]
[time ?]
[queue_priority low|normal|high|last] [maxrun ?]
[nice 0|1]
```

### Arguments

tag	(Optional) String identifying a tag that can be used with the trigger Tcl command extension to support multiple event statements within a Tcl script.
watchdog	(Mandatory) Watchdog timer.
countdown	(Mandatory) Countdown timer.
absolute	(Mandatory) Absolute timer.
cron	(Mandatory) CRON timer.
name	(Optional) Name of the timer.

cron_entry	<p>(Optional) Must be specified if the CRON timer type is specified. Must not be specified if any other timer type is specified. A cron_entry is a partial UNIX crontab entry (the first five fields) as used with the UNIX CRON daemon.</p> <p>A cron_entry specification consists of a text string with five fields. The fields are separated by spaces. The fields represent the time and date when CRON timer events will be triggered. The fields are described in <a href="#">Table 16</a>.</p> <p>Ranges of numbers are allowed. Ranges are two numbers separated with a hyphen. The specified range is inclusive. For example, 8-11 for an hour entry specifies execution at hours 8, 9, 10, and 11.</p> <p>A field may be an asterisk (*), which always stands for “first-last.”</p> <p>Lists are allowed. A list is a set of numbers (or ranges) separated by commas. Examples: “1,2,5,9” and “0-4,8-12”.</p> <p>Step values can be used in conjunction with ranges. Following a range with “/&lt;number&gt;” specifies skips of the number’s value through the range. For example, “0-23/2” can be used in the hour field to specify an event that is triggered every other hour. Steps are also permitted after an asterisk, so if you want to say “every two hours”, use “*/2”.</p> <p>Names can also be used for the month and the day of week fields. Use the first three letters of the particular day or month (case does not matter). Ranges or lists of names are not allowed.</p> <p>The day on which a timer event is triggered can be specified by two fields: day of month and day of week. If both fields are restricted (that is, are not *), an event will be triggered when either field matches the current time. For example, “30 4 1,15 * 5” would cause an event to be triggered at 4:30 a.m. on the 1st and 15th of each month, plus every Friday.</p> <p>Instead of the first five fields, one of seven special strings may appear. These seven special strings are described in <a href="#">Table 17</a>.</p> <p>Example 1: “0 0 1,15 * 1” would trigger an event at midnight on the 1st and 15th of each month, as well as on every Monday. To specify days by only one field, the other field should be set to *; “0 0 * * 1” would trigger an event at midnight only on Mondays.</p> <p>Example 2: “15 16 1 * *” would trigger an event at 4:15 p.m. on the first day of each month.</p> <p>Example 3: “0 12 * * 1-5” would trigger an event at noon on Monday through Friday of each week.</p> <p>Example 4: “@weekly” would trigger an event at midnight once a week on Sunday.</p>
time	<p>(Optional) Must be specified if a timer type other than CRON is specified. Must not be specified if the CRON timer type is specified. For watchdog and countdown timers, the number of seconds and milliseconds until the timer expires; for the absolute timer, the calendar time of the expiration time. Time is specified in SSSSSSSSS[.MMM] format, where SSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999. An absolute expiration date is the number of seconds and milliseconds since January 1, 1970. If the date specified has already passed, the timer expires immediately.</p>



queue_priority	<p>(Optional) Priority level at which the script will be queued:</p> <ul style="list-style-type: none"> <li>queue_priority low—Specifies that the script is to be queued at the lowest of the three priority levels.</li> <li>queue_priority normal—Specifies that the script is to be queued at a priority level greater than low priority but less than high priority.</li> <li>queue_priority high—Specifies that the script is to be queued at the highest of the three priority levels.</li> <li>queue_priority last—Specifies that the script is to be queued at the lowest priority level.</li> </ul> <p>If more than one script is registered with the “queue_priority_last” argument set, these scripts will execute in the order in which the events are published.</p> <p><b>Note</b> The queue_priority argument specifies the queuing priority, but not the execution priority, of the script being registered.</p> <p>If this argument is not specified, the default queuing priority is normal.</p>
maxrun	<p>(Optional) Maximum run time of the script (specified in SSSSSSSSS[.MMM] format, where SSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the default 20-second run-time limit is used.</p>
nice	<p>(Optional) Policy run-time priority setting. When the nice argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.</p>

**Table 16** Time and Date When CRON Events Will Be Triggered

Field	Allowed Values
minute	0-59
hour	0-23
day of month	1-31
month	1-12 (or names, see below)
day of week	0-7 (0 or 7 is Sun, or names; see <a href="#">Table 17</a> )

**Table 17** Special Strings for cron\_entry

String	Meaning
@yearly	Trigger once a year, “0 0 1 1 *”.
@annually	Same as @yearly.
@monthly	Trigger once a month, “0 0 1 * *”.
@weekly	Trigger once a week, “0 0 * * 0”.
@daily	Trigger once a day, “0 0 * * *”.

**Table 17**      **Special Strings for cron\_entry**

@midnight	Same as @daily.
@hourly	Trigger once an hour, "0 * * * *".

**Result String**

None

**Set\_cerrno**

No

**See Also**

event\_register\_timer\_subscriber

## event\_register\_timer\_subscriber

Registers for a timer event as a subscriber. Use this Tcl command extension to identify the name of the timer to which the event timer, as a subscriber, wants to subscribe. The event timer depends on another policy or another process to actually manipulate the timer. For example, let policyB act as a timer subscriber policy, but policyA (although it does not need to be a timer policy) uses register\_timer, timer\_arm, or timer\_cancel Tcl command extensions to manipulate the timer referenced in policyB.

### Syntax

```
event_register_timer_subscriber watchdog|countdown|absolute|cron
name ? [queue_priority low|normal|high|last] [maxrun ?] [nice 0|1]
```

### Arguments

watchdog	(Mandatory) Watchdog timer.
countdown	(Mandatory) Countdown timer.
absolute	(Mandatory) Absolute timer.
cron	(Mandatory) CRON timer.
name	(Mandatory) Name of the timer.
queue_priority	<p>(Optional) Priority level at which the script will be queued:</p> <ul style="list-style-type: none"> <li>queue_priority low—Specifies that the script is to be queued at the lowest of the three priority levels.</li> <li>queue_priority normal—Specifies that the script is to be queued at a priority level greater than low priority but less than high priority.</li> <li>queue_priority high—Specifies that the script is to be queued at the highest of the three priority levels.</li> <li>queue_priority last—Specifies that the script is to be queued at the lowest priority level.</li> </ul> <p>If more than one script is registered with the “queue_priority_last” argument set, these scripts will execute in the order in which the events are published.</p> <p><b>Note</b> The queue_priority argument specifies the queuing priority, but not the execution priority, of the script being registered.</p> <p>If this argument is not specified, the default queuing priority is normal.</p>
maxrun	(Optional) Maximum run time of the script (specified in SSSSSSSSS[.MMM] format, where SSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the default 20-second run-time limit is used.
nice	(Optional) Policy run-time priority setting. When the nice argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.

**Note**

---

An EEM policy that registers for a timer event or a counter event can act as both publisher and subscriber.

---

**Result String**

None

**Set\_cerrno**

No

**See Also**

**event\_register\_timer**

## event\_register\_track

Registers for a report event from the Cisco IOS Object Tracking subsystem. Use this Tcl command extension to trigger a policy on the basis of a Cisco IOS Object Tracking subsystem report for a specified object number.

### Syntax

```
event_register_track ? [tag ?] [state up|down|any] [queue_priority low|normal|high|last]
[maxrun ?]
[nice 0|1]
```

### Arguments

? (represents a number)	(Mandatory) Tracked object number in the range from 1 to 500, inclusive.
tag	(Optional) String identifying a tag that can be used with the trigger Tcl command extension to support multiple event statements within a Tcl script.
state	(Optional) Specifies that the tracked object transition will cause an event to be raised. If <b>up</b> is specified, an event will be raised when the tracked object transitions from a down state to an up state. If <b>down</b> is specified, an event will be raised when the tracked object transitions from an up state to a down state. If <b>any</b> is specified, an event will be raised when the tracked object transitions to or from any state.
queue_priority	<p>(Optional) Priority level at which the script will be queued:</p> <ul style="list-style-type: none"> <li>• queue_priority low—Specifies that the script is to be queued at the lowest of the three priority levels.</li> <li>• queue_priority normal—Specifies that the script is to be queued at a priority level greater than low priority but less than high priority.</li> <li>• queue_priority high—Specifies that the script is to be queued at the highest of the three priority levels.</li> <li>• queue_priority last—Specifies that the script is to be queued at the lowest priority level.</li> </ul> <p>If more than one script is registered with the “queue_priority_last” argument set, these scripts will execute in the order in which the events are published.</p> <p><b>Note</b> The queue_priority argument specifies the queuing priority, but not the execution priority, of the script being registered.</p> <p>If this argument is not specified, the default queuing priority is normal.</p>

maxrun	(Optional) Maximum run time of the script (specified in SSSSSSSSS[.MMM] format, where SSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the default 20-second run-time limit is used.
nice	(Optional) Policy run-time priority setting. When the nice argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.

If an optional argument is not specified, the event matches all possible values of the argument.

**Result String**

None

**Set\_cerrno**

No

# event\_register\_wdsysmon

Registers for a Watchdog system monitor event. Use this Tcl command extension to register for a composite event which is a combination of several subevents or conditions. For example, you can use this command to register for the combination of conditions wherein the CPU usage of a certain process is over 80 percent and the memory used by the process is greater than 50 percent of its initial allocation. This Tcl command extension is supported only in Software Modularity images.

## Syntax

```
event_register_wdsysmon [tag ?] [timewin ?]
[sub12_op and|or|andnot]
[sub23_op and|or|andnot]
[sub34_op and|or|andnot]
[sub1 subevent-description]
[sub2 subevent-description]
[sub3 subevent-description]
[sub4 subevent-description] [node ?]
[queue_priority low|normal|high|last]
[maxrun ?] [nice 0|1]
```

Each argument is position independent.



**Note**

Operator definitions: and (logical and operation), or (logical or operation), andnot (logical and not operation). For example, “sub12\_op and” is defined as raise an event when subevent 1 and subevent 2 are true; “sub23\_op or” is defined as raise an event when the condition specified in sub12\_op is true or subevent 3 is true. The logic can be diagrammed using:

```
if (((sub1 sub12_op sub2) sub23_op sub3) sub34_op sub4) is TRUE, raise event
```

## Arguments

tag	(Optional) String identifying a tag that can be used with the trigger Tcl command extension to support multiple event statements within a Tcl script.
timewin	(Optional) Time window within which all of the subevents have to occur in order for an event to be generated (specified in SSSSSSSSS[.MMM] format, where SSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999).
sub12_op	(Optional) Combination operator for comparison between subevent 1 and subevent 2.
sub23_op	(Optional) Combination operator for comparison between subevent 1 and 2 and subevent 3.
sub34_op	(Optional) Combination operator for comparison between subevent 1 and 2 and subevent 3 and subevent 4.
sub1	(Optional) Indicates that subevent 1 is specified.
subevent-description	(Optional) Syntax for the subevent.
sub2	(Optional) Indicates that subevent 2 is specified.
sub3	(Optional) Indicates that subevent 3 is specified.

sub4	(Optional) Indicates that subevent 4 is specified.
node	<p>(Optional) The node name to be monitored for deadlock conditions is a string that consists of the word “node” followed by two fields separated by a slash character using the following format:</p> <pre>node&lt;slot-number&gt;/&lt;cpu-number&gt;</pre> <p>The slot-number is the hardware slot number. The cpu-number is the hardware CPU number. For example, the SP CPU in a Supervisor card on a Cisco Catalyst 6500 series switch located in slot 0 would be specified as node0/0. The RP CPU in a Supervisor card on a Cisco Catalyst 6500 series switch located in slot 0 would be addressed as node0/1. If the node argument is not specified, the default node specification is the local node on which the registration is done.</p>
queue_priority	<p>(Optional) Priority level at which the script will be queued:</p> <ul style="list-style-type: none"> <li>• queue_priority low—Specifies that the script is to be queued at the lowest of the three priority levels.</li> <li>• queue_priority normal—Specifies that the script is to be queued at a priority level greater than low priority but less than high priority.</li> <li>• queue_priority high—Specifies that the script is to be queued at the highest of the three priority levels.</li> <li>• queue_priority last—Specifies that the script is to be queued at the lowest priority level.</li> </ul> <p>If more than one script is registered with the “queue_priority_last” argument set, these scripts will execute in the order in which the events are published.</p> <p><b>Note</b> The queue_priority argument specifies the queuing priority, but not the execution priority, of the script being registered.</p> <p>If this argument is not specified, the default queuing priority is normal.</p>
maxrun	<p>(Optional) Maximum run time of the script (specified in SSSSSSSSS[.MMM] format, where SSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the default 20-second run-time limit is used.</p>
nice	<p>(Optional) Policy run-time priority setting. When the nice argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.</p>

### Subevents

The syntax of subevent descriptions can be one of seven cases.

For arguments in subevent description, the following constraints apply on the value of number arguments:

- For dispatch\_mgr, val must be an integer between 0 and 4294967295, inclusive.
- For cpu\_proc and cpu\_tot, val must be an integer between 0 and 100, inclusive.
- For mem\_proc, mem\_tot\_avail, and mem\_tot\_used, if is\_percent is FALSE, val must be an integer between 0 and 4294967295, inclusive.



1. deadlock procname ?

**Arguments**

procname	(Mandatory) A regular expression that specifies the process name that you wish to monitor for deadlock conditions. This subevent will ignore the time window even if it is given.
----------	---

2. dispatch\_mgr [procname ?] [op gt|ge|eq|ne|lt|le] [val ?] [period ?]

**Arguments**

procname	(Optional) A regular expression that specifies the process name that you wish to monitor for dispatch_manager status.
op	(Optional) Comparison operator used to compare the collected number of events with the specified value; if true, an event will be raised.
val	(Optional) The value with which the number of events that have occurred should be compared.
period	(Optional) The time period for the number of events that have occurred (specified in SSSSSSSSS[.MMM] format, where SSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the most recent sample is used.

3. cpu\_proc [procname ?] [op gt|ge|eq|ne|lt|le] [val ?] [period ?]

**Arguments**

procname	(Optional) A regular expression that specifies the process name that you wish to monitor for CPU utilization conditions.
op	(Optional) Comparison operator used to compare the collected CPU usage sample percentage with the specified percentage value; if true, an event will be raised.
val	(Optional) The percentage value with which the average CPU usage during the sample period should be compared.
period	(Optional) The time period for averaging the collection of samples (specified in SSSSSSSSS[.MMM] format, where SSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the most recent sample is used.

4. cpu\_tot [op gt|ge|eq|ne|lt|le] [val ?] [period ?]

**Arguments**

op	(Optional) Comparison operator used to compare the collected total system CPU usage sample percentage with the specified percentage value; if true, an event will be raised.
----	--

val	(Optional) The percentage value with which the average CPU usage during the sample period should be compared.
period	(Optional) The time period for averaging the collection of samples (specified in SSSSSSSSS[.MMM] format, where SSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the most recent sample is used.

```
5. mem_proc [procname ?] [op gt|ge|eq|ne|lt|le] [val ?] [is_percent TRUE|FALSE] [period ?]
```

### Arguments

procname	(Optional) A regular expression that specifies the process name that you wish to monitor for memory usage.
op	(Optional) Comparison operator used to compare the collected memory used with the specified value; if true, an event will be raised.
val	(Optional) A percentage or an absolute value specified in kilobytes. A percentage represents the difference between the oldest sample in the specified time period and the latest sample. If memory usage has increased from 150 KB to 300 KB within the time period, the percentage increase is 100. This is the value with which the measured value should be compared.
is_percent	(Optional) If TRUE, the percentage value is collected and compared. Otherwise, the absolute value is collected and compared.
period	(Optional) If is_percent is set to TRUE, the time period for the percentage to be computed. Otherwise, the time period for the collection samples to be averaged (specified in SSSSSSSSS[.MMM] format, where SSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the most recent sample is used.

```
6. mem_tot_avail [op gt|ge|eq|ne|lt|le] [val ?] [is_percent TRUE|FALSE] [period ?]
```

### Arguments

op	(Optional) Comparison operator used to compare the collected available memory with the specified value; if true, an event will be raised.
val	(Optional) A percentage or an absolute value specified in kilobytes. A percentage represents the difference between the oldest sample in the specified time period and the latest sample. If available memory usage has decreased from 300 KB to 150 KB within the time period, the percentage decrease is 50. This is the value with which the measured value should be compared.
is_percent	(Optional) If TRUE, the percentage value is collected and compared. Otherwise, the absolute value is collected and compared.
period	(Optional) If is_percent is set to TRUE, the time period for the percentage to be computed. Otherwise, the time period for the collection samples to be averaged (specified in SSSSSSSSS[.MMM] format, where SSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the most recent sample is used.

7. mem\_tot\_used [op gt|ge|eq|ne|lt|le] [val ?] [is\_percent TRUE|FALSE] [period ?]

**Arguments**

op	(Optional) Comparison operator used to compare the collected used memory with the specified value; if true, an event will be raised.
val	(Optional) A percentage or an absolute value specified in kilobytes. A percentage represents the difference between the oldest sample in the specified time period and the latest sample. If memory usage has increased from 150 KB to 300 KB within the time period, the percentage increase is 100. This is the value with which the measured value should be compared.
is_percent	(Optional) If TRUE, the percentage value is collected and compared. Otherwise, the absolute value is collected and compared.
period	(Optional) If is_percent is set to TRUE, the time period for the percentage to be computed. Otherwise, the time period for the collection samples to be averaged (specified in SSSSSSSSS[.MMM] format, where SSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the most recent sample is used.  <b>Note</b> This argument is mandatory if is_percent is set to TRUE; otherwise, it is optional.

**Result String**

None

**Set\_cernno**

No



**Note**

Inside a subevent description, each argument is position independent.

## EEM Event Information Tcl Command Extension

- [event\\_reqinfo](#), page 125

# event\_reqinfo

Queries information for the event that caused the current policy to run.

## Syntax

```
event_reqinfo
```

## Arguments

None

## Result String

If the policy runs successfully, the characteristics for the event that triggered the policy will be returned. The following sections show the characteristics returned for each event detector.

### For EEM\_EVENT\_APPLICATION

```
"event_id %u event_type %u event_type_string {%s} event_pub_sec %u event_pub_msec %u"
"sub_system 0x%x type %u data1 {%s} data2 {%s} data3 {%s} data4 {%s}"
```

Event Type	Description
<b>event_id</b>	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
<b>event_type</b>	Type of event.
<b>event_type_string</b>	An ASCII string that represents the name of the event for this event type.
<b>event_pub_sec</b> <b>event_pub_msec</b>	The time, in seconds and milliseconds, when the event was published to the Embedded Event Manager (EEM).
<b>sub_system</b>	Number assigned to the EEM policy that published the application event. Number is set to 798 because all other numbers are reserved for Cisco use.
<b>type</b>	Event subtype within the specified component.
<b>data1</b> <b>data2</b> <b>data3</b> <b>data4</b>	Argument data that is passed to the application-specific event when the event is published. The data is character text, an environment variable, or a combination of the two.

### For EEM\_EVENT\_CLI

```
"event_id %u event_type %u event_type_string {%s} event_pub_sec %u event_pub_msec %u
event_severity %u msg {%s} msg_count %d line %u key %u tty %u error_code %u"
```

Event Type	Description
<b>event_id</b>	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
<b>event_type</b>	Type of event.
<b>event_type_string</b>	An ASCII string that represents the name of the event for this event type.
<b>event_pub_sec</b> <b>event_pub_msec</b>	The time, in seconds and milliseconds, at which the event was published to the EEM.
<b>event_severity</b>	The severity of the event.

<b>msg</b>	Text entered at the CLI prompt.
<b>msg_count</b>	Number of times the pattern matched before the event was triggered.
<b>line</b>	The text the parser was able to expand up to the point where the matched key was entered.
<b>key</b>	The enter, questionmark, or tab key.
<b>tty</b>	Corresponds to the line number the user is executing the command on.
<b>error_code</b>	The error code in CLI. 0—No error from parser up to point where a key was entered. 1—Command is ambiguous up to point where a key was entered. 4—Unknown command up to point where a key was entered.

**For EEM\_EVENT\_COUNTER**

```
"event_id %u event_type %u event_type_string {%s} %u event_pub_sec %u event_pub_msec %u"
"name {%s}"
```

Event Type	Description
<b>event_id</b>	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
<b>event_type</b>	Type of event.
<b>event_type_string</b>	An ASCII string that represents the name of the event for this event type.
<b>event_pub_sec</b> <b>event_pub_msec</b>	The time, in seconds and milliseconds, when the event was published to the EEM.
<b>name</b>	Counter name.

**For EEM\_EVENT\_GOLD**

```
"event_id %u event_type %u event_type_string {%s} %u card %u sub_card %u"
"event_severity {%s} event_pub_sec %u event_pub_msec %u overall_result %u"
"new_failure {%s} action_notify {%s} tt %u tc %u bl %u ci %u pc %u cn {%s}"
"sn {%s} tn# {%s} ta# %s ec# {%s} rc# %u lf# {%s} tf# %u cf# %u tr# {%s}"
"tr#p# {%s} tr#d# {%s}"
```

Event Type	Description
<b>action_notify</b>	Action notify information in GOLD event: true or false.
<b>bl</b>	The boot-up diagnostic level, which can be one of the following values: <ul style="list-style-type: none"> <li>0: complete diagnostic</li> <li>1: minimal diagnostics</li> <li>2: bypass diagnostic</li> </ul>
<b>card</b>	Card information for the GOLD event.
<b>ctestnum</b>	Consecutive failure, where <i>testnum</i> is the test number. For example, <b>cf3</b> is the EEM built-in environment variable for consecutive failure of test 3.
<b>ci</b>	Card index.
<b>cn</b>	Card name.

<b>ectestnum</b>	Test error code, where <i>testnum</i> is the test number. For example, <b>ec3</b> is the EEM built-in environment variable for the error code of test 3.
<b>event_id</b>	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same <i>event_id</i> .
<b>event_pub_msec</b> <b>event_pub_sec</b>	The time, in milliseconds and seconds, when the event was published to the EEM.
<b>event_severity</b>	GOLD event severity, which can be one of the following values: <ul style="list-style-type: none"> <li>• normal</li> <li>• minor</li> <li>• major.</li> </ul>
<b>event_type</b>	Type of event.
<b>event_type_string</b>	An ASCII string that represents the name of the event for this event type.
<b>lftestnum</b>	Last fail time, where <i>testnum</i> is the test number. For example, <b>lf3</b> is the EEM built-in variable for the last fail time of test 3.  The timestamp format is <i>mmm dd yyyy hh:mm:ss</i> . For example, Mar 11 1960 08:47:00.
<b>new_failure</b>	The new test failure information in a GOLD event flag: true or false.
<b>overall_result</b>	The overall diagnostic result, which can be one of the following values: <ul style="list-style-type: none"> <li>• 0: OK</li> <li>• 3: minor error</li> <li>• 4: major error</li> <li>• 14: unknown result</li> </ul>
<b>pc</b>	Port counts.
<b>rc<sub>testnum</sub></b>	Test total run count, where <i>testnum</i> is the test number. For example, <b>rc3</b> is the EEM built-in variable for the total run count of test 3.
<b>sn</b>	Card serial number.
<b>sub_card</b>	The subcard on which a GOLD failure event was detected.
<b>ta<sub>testnum</sub></b>	Test attribute, where <i>testnum</i> is the test number. For example, <b>ta3</b> is the EEM built-in variable for the test attribute of test 3.
<b>tc</b>	Test counts.
<b>tf<sub>testnum</sub></b>	Total failure count, where <i>testnum</i> is the test number. For example, <b>tf3</b> is the EEM built-in variable for the total failure count of test 3.
<b>tn<sub>testnum</sub></b>	Test name, where <i>testnum</i> is the test number. For example, <b>tn3</b> is the EEM built-in variable for the name of test 3.

<b>trtestnum</b>	<p>Test result, where <i>testnum</i> is the test number. For example, <b>tr6</b> is the EEM built-in variable for test 6 where test 6 is not a per-port test and not a per-device test.</p> <p>The test result is one of the following values:</p> <ul style="list-style-type: none"> <li>• P: diagnostic result Pass</li> <li>• F: diagnostic result Fail</li> <li>• U: diagnostic result Unknown</li> </ul>
<b>trtestnumddevnum</b>	<p>Per-device test result, where <i>testnum</i> is the test number and <i>devnum</i> is the device number. For example, <b>tr3d20</b> is the EEM built-in variable for the test result for test 3, device 20.</p> <p>The test result is one of the following values:</p> <ul style="list-style-type: none"> <li>• P: diagnostic result Pass</li> <li>• F: diagnostic result Fail</li> <li>• U: diagnostic result Unknown</li> </ul>
<b>trtestnumpportnum</b>	<p>Per-port test result, where <i>testnum</i> is the test number and <i>portnum</i> is the device number. For example, <b>tr5p20</b> is the EEM built-in variable for the test result for test 3, port 20.</p> <p>The test result is one of the following values:</p> <ul style="list-style-type: none"> <li>• P: diagnostic result Pass</li> <li>• F: diagnostic result Fail</li> <li>• U: diagnostic result Unknown</li> </ul>
<b>tt</b>	<p>The testing type, which can be one of the following:</p> <ul style="list-style-type: none"> <li>• 1: A boot-up diagnostic</li> <li>• 2: An on-demand diagnostic</li> <li>• 3: A schedule diagnostic</li> <li>• 4: A monitoring diagnostic</li> </ul>

**For EEM\_EVENT\_INTERFACE**

```
"event_id %u event_type %u event_type_string {%s} %u event_pub_sec %u event_pub_msec %u"
"event_severity {%s} name {%s} parameter {%s} value %d"
```

Event Type	Description
<b>event_id</b>	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
<b>event_type</b>	Type of event.
<b>event_type_string</b>	An ASCII string that represents the name of the event for this event type.
<b>event_pub_sec</b> <b>event_pub_msec</b>	The time, in seconds and milliseconds, when the event was published to the EEM.



<b>event_severity</b>	Interface event severity, which can be one of the following values: <ul style="list-style-type: none"><li>• normal</li><li>• minor</li><li>• major</li></ul>
<b>name</b>	Name of the interface.
<b>parameter</b>	Name of the parameter.
<b>value</b>	The incremental/decremental difference compared to the last event triggered or the absolute value of the parameter being monitored, depending on the specified value of entry_val_is_increment.

**For EEM\_EVENT\_IOSWDSYSMON**

```
"event_id %u event_type %u event_type_string {%s} %u event_pub_sec %u event_pub_msec %u"
"num_subs %u"
```

Event Type	Description
<b>event_id</b>	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
<b>event_type</b>	Type of event.
<b>event_type_string</b>	An ASCII string that represents the name of the event for this event type.
<b>event_pub_sec</b> <b>event_pub_msec</b>	The time, in seconds and milliseconds, when the event was published to the EEM.
<b>num_subs</b>	Number of subevents.

Where the subevent info string is for a CPU\_UTIL subevent,

```
"{type %s procname {%s} pid %u taskname {%s} taskid %u value %u sec %ld msec %ld}"
```

Subevent Type	Description
<b>type</b>	Type of subevent.
<b>procname</b>	POSIX process name for this subevent.
<b>pid</b>	POSIX process ID for this subevent.
<b>taskname</b>	Cisco IOS task name for this subevent.
<b>taskid</b>	Cisco IOS task ID for this subevent.
<b>value</b>	Actual average CPU utilization over the measured interval.
<b>sec,</b> <b>msec</b>	Elapsed time period for this measured interval.

Where the subevent info string is for a MEM\_UTIL subevent,

```
"{type %s procname {%s} pid %u taskname {%s} taskid %u is_percent %s value %u diff %d"
"sec %ld msec %ld}"
```

Subevent Type	Description
<b>type</b>	Type of subevent.
<b>procname</b>	POSIX process name for this subevent.
<b>pid</b>	POSIX process ID for this subevent.
<b>taskname</b>	Cisco IOS task name for this subevent.
<b>taskid</b>	Cisco IOS task ID for this subevent.
<b>is_percent</b>	TRUE or FALSE depending on whether the value is a percentage value.
<b>value</b>	Total memory use in KB or the actual average memory utilization for this measured interval.

<b>diff</b>	The percentage difference between the oldest sample in the measured interval and the latest sample; a negative value represents a decrease.
<b>sec, msec</b>	Elapsed time period for this measured interval.

**For EEM\_Event\_IPSLA**

```
"event_ID %u event_type %u event_pub_sec %u event_pub_msec %u event_severity %u"
"group_name %u operation_id %u condition %u reaction_type %u dest_ip_addr %u"
"threshold_rising %u threshold_falling%u measured_threshold_value %u"
"threshold_count1 %u threshold_count2 %u"
```

Event Type	Description
<b>event_id</b>	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
<b>event_type</b>	The type of event to monitor for the create, update, and delete flow.
<b>event_type_string</b>	An ASCII string that represents the name of the event for this event type.
<b>event_pub_sec event_pub_msec</b>	The time, in seconds and milliseconds, when the event was published to the EEM.
<b>event_severity</b>	The severity of the event.
<b>group_name</b>	The name of the IPSLA group.
<b>operation_id</b>	The IPSLA operation ID.
<b>condition</b>	The condition of IPSLA, which can be one of the following: <ul style="list-style-type: none"> <li>cleared</li> <li>occurred</li> </ul>
<b>reaction_type</b>	The IPSLA reaction type.
<b>dest_ip_address</b>	The IPSLA destination IP address.
<b>threshold rising</b>	The IPSLA configured rising threshold value.
<b>threshold falling</b>	The IPSLA configured falling threshold value.
<b>measured_threshold_ value</b>	The measured threshold value of the IPSLA operation.
<b>threshold_count1</b>	Corresponds to the argument of the threshold type1.
<b>threshold_count2</b>	Corresponds to the argument of the threshold type2.

**For EEM\_EVENT\_NF**

```
"event_ID %u event_type %u event_type_string {%s} event_pub_sec %u event_pub_msec %u
event_severity %u monitor_name %u event1-event4_field %u event1-event4_value"
```

Event Type	Description
<b>event_id</b>	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
<b>event_type</b>	The type of event to monitor for the create, update, and delete flow.

<b>event_type_string</b>	An ASCII string that represents the name of the event for this event type.
<b>event_pub_sec</b> <b>event_pub_msec</b>	The time, in seconds and milliseconds, when the event was published to the EEM.
<b>event_severity</b>	The severity of the NetFlow event.
<b>monitor_name</b>	The name of the NetFlow monitor.
<b>event1-event4_field</b>	Specifies the event and its attributes to monitor. Valid values are <b>event1</b> , <b>event2</b> , <b>event3</b> , and <b>event4</b> .
<b>event1-event4_value</b>	Specifies the event value and its attributes to monitor. Valid values are <b>event1</b> , <b>event2</b> , <b>event3</b> , and <b>event4</b> .

**For EEM\_EVENT\_NONE**

```
"event_id %u event_type %u event_type_string {%s} event_pub_sec %u event_pub_msec %u
event_severity %u arg %u"
```

<b>Event Type</b>	<b>Description</b>
<b>event_id</b>	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
<b>event_type</b>	Type of event.
<b>event_type_string</b>	An ASCII string that represents the name of the event for this event type.
<b>event_pub_sec</b> <b>event_pub_msec</b>	The time, in seconds and milliseconds, when the event was published to the EEM.
<b>event_severity</b>	The severity of the event.
<b>argc</b> <b>arg1</b> <b>arg2</b> <b>arg3</b> <b>arg4</b> <b>arg6</b> <b>arg7</b> <b>arg8</b> <b>arg9</b> <b>arg10</b> <b>arg11</b> <b>arg12</b> <b>arg13</b> <b>arg14</b> <b>arg15</b>	The parameters that are passed from the XML SOAP command to the script.

**For EEM\_EVENT\_OIR**

```
"event_id %u event_type %u event_type_string {%s} event_pub_sec %u event_pub_msec %u"
```

```
"slot %u event %s"
```

Event Type	Description
<b>event_id</b>	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event ID.
<b>event_type</b>	Type of event.
<b>event_type_string</b>	An ASCII string that represents the name of the event for this event type.
<b>event_pub_sec</b> <b>event_pub_msec</b>	The time, in seconds and milliseconds, when the event was published to the EEM.
<b>slot</b>	Slot number for the affected card.
<b>event</b>	Indicates a string, removed or online, that represents either an OIR removal event or an OIR insertion event.

#### For EEM\_EVENT\_PROCESS (Software Modularity Only)

```
"event_id %u event_type %u event_type_string {%s} event_pub_sec %u event_pub_msec %u"
"sub_system 0x%x instance %u process_name {%s} path {%s} exit_status 0x%x"
"respawn_count %u last_respawn_sec %ld last_respawn_msec %ld fail_count %u"
"dump_count %u node_name {%s}"
```

Event Type	Description
<b>event_id</b>	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
<b>event_type</b>	Type of event.
<b>event_type_string</b>	An ASCII string that represents the name of the event for this event type.
<b>event_pub_sec</b> <b>event_pub_msec</b>	The time, in seconds and milliseconds, when the event was published to the EEM.
<b>sub_system</b>	Number assigned to the EEM policy that published the application-specific event. Number is set to 798 because all other numbers are reserved for Cisco use.
<b>instance</b>	Process instance ID.
<b>process_name</b>	Process name.
<b>path</b>	Process absolute name including path.
<b>exit_status</b>	Process last exit status.
<b>respawn_count</b>	Number of times that the process was restarted.
<b>last_respawn_sec</b> <b>last_respawn_msec</b>	The calendar time when the last restart occurred.
<b>fail_count</b>	Number of restart attempts of the process that failed. This count will be reset to 0 when the process is successfully restarted.

<b>dump_count</b>	Number of core dumps taken of the process.
<b>node_name</b>	Name of the node that the process is on. The node name is a string that consists of the word “node” followed by two fields separated by a slash character using the following format:  <b>nodeslot-number/cpu-number</b>  The slot-number is the hardware slot number. The cpu-number is the hardware CPU number.

**For EEM\_EVENT\_RESOURCE**

```
"event_id %u event_type %u event_type_string {%s} %u event_pub_sec %u event_pub_msec %u"
"owner_id %lld user_id %lld" time_sent %llu dampen_time %d notify_data_flags %u"
"level {%s} direction {%s} configured_threshold %u current_value %u"
"policy_violation_flag {%s} policy_id %d"
```

Event Type	Description
<b>event_id</b>	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
<b>event_type</b>	Type of event.
<b>event_type_string</b>	An ASCII string that represents the name of the event for this event type.
<b>event_pub_sec</b> <b>event_pub_msec</b>	The time, in seconds and milliseconds, when the event was published to the EEM.
<b>owner_id</b>	The Embedded Resource Manager (ERM) owner ID.
<b>user_id</b>	The ERM user ID.
<b>time_sent</b>	The ERM event time, in nanoseconds.
<b>dampen_time</b>	The ERM dampen time, in nanoseconds.
<b>notify_data_flags</b>	The ERM notify data flag.
<b>level</b>	The ERM event level. The four event levels are normal, minor, major, and critical.
<b>direction</b>	The ERM event direction. The event direction can be one of the following: up, down, or no change.
<b>configured_threshold</b>	The configured ERM threshold.
<b>current_value</b>	The current value reported by ERM.
<b>policy_violation_flag</b>	The ERM policy violation flag; either false or true.
<b>policy_id</b>	The ERM policy ID.

**For EEM\_EVENT\_RF**

```
"event_id %u event_type %u event_type_string {%s} %u event_pub_sec %u event_pub_msec %u"
"event {%s}"
```

Event Type	Description
<b>event_id</b>	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.

<b>event_type</b>	Type of event.
<b>event_type_string</b>	An ASCII string that represents the name of the event for this event type.
<b>event_pub_sec</b> <b>event_pub_msec</b>	The time, in seconds and milliseconds, when the event was published to the EEM.
<b>event</b>	RF progression or status event notification that caused this event to be published.

**For EEM\_EVENT\_Routing**

```
"event_id %u event_type %u event_type_string {%s} %u event_pub_sec %u event_pub_msec %u"
"event_severity {%s} %u network %u mask %u protocol %u lastgateway %u distance %u"
"time_sec %u time_msec %u metric %u lastinterface %u"
```

<b>Event Type</b>	<b>Description</b>
<b>event_id</b>	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
<b>event_type</b>	Type of event.
<b>event_type_string</b>	An ASCII string that represents the name of the event for this event type.
<b>event_pub_sec</b> <b>event_pub_msec</b>	The time, in seconds and milliseconds, when the event was published to the EEM.
<b>event_severity</b>	The severity of the event.
<b>network</b>	The network prefix in IP address format
<b>mask</b>	The network mask in IP address format
<b>protocol</b>	Type of network protocol.
<b>type</b>	Type of event to add, remove or modify.
<b>lastgateway</b>	The last known gateway.
<b>distance</b>	The administrative distance.
<b>time_sec</b> <b>time_msec</b>	Time of event in seconds and milliseconds, when the event was published to the EEM.
<b>metric</b>	Path metric.
<b>lastinterface</b>	The last known interface.

**For EEM\_EVENT\_RPC**

```
"event_id %u event_type %u event_type_string {%s} event_pub_sec %u event_pub_msec %u
arg %u"
```

<b>Event Type</b>	<b>Description</b>
<b>event_id</b>	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
<b>event_type</b>	Type of event.
<b>event_type_string</b>	An ASCII string that represents the name of the event for this event type.

<b>event_pub_sec</b> <b>event_pub_msec</b>	The time, in seconds and milliseconds, when the event was published to the EEM.
<b>argc</b> <b>arg0</b> <b>arg1</b> <b>arg2</b> <b>arg3</b> <b>arg4</b> <b>arg6</b> <b>arg7</b> <b>arg8</b> <b>arg9</b> <b>arg10</b> <b>arg11</b> <b>arg12</b> <b>arg13</b> <b>arg14</b>	The parameters that are passed from the XML SOAP command to the script.

**For EEM\_EVENT\_SNMP**

```
"event_id %u event_type %u event_type_string {%s} %u event_pub_sec %u event_pub_msec %u"
"event_severity {%s} oid {%s} val {%s} delta_val {%s}"
```

<b>Event Type</b>	<b>Description</b>
<b>event_id</b>	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
<b>event_type</b>	Type of event.
<b>event_type_string</b>	An ASCII string that represents the name of the event for this event type.
<b>event_pub_sec</b> <b>event_pub_msec</b>	The time, in seconds and milliseconds, when the event was published to the EEM.
<b>event_severity</b>	SNMP event severity, which can be one of the following values: <ul style="list-style-type: none"> <li>• normal</li> <li>• minor</li> <li>• major</li> </ul>
<b>oid</b>	Object ID of data element, in SNMP dot notation.
<b>val</b>	Value of the data element.
<b>delta_val</b>	Delta value between the value of the policies.

**For EEM\_EVENT\_SNMP\_Notification**

```
"event_id %u event_type %u event_type_string {%s} %u event_pub_sec %u event_pub_msec %u"
```



```
event_severity {%s}" "oid {%s} oid_val {%s} src_ip_addr {%s} dest_ip_addr {%s} x_x_x_x_x
(varbinds) {%s} trunc_vb_buf {%s} trap_oid {%s} enterprise_oid {%s} generic_trap %u
specific_trap %u"
```

Event Type	Description
<b>event_id</b>	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
<b>event_type</b>	Type of event.
<b>event_type_string</b>	An ASCII string that represents the name of the event for this event type.
<b>event_pub_sec</b> <b>event_pub_msec</b>	The time, in seconds and milliseconds, when the event was published to the EEM.
<b>oid</b>	An user specified object ID.
<b>oid_val</b>	An user specified object ID value.
<b>src_ip_addr</b>	The source IP address of the SNMP protocol data unit (PDU).
<b>dest_ip_addr</b>	The destination IP address of the SNMP PDU.
<b>x_x_x_x_x (varbinds)</b>	The SNMP PDU varbind information.
<b>trap_oid</b>	Indicates the trap OID value.
<b>enterprise_oid</b>	Indicates the enterprise OID value.
<b>generic_trap</b>	Indicates one of a number of generic trap types. There are seven generic trap numbers zero to six.
<b>specific_trap</b>	Indicates one of a number of specific trap codes.

#### Event\_reqinfo for EEM\_EVENT\_SNMP\_Object

```
"event_id %u event_type %u event_type_string {%s} %u event_pub_sec %u event_pub_msec %u
event_severity {%s}" "oid {%s} request {%s} request_type {%s} value %u"
```

Event Type	Description
<b>event_id</b>	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
<b>event_type</b>	Type of event.
<b>event_type_string</b>	An ASCII string that represents the name of the event for this event type.
<b>event_pub_sec</b> <b>event_pub_msec</b>	The time, in seconds and milliseconds, when the event was published to the EEM.
<b>event_severity</b>	The severity of the event.
<b>oid</b>	The ID of the SNMP object in the received get or set request.
<b>request</b>	The get or set request type.
<b>request_type</b>	The type of request (exact or next).
<b>value</b>	For set requests only. The value to set the object to.

**For EEM\_EVENT\_SYSLOG\_MSG**

```
"event_id %u event_type %u event_type_string {%s} event_pub_sec %u event_pub_msec %u"
"msg {%s}"
```

Event Type	Description
<b>event_id</b>	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
<b>event_type</b>	Type of event.
<b>event_type_string</b>	An ASCII string that represents the name of the event for this event type.
<b>event_pub_sec</b> <b>event_pub_msec</b>	The time, in seconds and milliseconds, when the event was published to the EEM.
<b>msg</b>	The last syslog message that matches the pattern.

**For EEM\_EVENT\_TIMER\_ABSOLUTE**  
**EEM\_EVENT\_TIMER\_COUNTDOWN**  
**EEM\_EVENT\_TIMER\_WATCHDOG**

```
"event_id %u event_type %u event_type_string {%s} event_pub_sec %u event_pub_msec %u"
"timer_type %s timer_time_sec %ld timer_time_msec %ld"
"timer_remain_sec %ld timer_remain_msec %ld"
```

Event Type	Description
<b>event_id</b>	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
<b>event_type</b>	Type of event.
<b>event_type_string</b>	An ASCII string that represents the name of the event for this event type.
<b>event_pub_sec</b> <b>event_pub_msec</b>	The time, in seconds and milliseconds, when the event was published to the EEM.
<b>timer_type</b>	Type of the timer. Can be one of the following: <ul style="list-style-type: none"> <li>• watchdog</li> <li>• countdown</li> <li>• absolute</li> </ul>
<b>timer_time_sec</b> <b>timer_time_msec</b>	Time when the timer expired.
<b>timer_remain_sec</b> <b>timer_remain_msec</b>	The remaining time before the next expiration.

**For EEM\_EVENT\_TIMER\_CRON**

```
"event_id %u event_type %u event_type_string {%s} event_pub_sec %u event_pub_msec %u"
"timer_type {%s} timer_time_sec %ld timer_time_msec %ld"
```

<b>Event Type</b>	<b>Description</b>
<b>event_id</b>	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
<b>event_type</b>	Type of event.
<b>event_type_string</b>	An ASCII string that represents the name of the event for this event type.
<b>event_pub_sec</b> <b>event_pub_msec</b>	The time, in seconds and milliseconds, when the event was published to the EEM.
<b>timer_type</b>	Type of the timer.
<b>timer_time_sec</b> <b>timer_time_msec</b>	Time when the timer expired.

**For EEM\_EVENT\_TRACK**

```
"event_id %u event_type %u event_type_string {%s} %u event_pub_sec %u event_pub_msec %u"
"track_number {%u} track_state {%s}"
```

Event Type	Description
<b>event_id</b>	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event ID.
<b>event_type</b>	Type of event.
<b>event_type_string</b>	An ASCII string that represents the name of the event for this event type.
<b>event_pub_sec</b> <b>event_pub_msec</b>	The time, in seconds and milliseconds, when the event was published to the EEM.
<b>track_number</b>	Number of the tracked object that caused the event to be triggered.
<b>track_state</b>	State of the tracked object when the event was triggered; valid states are up or down.

**For EEM\_EVENT\_WDSYSMON (Software Modularity Only)**

```
"event_id %u event_type %u event_type_string {%s} %u event_pub_sec %u event_pub_msec %u"
"num_subs %u"
```

Event Type	Description
<b>event_id</b>	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
<b>event_type</b>	Type of event.
<b>event_type_string</b>	An ASCII string that represents the name of the event for this event type.
<b>event_pub_sec</b> <b>event_pub_msec</b>	The time, in seconds and milliseconds, when the event was published to the EEM.
<b>num_subs</b>	Subevent number.

Where the subevent info string is for a deadlock subevent:

```
"{type %s num_entries %u entries {entry 1, entry 2, ...}}"
```

Subevent Type	Description
<b>type</b>	Type of wdsysmon subevent.
<b>num_entries</b>	Number of processes and threads in the deadlock.
<b>entries</b>	Information of processes and threads in the deadlock.

Where each entry is:

```
"{node {s} procname {s} pid %u tid %u state %s b_node %s b_procname %s b_pid %u
b_tid %u}"
```

Assume that the entry describes the scenario in which Process A thread m is blocked on process B thread n:

Subevent Type	Description
<b>node</b>	Name of the node that process A thread m is on.
<b>procname</b>	Name of process A.
<b>pid</b>	Process ID of process A.
<b>tid</b>	Thread ID of process A thread m.
<b>state</b>	Thread state of process A thread m. Can be one of the following: <ul style="list-style-type: none"> <li>• STATE_CONDVAR</li> <li>• STATE_DEAD</li> <li>• STATE_INTR</li> <li>• STATE_JOIN</li> <li>• STATE_MUTEX</li> <li>• STATE_NANOSLEEP</li> <li>• STATE_READY</li> <li>• STATE_RECEIVE</li> <li>• STATE_REPLY</li> <li>• STATE_RUNNING</li> <li>• STATE_SEM</li> <li>• STATE_SEND</li> <li>• STATE_SIGSUSPEND</li> <li>• STATE_SIGWAITINFO</li> <li>• STATE_STACK</li> <li>• STATE_STOPPED</li> <li>• STATE_WAITPAGE</li> <li>• STATE_WAITTHREAD</li> </ul>
<b>b_node</b>	Name of the node that process B thread is on.
<b>b_procname</b>	Name of process B.
<b>b_pid</b>	Process ID of process B.
<b>b_tid</b>	Thread ID of process B thread n; 0 means that process A thread m is blocked on all threads of process B.

**For dispatch\_mgr Subevent**

```
"{type %s node {%s} procname {%s} pid %u value %u sec %ld msec %ld}"
```

Subevent Type	Description
<b>type</b>	Type of wdsysmon subevent.
<b>node</b>	Name of the node that the POSIX process is on.
<b>procname</b>	POSIX process name for this subevent.
<b>pid</b>	POSIX process ID for this subevent. <b>Note</b> The three fields above describe the owner process of this dispatch manager.
<b>value</b>	If the <b>sec</b> and <b>msec</b> variables are specified as 0 or are unspecified in the event registration Tcl command extension, the number of events processed by the dispatch manager is in the latest sample. If a time window is specified and is greater than zero in the event registration Tcl command extension, the total number of events processed by this dispatch manager is in the given time window.
<b>sec</b> <b>msec</b>	If the <b>sec</b> and <b>msec</b> variables are specified as 0 or are unspecified in the event registration Tcl command extension, they are both 0. If a time window is specified and is greater than zero in the event registration Tcl command extension, the <b>sec</b> and <b>msec</b> variables are the actual time difference between the time stamps of the oldest and latest samples in this time window.

**For cpu\_proc Subevent**

```
"{type %s node {%s} procname {%s} pid %u value %u sec %ld msec %ld}"
```

Subevent Type	Description
<b>type</b>	Type of wdsysmon subevent.
<b>node</b>	Name of the node that the POSIX process is on.
<b>procname</b>	POSIX process name for this subevent.
<b>pid</b>	POSIX process ID for this subevent. <b>Note</b> The three fields above describe the process whose CPU utilization is being monitored.
<b>value</b>	If the <b>sec</b> and <b>msec</b> variables are specified as 0 or are unspecified in the event registration Tcl command extension, the process CPU utilization is in the latest sample. If a time window is specified and is greater than zero in the event registration Tcl command extension, the averaged process CPU utilization is in the given time window.
<b>sec</b> <b>msec</b>	If the <b>sec</b> and <b>msec</b> variables are specified as 0 or are unspecified in the event registration Tcl command extension, they are both 0. If a time window is specified and is greater than zero in the event registration Tcl command extension, the <b>sec</b> and <b>msec</b> variables are the actual time difference between the time stamps of the oldest and latest samples in this time window.

**For cpu\_tot Subevent**

```
"(type %s node {%s} value %u sec %ld msec %ld)"
```

Subevent Type	Description
<b>type</b>	Type of wdsysmon subevent.
<b>node</b>	Name of the node on which the total CPU utilization is being monitored.
<b>value</b>	If the <b>sec</b> and <b>msec</b> variables are specified as 0 or are unspecified in the event registration Tcl command extension, the total CPU utilization is in the latest sample. If a time window is specified and is greater than zero in the event registration Tcl command extension, the averaged total CPU utilization is in the given time window.
<b>sec</b> <b>msec</b>	If the <b>sec</b> and <b>msec</b> variables are specified as 0 or are unspecified in the event registration Tcl command extension, they are both 0. If a time window is specified and is greater than zero in the event registration Tcl command extension, the <b>sec</b> and <b>msec</b> variables are the actual time difference between the time stamps of the oldest and latest samples in this time window.

**For mem\_proc Subevent**

```
"(type %s node {%s} procname {%s} pid %u is_percent %s value %u diff %d sec %ld msec %ld)"
```

Subevent Type	Description
<b>type</b>	Type of wdsysmon subevent.
<b>node</b>	Name of the node that the POSIX process is on.
<b>procname</b>	POSIX process name for this subevent.
<b>pid</b>	POSIX process ID for this subevent.  <b>Note</b> The three fields above describe the process whose memory usage is being monitored.
<b>is_percent</b>	Can be either TRUE or FALSE. TRUE means that the value is a percentage value; FALSE means that the value is an absolute value (may be an averaged value).
<b>value</b>	If the <b>sec</b> and <b>msec</b> variables are specified as 0 or are unspecified in the event registration Tcl command extension, the process used memory is in the latest sample. If a time window is specified and is greater than zero in the event registration Tcl command extension, the averaged process used memory utilization is in the given time window.

Subevent Type	Description
<b>diff</b>	If the <b>sec</b> and <b>msec</b> variables are specified as 0 or are unspecified in the event registration Tcl command extension, the <b>diff</b> is the percentage difference between the first process used memory sample ever collected and the latest process used memory sample. If a time window is specified and is greater than zero in the event registration Tcl command extension, the <b>diff</b> is the percentage difference between the oldest and latest process used memory utilization in the specified time window.
<b>sec</b> <b>msec</b>	If the <b>sec</b> and <b>msec</b> variables are specified as 0 or are unspecified in the event registration Tcl command extension, they are both 0. If a time window is specified and is greater than zero in the event registration Tcl command extension, the <b>sec</b> and <b>msec</b> variables are the actual time difference between the time stamps of the oldest and latest samples in this time window.

If the **is\_percent** argument is FALSE, and the **sec** and **msec** arguments are specified as 0 or are unspecified in the event registration Tcl command extension:

- **value** is the process used memory in the latest sample.
- **diff** is 0.
- **sec** and **msec** are both 0.

If the **is\_percent** argument is FALSE, and a time window is specified as greater than zero in the event registration Tcl command extension:

- **value** is the averaged process used memory sample value in the specified time window.
- **diff** is 0.
- **sec** and **msec** are both the actual time difference between the time stamps of the oldest and latest samples in this time window.

If the **is\_percent** argument is TRUE, and a time window is specified as greater than zero in the event registration Tcl command extension:

- **value** is 0.
- **diff** is the percentage difference between the oldest and latest process used memory samples in the specified time window.
- **sec** and **msec** are the actual time difference between the time stamps of the oldest and latest process used memory samples in this time window.

If the **is\_percent** argument is TRUE, and the **sec** and **msec** arguments are specified as 0 or are unspecified in the event registration Tcl command extension:

- **value** is 0.
- **diff** is the percentage difference between the first process used memory sample ever collected and the latest process used memory sample.
- **sec** and **msec** are the actual time difference between the time stamps of the first process used memory sample ever collected and the latest process used memory sample.



**For mem\_tot\_avail Subevent**

```
"(type %s node {%s} is_percent %s used %u avail %u diff %d sec %ld msec %ld)"
```

Subevent Type	Description
<b>type</b>	Type of wdsysmon subevent.
<b>node</b>	Name of the node for which the total available memory is being monitored.
<b>is_percent</b>	Can be either TRUE or FALSE. TRUE means that the value is a percentage value; FALSE means that the value is an absolute value (may be an averaged value).
<b>used</b>	If the <b>sec</b> and <b>msec</b> variables are specified as 0 or are unspecified in the event registration Tcl command extension, the total used memory is in the latest sample. If a time window is specified and is greater than zero in the event registration Tcl command extension, the averaged total used memory utilization is in the given time window.
<b>avail</b>	If the <b>sec</b> and <b>msec</b> variables are specified as 0 or are unspecified in the event registration Tcl command extension, the <b>avail</b> is in the latest total available memory sample. If a time window is specified and is greater than zero in the event registration Tcl command extension, the <b>avail</b> is the total available memory utilization in the specified time window.
<b>diff</b>	If the <b>sec</b> and <b>msec</b> variables are specified as 0 or are unspecified in the event registration Tcl command extension, the <b>diff</b> is the percentage difference between the first total available memory sample ever collected and the latest total available memory sample. If a time window is specified and is greater than zero in the event registration Tcl command extension, the <b>diff</b> is the percentage difference between the oldest and latest total available memory utilization in the specified time window.
<b>sec</b> <b>msec</b>	If the <b>sec</b> and <b>msec</b> variables are specified as 0 or are unspecified in the event registration Tcl command extension, they are both 0. If a time window is specified and is greater than zero in the event registration Tcl command extension, they are the actual time difference between the time stamps of the oldest and latest samples in this time window.

If the **is\_percent** argument is FALSE, and the **sec** and **msec** arguments are specified as 0 or are unspecified in the event registration Tcl command extension:

- **used** is the total used memory in the latest sample.
- **avail** is the total available memory in the latest sample.
- **diff** is 0.
- **sec** and **msec** are both 0.

If the **is\_percent** argument is FALSE, and a time window is specified as greater than zero in the event registration Tcl command extension:

- **used** is 0.
- **avail** is the averaged total available memory sample value in the specified time window.

- **diff** is 0.
- **sec** and **msec** are both the actual time difference between the time stamps of the oldest and latest total available memory samples in this time window.

If the **is\_percent** argument is TRUE, and a time window is specified as greater than zero in the event registration Tcl command extension:

- **used** is 0.
- **avail** is 0.
- **diff** is the percentage difference between the oldest and latest total available memory samples in the specified time window.
- **sec** and **msec** are both the actual time difference between the time stamps of the oldest and latest total available memory samples in this time window.

If the **is\_percent** argument is TRUE, and the **sec** and **msec** arguments are specified as 0 or are unspecified in the event registration Tcl command extension:

- **used** is 0.
- **avail** is 0.
- **diff** is the percentage difference between the first total available memory sample ever collected and the latest total available memory sample.
- **sec** and **msec** are the actual time difference between the time stamps of the first total available memory sample ever collected and the latest total available memory sample.

#### For mem\_tot\_used Subevent

```
"{type %s node {%s} is_percent %s used %u avail %u diff %d sec %ld msec %ld}"
```

Subevent Type	Description
<b>type</b>	Type of wdsysmon subevent.
<b>node</b>	Name of the node for which the total used memory is being monitored.
<b>is_percent</b>	Can be either TRUE or FALSE. TRUE means that the value is a percentage value; FALSE means that the value is an absolute value (may be an averaged value).
<b>used</b>	If the <b>sec</b> and <b>msec</b> variables are specified as 0 or are unspecified in the event registration Tcl command extension, the total used memory is in the latest sample. If a time window is specified and is greater than zero in the event registration Tcl command extension, the averaged total used memory utilization is in the given time window.
<b>avail</b>	If the <b>sec</b> and <b>msec</b> variables are specified as 0 or are unspecified in the event registration Tcl command extension, the <b>avail</b> is in the latest total used memory sample. If a time window is specified and is greater than zero in the event registration Tcl command extension, the <b>avail</b> is the total used memory utilization in the specified time window.

<b>diff</b>	If the <b>sec</b> and <b>msec</b> variables are specified as 0 or are unspecified in the event registration Tcl command extension, the <b>diff</b> is the percentage difference between the first total used memory sample ever collected and the latest total used memory sample. If a time window is specified and is greater than zero in the event registration Tcl command extension, the <b>diff</b> is the percentage difference between the oldest and latest total used memory utilization in the specified time window.
<b>sec</b> <b>msec</b>	If the <b>sec</b> and <b>msec</b> variables are specified as 0 or are unspecified in the event registration Tcl command extension, they are both 0. If a time window is specified and is greater than zero in the event registration Tcl command extension, the <b>sec</b> and <b>msec</b> variables are the actual time difference between the time stamps of the oldest and latest samples in this time window.

If the **is\_percent** argument is FALSE, and the **sec** and **msec** arguments are specified as 0 or are unspecified in the event registration Tcl command extension:

- **used** is the total used memory in the latest sample,
- **avail** is the total available memory in the latest sample,
- **diff** is 0,
- **sec** and **msec** are both 0,

If the **is\_percent** argument is FALSE, and a time window is specified as greater than zero in the event registration Tcl command extension:

- **used** is the averaged total used memory sample value in the specified time window,
- **avail** is 0,
- **diff** is 0,
- **sec** and **msec** are both the actual time difference between the time stamps of the oldest and latest total used memory samples in this time window,

If the **is\_percent** argument is TRUE, and a time window is specified as greater than zero in the event registration Tcl command extension:

- **used** is 0.
- **avail** is 0.
- **diff** is the percentage difference between the oldest and latest total used memory samples in the specified time window.
- **sec** and **msec** are both the actual time difference between the time stamps of the oldest and latest total used memory samples in this time window.

If the **is\_percent** argument is TRUE, and the **sec** and **msec** arguments are specified as 0 or are unspecified in the event registration Tcl command extension:

- **used** is 0.
- **avail** is 0.
- **diff** is the percentage difference between the first total used memory sample ever collected and the latest total used memory sample.
- **sec** and **msec** are the actual time difference between the time stamps of the first total used memory sample ever collected and the latest total used memory sample.

■ event\_reqinfo

**Set\_cerrno**

Yes

## EEM Event Tcl Command Extension

- [event\\_completion](#)
- [event\\_completion\\_with\\_wait](#)
- [event\\_publish](#), page 153
- [event\\_wait](#), page 156

# event\_completion

Sends a notification to the EEM server that the policy is done servicing the event that triggered it. The event only takes a single argument which is the **return\_code** of this event instance.

## Syntax

```
event_completion status ?
```

## Arguments

status	(Mandatory) Exit status (return_code) of this event instance. A value of zero indicates no error and any other integer value indicates an error.
--------	--

## Result String

None

## Set\_cerrno

No

# event\_completion\_with\_wait

The **event\_completion\_with\_wait** command combines the two commands **event\_completion** and **event\_wait** into a single command for ease of use.

The **event\_completion** command sends a notification to the EEM server that the policy is done servicing the event that triggered it. The event only takes a single argument which is the **return\_code** of this event instance.

The **event\_wait** places the Tcl policy into a sleep state. When the Tcl policy receives a new signal announcing a new event, the policy is placed into a wake state and again returns to a sleep state. This loop continues. If **event\_wait** policy is invoked before **event\_completed** policy, an error results and the policy exits.

## Syntax

```
event_completion_with_wait status ? [refresh_vars]
```

## Arguments

status	(Mandatory) exit_status (return_code) of this event instance. A value of zero indicates no error. Any other integer value indicates an error.
refresh_vars	(Optional) Indicates whether built-in and environment variables should be updated (refreshed) from the EEM Policy Director during this event instance.

## Result String

None

## Set\_cerrno

Yes

## Sample Usage

Here is a similar example as above using this single command:

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

set i 1

while {1 == 1} { # Start high performance policy loop

    array set arr_einfo [event_reqinfo]
    if {$_cerrno != 0} {
        set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
            $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
        error $result
    }

    action_syslog msg "event $i serviced" priority info

    if {$i == 5} {
        action_syslog msg "Exiting after servicing 5 events" priority info
        exit 0
    }
}
```

```
incr i

array set _event_state_arr [event_completion_with_wait status 0 refresh_vars 1]

if {$_event_state_arr(event_state) != 0} {
    action_syslog msg "Exiting: failed event_state " \
                    " $_event_state_arr(event_state)" priority info
    exit 0
}
}
```

**Note**

---

The running configuration output is the same as the [event\\_publish](#) Tcl command.

---



# event\_publish

Publishes an application-specific event.

## Syntax

```
event_publish sub_system ? type ? [arg1 ?] [arg2 ?] [arg3 ?] [arg4 ?]
```

## Arguments

sub_system	(Mandatory) Number assigned to the EEM policy that published the application-specific event. Number is set to 798 because all other numbers are reserved for Cisco use.
type	(Mandatory) Event subtype within the specified component. The sub_system and type arguments uniquely identify an application event. Must be an integer between 1 and 4294967295, inclusive.
[arg1 ?]-[arg4 ?]	(Optional) Four pieces of application event publisher string data.

## Result String

None

## Set\_cerrno

Yes

```
(_cerr_sub_err = 2)   FH_ESYSERR   (generic/unknown error from OS/system)
```

This error means that the operating system reported an error. The POSIX errno value that is reported with the error should be used to determine the cause of the operating system error.

## Sample Usage

This example demonstrates how to use the **event\_publish** Tcl command extension to execute a script *n* times repeatedly to perform some function (for example, to measure the amount of CPU time taken by a given group of Tcl statements). This example uses two Tcl scripts.

Script1 publishes a type 9999 EEM event to cause Script2 to run for the first time. Script1 is registered as a none event and is run using the Cisco IOS CLI **event manager run** command. Script2 is registered as an EEM application event of type 9999, and this script checks to see if the application publish arg1 data (the iteration number) exceeds the EEM environment variable test\_iterations value. If the test\_iterations value is exceeded, the script writes a message and exits; otherwise the script executes the remaining statements and reschedules another run. To measure the CPU utilization for Script2, use a value of test\_iterations that is a multiple of 10 to calculate the amount of average CPU time used by Script2.

To run the Tcl scripts, enter the following Cisco IOS commands:

```
configure terminal
  event manager environment test_iterations 100
  event manager policy script1.tcl
  event manager policy script2.tcl
end
event manager run script1.tcl
```

The Tcl script Script2 will be executed 100 times. If you execute the script without the extra processing and derive the average CPU utilization, and then add the extra processing and repeat the test, you can subtract the former CPU utilization from the later CPU utilization to determine the average for the extra processing.

### Script1 (script1.tcl)

```
::cisco::eem::event_register_none

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

# Query the event info.
array set arr_einfo [event_reqinfo]
if {$_cerrno != 0} {
    set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
        $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}

action_syslog priority info msg "EEM application_publish test start"
if {$_cerrno != 0} {
    set result [format \
        "component=%s; subsys err=%s; posix err=%s;\n%s" \
        $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}

# Cause the first iteration to run.
event_publish sub_system 798 type 9999 arg1 0
if {$_cerrno != 0} {
    set result [format \
        "component=%s; subsys err=%s; posix err=%s;\n%s" \
        $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}
```

### Script2 (script2.tcl)

```
::cisco::eem::event_register_appl sub_system 798 type 9999

# Check if all the required environment variables exist.
# If any required environment variable does not exist, print out an error msg and quit.
if {![info exists test_iterations]} {
    set result \
        "Policy cannot be run: variable test_iterations has not been set"
    error $result $errorInfo
}

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

# Query the event info.
array set arr_einfo [event_reqinfo]
if {$_cerrno != 0} {
    set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
        $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}

# Data1 contains the arg1 value used to publish this event.
set iter $arr_einfo(data1)

# Use the arg1 info from the previous run to determine when to end.
if {$iter >= $test_iterations} {
```

```
# Log a message.
action_syslog priority info msg "EEM application_publish test end"
if {$_cerrno != 0} {
    set result [format \
        "component=%s; subsys err=%s; posix err=%s;\n%s" \
        $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}
exit 0
}
set iter [expr $iter + 1]

# Log a message.
set msg [format "EEM application_publish test iteration %s" $iter]
action_syslog priority info msg $msg
if {$_cerrno != 0} {
    set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
        $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}

# Do whatever processing that you want to measure here.

# Cause the next iteration to run. Note that the iteration is passed to the
# next operation as arg1.
event_publish sub_system 798 type 9999 arg1 $iter
if {$_cerrno != 0} {
    set result [format \
        "component=%s; subsys err=%s; posix err=%s;\n%s" \
        $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}
```

# event\_wait

Places the Tcl policy into a sleep state. When the Tcl policy receives a new signal announcing a new event, the policy is placed into a wake state and again returns to a sleep state. This loop continues. If **event\_wait** policy is invoked before **event\_completed** policy, an error results and the policy exits.

## Syntax

```
event_wait [refresh_vars]
```

## Arguments

refresh_vars	(Optional) Indicates whether built-in and environment variables should be updated (refreshed) from the EEM Policy Director during this event instance.
--------------	--

## Result String

None

## Set\_cerrno

No

## Sample Usage

The **event\_wait** event detector returns an array type value with a single element named **event\_state**. Event\_state is a value sent back from the EEM Server indicating whether or not an error has occurred in processing the event. An example of an error here would be if the user configured **event\_wait** before configuring **event\_completion** when handling the event instance.

The following sample output shows the use of both **event\_completion** and **event\_wait** Tcl commands:

```
::cisco::eem::event_register_syslog tag e1 occurs 1 pattern CLEAR maxrun 0

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

set i 1

while {1 == 1} { # Start high performance policy loop

    array set arr_einfo [event_reqinfo]
    if {$_cerrno != 0} {
        set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
            $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
        error $result
    }

    action_syslog msg "event $i serviced" priority info

    if {$i == 5} {
        action_syslog msg "Exiting after servicing 5 events" priority info
        exit 0
    }

    incr i

    event_completion status 0

    array set _event_state_arr [event_wait refresh_vars 0]
```

```

    if (${_event_state_arr(event_state) != 0} {
        action_syslog msg "Exiting: failed event_state " \
            " $_event_state_arr(event_state)" priority info
        exit 0
    }
}

```

Here is an example of the running configuration:

```

Router#
01:00:44: %SYS-5-CONFIG_I: Configured from console by consoleclear counters
Clear "show interface" counters on all interfaces [confirm]
Router#
01:00:49: %CLEAR-5-COUNTERS: Clear counter on all interfaces by console
01:00:49: %HA_EM-6-LOG: high_perf_example.tcl: event 1 serviced
Router#
Router#clear counters
Clear "show interface" counters on all interfaces [confirm]
Router#
Router#
01:00:53: %CLEAR-5-COUNTERS: Clear counter on all interfaces by console
01:00:53: %HA_EM-6-LOG: high_perf_example.tcl: event 2 serviced
Router#clear counters
Clear "show interface" counters on all interfaces [confirm]
Router#
Router#
01:00:56: %CLEAR-5-COUNTERS: Clear counter on all interfaces by console
01:00:56: %HA_EM-6-LOG: high_perf_example.tcl: event 3 serviced
Router#
Router#
Router#clear counters
Clear "show interface" counters on all interfaces [confirm]
Router#
01:00:59: %CLEAR-5-COUNTERS: Clear counter on all interfaces by console
Router#
01:00:59: %HA_EM-6-LOG: high_perf_example.tcl: event 4 serviced
01:00:59: %HA_EM-6-LOG: high_perf_example.tcl: Exiting after servicing 5 events
Router#
Router#
Router#copy tftp disk1:
Address or name of remote host [dirt]?
Source filename [user/eem_scripts/high_perf_example.tcl]?
Destination filename [high_perf_example.tcl]?
%Warning:There is a file already existing with this name
Do you want to over write? [confirm]
Accessing tftp://dirt/user/eem_scripts/high_perf_example.tcl...
Loading user/eem_scripts/high_perf_example.tcl from 192.0.2.19 (via FastEthernet0/0): !
[OK - 909 bytes]

909 bytes copied in 0.360 secs (2525 bytes/sec)
Router#
Router#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#no event manager policy high_perf_example.tcl
Router(config)#event manager po high_perf_example.tcl
Router(config)#end
Router#
Router#
Router#
Router#
01:02:19: %SYS-5-CONFIG_I: Configured from console by consoleclear counters
Clear "show interface" counters on all interfaces [confirm]

```

```

Router#
01:02:23: %CLEAR-5-COUNTERS: Clear counter on all interfaces by console
Router#
Router#
01:02:23: %HA_EM-6-LOG: high_perf_example.tcl: event 1 serviced
Router#
Router#clear counters
Clear "show interface" counters on all interfaces [confirm]
Router#
Router#
01:02:26: %CLEAR-5-COUNTERS: Clear counter on all interfaces by console
01:02:26: %HA_EM-6-LOG: high_perf_example.tcl: event 2 serviced
Router#
Router#clear counters
Clear "show interface" counters on all interfaces [confirm]
Router#
Router#
01:02:29: %CLEAR-5-COUNTERS: Clear counter on all interfaces by console
01:02:29: %HA_EM-6-LOG: high_perf_example.tcl: event 3 serviced
Router#
Router#clear counters
Clear "show interface" counters on all interfaces [confirm]
Router#
Router#
01:02:33: %CLEAR-5-COUNTERS: Clear counter on all interfaces by console
Router#
01:02:33: %HA_EM-6-LOG: high_perf_example.tcl: event 4 serviced
Router#
Router#clear counters
Clear "show interface" counters on all interfaces [confirm]
Router#
Router#
Router#
01:02:36: %CLEAR-5-COUNTERS: Clear counter on all interfaces by console
01:02:36: %HA_EM-6-LOG: high_perf_example.tcl: event 5 serviced
01:02:36: %HA_EM-6-LOG: high_perf_example.tcl: Exiting after servicing 5 events
Router#

```

Also while an event has been serviced and is waiting for the next event to come in **show event manager policy active** command will display the following output:

```

Router#show event manager policy active
Key: p - Priority          :L - Low, H - High, N - Normal, Z - Last
      s - Scheduling node :A - Active, S - Standby

default class - 1 script event
no.  job id   p s status  time of event          event type      name
 1    11      N A wait   Mon Oct20 14:15:24 2008  syslog
high_perf_example.tcl

```

In the above example the status is wait. This indicates that the policy is waiting for the next event to come in.

## EEM Multiple Event Support Tcl Command Extensions

- [trigger](#), page 160
- [correlate](#), page 161
- [attribute](#), page 162

# trigger

Specifies the multiple event configuration ability of Embedded Event Manager (EEM) events. A multiple event is one that can involve one or more event occurrences, one or more tracked object states, and a time period for the event to occur. The events are raised based on the specified parameters.

## Syntax

```
trigger [occurs ?] [period ?] [period-start ?] [delay ?]
```

## Arguments

<b>occurs</b>	(Optional) Specifies the number of times the total correlation occurs before an EEM event is raised. When a number is not specified, an EEM event is raised on the first occurrence. The range is from 1 to 4294967295.
<b>period</b>	(Optional) Time interval in seconds and optional milliseconds, during which the one or more occurrences must take place. This is specified in the format sssssssss[.mmm], where sssssssss must be an integer number representing seconds between 0 and 4294967295, inclusive and mmm represents milliseconds and must be an integer number between 0 to 999.
<b>period-start</b>	(Optional) Specifies the start of an event correlation window. If not specified, event monitoring is enabled after the first CRON period occurs.
<b>delay</b>	(Optional) Specifies the number of seconds and optional milliseconds after which an event will be raised if all the conditions are true (specified in the format sssssssss[.mmm], where sssssssss must be an integer number representing seconds between 0 and 4294967295, inclusive and mmm represents milliseconds and must be an integer number between 0 to 999).

## Result String

None

## Set\_cerrno

No



# correlate

Builds a single complex event and allows boolean logic to relate events and tracked objects.

## Syntax

```
correlate event ? track ? [andnot | and | or] event ? track ?
```

## Arguments

<b>event</b>	Specifies the event that can be used with the <b>trigger</b> command to support multiple event statements within an script.  If the event associated with the <i>event-tag</i> argument occurs for the number of times specified by the <b>trigger</b> command, the result is true. If not, the result is false.
<b>track</b>	Specifies the event object number for tracking. The range is from 1 to 500.  If the tracked object is set, the result of the evaluation is true. If the tracked object is not set or is undefined, the result of the evaluation is false. This result is regardless of the state of the object.
<b>andnot</b>	(Optional) Specifies that if event 1 occurs the action is executed, and if event 2 and event 3 occur together the action is not executed.
<b>and</b>	(Optional) Specifies that if event 1 occurs the action is executed, and if event 2 and event 3 occur together the action is executed.
<b>or</b>	(Optional) Specifies that if event 1 occurs the action is executed, or else if event 2 and event 3 occur together the action is executed.

## Result String

None

## Set \_cerrno

No

# attribute

Specifies a complex event.

## Syntax

```
attribute tag ? [occurs ?]
```

## Arguments

---

<b>tag</b>	Specifies a tag using the <i>event-tag</i> argument that can be used with the <b>attribute</b> command to associate an event.
<b>occurs</b>	(Optional) Specifies the number of occurrences before an EEM event is triggered. If not specified, an EEM event is triggered on the first occurrence. The range is from 1 to 4294967295.

---

## Result String

None

## Set\_cerrno

No

## EEM Action Tcl Command Extensions

- [action\\_policy](#), page 164
- [action\\_process](#), page 165
- [action\\_program](#), page 166
- [action\\_reload](#), page 167
- [action\\_script](#), page 168
- [action\\_snmp\\_trap](#), page 169
- [action\\_snmp\\_object\\_value](#), page 170
- [action\\_switch](#), page 171
- [action\\_syslog](#), page 172
- [action\\_track\\_read](#), page 173
- [action\\_track\\_set](#), page 174

# action\_policy

Allows a Tcl script to run an Embedded Event Manager (EEM) policy that has been registered with the None event detector. The action of running an EEM policy can also be performed using the **event manager run** command.

## Syntax

```
action_policy ?
```

## Arguments

? (represents a string)	(Mandatory) The name of the EEM policy to be scheduled for execution. The policy must have been previously registered with the None event detector.
-------------------------	---

None

## Result String

None

## Set\_cerrno

Yes

```
(_cerr_sub_err = 2) FH_ESYSERR (generic/unknown error from OS/system)
```

This error means that the operating system reported an error. The POSIX errno value that is reported with the error should be used to determine the cause of the operating system error.

```
(_cerr_sub_err = 12) FH_ENOSUCHEID (unknown event ID)
```

This error means that the policy is unknown because it is not registered.

```
(_cerr_sub_err = 14) FH_ENOSUCHACTION (unknown action type)
```

This error means that the action command requested was unknown.

# action\_process

Starts, restarts, or kills a Software Modularity process. This Tcl command extension is supported only in Software Modularity images.

## Syntax

```
action_process start|restart|kill [job_id ?]
[process_name ?] [instance ?]
```

## Arguments

start	(Mandatory) Specifies that a process is to be started.
restart	(Mandatory) Specifies that a process is to be restarted.
kill	(Mandatory) Specifies that a process is to be stopped (killed).
job_id	(Optional) System manager assigned job ID for the process. If you specify this argument, it must be an integer between 1 and 4294967295, inclusive.
process_name	(Optional) Process name. Either job_id must be specified or process_name and instance must be specified.
instance	(Optional) Process instance ID. If you specify this argument, it must be an integer between 1 and 4294967295, inclusive.

## Result String

None

## Set \_cerrno

Yes

```
(_cerr_sub_err = 14) FH_ENOSUCHACTION (unknown action type)
```

This error means that the action command requested was unknown.

```
(_cerr_sub_num = 425, _cerr_sub_err = 1) SYSMGR_ERROR_INVALID_ARGS (Invalid arguments passed)
```

This error means that the arguments passed in were invalid.

```
(_cerr_sub_num = 425, _cerr_sub_err = 2) SYSMGR_ERROR_NO_MEMORY (Could not allocate required memory)
```

This error means that an internal SYSMGR request for memory failed.

```
(_cerr_sub_num = 425, _cerr_sub_err = 5) SYSMGR_ERROR_NO_MATCH (This process is not known to sysmgr)
```

This error means that the process name was not known.

```
(_cerr_sub_num = 425, _cerr_sub_err = 14) SYSMGR_ERROR_TOO_BIG (outside the valid limit)
```

This error means that an object size exceeded its maximum.

```
(_cerr_sub_num = 425, _cerr_sub_err = 15) SYSMGR_ERROR_INVALID_OP (Invalid operation for this process)
```

This error means that the operation was invalid for the process.

## action\_program

Allows a Tcl script to run a POSIX process (program), optionally with a given argument string, environment string, Standard Input (stdin) pathname, Standard Output (stdout) pathname, or Standard Error (stderr) pathname. This Tcl command extension is supported only in Software Modularity images.

### Syntax

```
action_program path ? [argv ?] [envp ?] [stdin ?] [stdout ?] [stderr ?]
```

### Arguments

path	(Mandatory) The pathname of a program to run.
argv	(Optional) The argument string of the program.
envp	(Optional) The environment string of the program.
stdin	(Optional) The pathname for stdin.
stdout	(Optional) The pathname for stdout.
stderr	(Optional) The pathname for stderr.

### Result String

None

### Set\_cerrno

Yes

```
(_cerr_sub_err = 2)    FH_ESYSERR    (generic/unknown error from OS/system)
```

This error means that the operating system reported an error. The POSIX errno value that is reported with the error should be used to determine the cause of the operating system error.

```
(_cerr_sub_err = 14)   FH_ENOSUCHACTION    (unknown action type)
```

This error means that the action command requested was unknown.

```
(_cerr_sub_err = 34)   FH_EMAXLEN    (maximum length exceeded)
```

This error means that the object length or number exceeded the maximum.

# action\_reload

Reloads the router.

## Syntax

```
action_reload
```

## Arguments

None

## Result String

None

## Set\_cerrno

Yes

```
(_cerr_sub_err = 2)    FH_ESYSERR    (generic/unknown error from OS/system)
```

This error means that the operating system reported an error. The POSIX errno value that is reported with the error should be used to determine the cause of the operating system error.

```
(_cerr_sub_err = 14)  FH_ENOSUCHACTION  (unknown action type)
```

This error means that the action command requested was unknown.

# action\_script

Allows a Tcl script to enable or disable the execution of all Tcl scripts (enables or disables the script scheduler).

## Syntax

```
action_script [status enable|disable]
```

## Arguments

status	(Optional) Flag to indicate script execution status. If this argument is set to enable, script execution is enabled; if this argument is set to disable, script execution is disabled.
--------	--

## Result String

None

## Set\_cerrno

Yes

```
(_cerr_sub_err = 2)    FH_ESYSERR    (generic/unknown error from OS/system)
```

This error means that the operating system reported an error. The POSIX errno value that is reported with the error should be used to determine the cause of the operating system error.

```
(_cerr_sub_err = 14)   FH_ENOSUCHACTION    (unknown action type)
```

This error means that the action command requested was unknown.

```
(_cerr_sub_err = 52)   FH_ECONFIG    (configuration error)
```

This error means that a configuration error has occurred.



## action\_snmp\_trap

Sends a Simple Network Management Protocol (SNMP) trap using the Embedded Event Manager Notification MIB.

### Syntax

```
action_snmp_trap [intdata1 ?] [intdata2 ?] [strdata ?]
```

### Arguments

intdata1	(Optional) Arbitrary integer sent in trap.
intdata2	(Optional) Arbitrary integer sent in trap.
strdata	(Optional) Arbitrary string data sent in trap.

### Result String

None

### Set\_cerrno

Yes

```
(_cerr_sub_err = 2)    FH_ESYSERR    (generic/unknown error from OS/system)
```

This error means that the operating system reported an error. The POSIX errno value that is reported with the error should be used to determine the cause of the operating system error.

```
(_cerr_sub_err = 14)  FH_ENOSUCHACTION (unknown action type)
```

This error means that the action command requested was unknown.

## action\_snmp\_object\_value

Sets a Simple Network Management Protocol (SNMP) object ID and value to be returned for the SNMP get request.

### Syntax

```
action_snmp_object_value {int|uint|counter|gauge|ipv4|octet|counter64|string} ?
[next_oid ?]
```

### Arguments

int	A 32-bit number used to specify a numbered type within the context of a managed object.
uint	A 32-bit number used to represent decimal value.
counter	A 32-bit number with a minimum value of 0.
gauge	A 32-bit number with a minimum value of 0.
ipv4	IP version 4 address.
octet	An octet string in hex notation used to represent physical addresses.
counter 64	A 64-bit number with a minimum value of 0.
string	An octet string in text notation used to represent text strings.
next_oid	The OID of the next object in the table; NULL if it is the last object in the table.

### Result String

None

### Set\_cerrno

Yes

# action\_switch

Switches processing to a secondary processor in a fully redundant environment. Before using the **action\_switch** Tcl command extension, you must install a backup processor in the device. If the hardware is not fully redundant, the switchover action will not be performed.

**Syntax**

```
action_switch
```

**Arguments**

None

**Result String**

None

**Set\_cerrno**

Yes

```
(_cerr_sub_err = 2) FH_ESYSERR (generic/unknown error from OS/system)
```

This error means that the operating system reported an error. The POSIX errno value that is reported with the error should be used to determine the cause of the operating system error.

```
(_cerr_sub_err = 14) FH_ENOSUCHACTION (unknown action type)
```

This error means that the action command requested was unknown.

# action\_syslog

Generates a periodic syslog message using the specified facility when an EEM script is triggered.

## Syntax

```
action_syslog [priority emerg|alert|crit|err|warning|notice|info|debug]
[msg ?] [facility ?]
```

## Arguments

priority	(Optional) The action_syslog message facility level. If this argument is not specified, the default priority is LOG_INFO.
msg	(Optional) The message to be logged.
facility	(Optional) Syslog facility.

## Result String

None

## Set\_cerrno

Yes

# action\_track\_read

Reads the state of a tracked object when an Embedded Event Manager (EEM) script is triggered.

## Syntax

```
action_track_read ?
```

## Arguments

? (represents a number)	(Mandatory) Tracked object number in the range from 1 to 500, inclusive.
-------------------------	--

## Result String

```
number {%u}  
state {%s}
```

## Set\_cerrno

Yes

FH\_ENOTRACK

This error means that the tracked object number was not found.

## action\_track\_set

Sets the state of a tracked object when an Embedded Event Manager (EEM) script is triggered.

### Syntax

```
action_track_set ? state up|down
```

### Arguments

? (represents a number)	(Mandatory) Tracked object number in the range from 1 to 500, inclusive.
state	(Mandatory) Specifies that the state of the tracked object will be set. If up is specified, the state of the tracked object will be set to up. If down is specified, the state of the tracked object will be set to down.

### Result String

None

### Set\_cerrno

Yes

FH\_ENOTRACK

This error means that the tracked object number was not found.

## EEM Utility Tcl Command Extensions

- [appl\\_read](#), page 176
- [appl\\_reqinfo](#), page 177
- [appl\\_setinfo](#), page 178
- [counter\\_modify](#), page 179
- [description](#), page 180
- [fts\\_get\\_stamp](#), page 181
- [register\\_counter](#), page 182
- [register\\_timer](#), page 184
- [timer\\_arm](#), page 186
- [timer\\_cancel](#), page 188
- [unregister\\_counter](#), page 189

# appl\_read

Reads Embedded Event Manager (EEM) application volatile data. This Tcl command extension provides support for reading EEM application volatile data. EEM application volatile data can be published by a Cisco IOS software process that uses the EEM application publish API. EEM application volatile data cannot be published by an EEM policy.


**Note**


---

Currently there are no Cisco IOS software processes that publish application volatile data.

---

**Syntax**

```
appl_read name ? length ?
```

**Arguments**

name	(Mandatory) Name of the application published string data.
length	(Mandatory) Length of the string data to read. Must be an integer number between 1 and 4294967295, inclusive.

**Result String**

```
data %s
```

Where data is the application published string data to be read.

**Set\_cerrno**

Yes

```
(_cerr_sub_err = 2)    FH_ESYSERR  (generic/unknown error from OS/system)
```

This error means that the operating system reported an error. The POSIX errno value that is reported with the error should be used to determine the cause of the operating system error.

```
(_cerr_sub_err = 7)    FH_ENOSUCHKEY  (could not find key)
```

This error means that the application event detector info key or other ID was not found.

```
(_cerr_sub_err = 9)    FH_EMEMORY  (insufficient memory for request)
```

This error means that an internal EEM request for memory failed.



# appl\_reqinfo

Retrieves previously saved information from the Embedded Event Manager (EEM). This Tcl command extension provides support for retrieving information from EEM that has been previously saved with a unique key, which must be specified in order to retrieve the information. Note that retrieving the information deletes it from EEM. It must be resaved if it is to be retrieved again.

## Syntax

```
appl_reqinfo key ?
```

## Arguments

key	(Mandatory) The string key of the data.
-----	---

## Result String

```
data %s
```

Where data is the application string data to be retrieved.

## Set\_cerrno

Yes

```
(_cerr_sub_err = 2)    FH_ESYSERR  (generic/unknown error from OS/system)
```

This error means that the operating system reported an error. The POSIX errno value that is reported with the error should be used to determine the cause of the operating system error.

```
(_cerr_sub_err = 7)    FH_ENOSUCHKEY (could not find key)
```

This error means that the application event detector info key or other ID was not found.

# appl\_setinfo

Saves information in the Embedded Event Manager (EEM). This Tcl command extension provides support for saving information in the Embedded Event Manager that can be retrieved later by the same policy or by another policy. A unique key must be specified. This key allows the information to be retrieved later.

## Syntax

```
appl_setinfo key ? data ?
```

## Arguments

key	(Mandatory) The string key of the data.
data	(Mandatory) The application string data to save.

## Result String

None

## Set\_cerrno

Yes

```
(_cerr_sub_err = 2)    FH_ESYSERR  (generic/unknown error from OS/system)
```

This error means that the operating system reported an error. The POSIX errno value that is reported with the error should be used to determine the cause of the operating system error.

```
(_cerr_sub_err = 8)    FH_EDUPLICATEKEY  (duplicate appl info key)
```

This error means that the application event detector info key or other ID was a duplicate.

```
(_cerr_sub_err = 9)    FH_EMEMORY  (insufficient memory for request)
```

This error means that an internal EEM request for memory failed.

```
(_cerr_sub_err = 34)   FH_EMAXLEN  (maximum length exceeded)
```

This error means that the object length or number exceeded the maximum.

```
(_cerr_sub_err = 43)   FH_EBADLENGTH  (bad API length)
```

This error means that the API message length was invalid.

# counter\_modify

Modifies a counter value.

## Syntax

```
counter_modify event_id ? val ? op nop|set|inc|dec
```

## Arguments

event_id	(Mandatory) The counter event ID returned by the <b>register_counter</b> Tcl command extension. Must be an integer between 0 and 4294967295, inclusive.
val	(Mandatory) <b>Note</b> Mandatory except when the op nop argument value combination is specified. <ul style="list-style-type: none"><li>• If op is set, this argument represents the counter value that is to be set.</li><li>• If op is inc, this argument is the value by which to increment the counter.</li><li>• If op is dec, this argument is the value by which to decrement the counter.</li></ul>
op	(Mandatory) <ul style="list-style-type: none"><li>• nop—Retrieves the current counter value.</li><li>• set—Sets the counter value to the given value.</li><li>• inc—Increments the counter value by the given value.</li><li>• dec—Decrements the counter value by the given value.</li></ul>

## Result String

```
val_remain %d
```

Where val\_remain is the current value of the counter.

## Set\_cerrno

Yes

```
(_cerr_sub_err = 2)    FH_ESYSERR    (generic/unknown error from OS/system)
```

This error means that the operating system reported an error. The POSIX errno value that is reported with the error should be used to determine the cause of the operating system error.

```
(_cerr_sub_err = 11)   FH_ENOSUCHESID (unknown event specification ID)
```

This error means that the event specification ID could not be matched when the event was being registered or that an event detector internal event structure is corrupt.

```
(_cerr_sub_err = 22)   FH_ENULLPTR    (event detector internal error - ptr is null)
```

This error means that an internal EEM event detector pointer was null when it should have contained a value.

```
(_cerr_sub_err = 30)   FH_ECTBADOPER (bad counter threshold operator)
```

This error means that the counter event detector set or modify operator was invalid.

# description

Provides a brief description of the registered policy.

## Syntax

```
description ?
```

## Arguments

line	(Optional) Brief description of the policy consisting of 1 to 240 characters.
------	---

## Result String

None

## Set\_cerrno

Yes

## Sample Usage

The description statement is entered by the author of the policy. It can appear before or after any event registration statement in Tcl. The policy can have only one description.



### Note

Registration of a policy with more than one description statement will fail.

The following example shows how a brief description is provided for the **event\_register\_syslog** policy:

```
::cisco::eem::description "This Tcl command looks for the word count in syslog messages."
::cisco::eem::event_register_syslog tag 1 ...
::cisco::eem::event_register_snmp_object tag 2 ...
::cisco::eem::trigger
    ::cisco::eem::correlate event 1 and event 2
    ::cisco::eem::attribute tag 1 occurs 1
    ::cisco::eem::attribute tag 2 occurs 1
```

## fts\_get\_stamp

Returns the time period elapsed since the last software boot. Use this Tcl command extension to return the number of nanoseconds since boot in an array “nsec nnnn” where nnnn is the number of nanoseconds.

**Syntax**

```
fts_get_stamp
```

**Arguments**

None

**Result String**

```
nsec %d
```

Where nsec is the number of nanoseconds since boot.

**Set\_cerrno**

No

# register\_counter

Registers a counter and returns a counter event ID. This Tcl command extension is used by a counter publisher to perform this registration before using the event ID to manipulate the counter.

## Syntax

```
register_counter name ?
```

## Arguments

name	(Mandatory) The name of the counter to be manipulated.
------	--

## Result String

```
event_id %d
event_spec_id %d
```

Where `event_id` is the counter event ID for the specified counter; it can be used to manipulate the counter by the **unregister\_counter** or **counter\_modify** Tcl command extensions. The `event_spec_id` argument is the event specification ID for the specified counter.

## Set\_cerrno

Yes

```
(_cerr_sub_err = 2)    FH_ESYSERR    (generic/unknown error from OS/system)
```

This error means that the operating system reported an error. The POSIX `errno` value that is reported with the error should be used to determine the cause of the operating system error.

```
(_cerr_sub_err = 4)    FH_EINITONCE  (Init() is not yet done, or done twice.)
```

This error means that the request to register the specific event was made before the EEM event detector had completed its initialization.

```
(_cerr_sub_err = 6)    FH_EBADEVENTTYPE (unknown EEM event type)
```

This error means that the event type specified in the internal event specification was invalid.

```
(_cerr_sub_err = 9)    FH_EMEMORY    (insufficient memory for request)
```

This error means that an internal EEM request for memory failed.

```
(_cerr_sub_err = 10)   FH_ECORRUPT  (internal EEM API context is corrupt)
```

This error means that the internal EEM API context structure is corrupt.

```
(_cerr_sub_err = 11)   FH_ENOSUCHESID (unknown event specification ID)
```

This error means that the event specification ID could not be matched when the event was being registered or that an event detector internal event structure is corrupt.

```
(_cerr_sub_err = 12)   FH_ENOSUCHEID (unknown event ID)
```

This error means that the event ID could not be matched when the event was being registered or that an event detector internal event structure is corrupt.

```
(_cerr_sub_err = 16)   FH_EBADFMPPTR  (bad ptr to fh_p data structure)
```

This error means that the context pointer that is used with each EEM API call is incorrect.

```
(_cerr_sub_err = 17)    FH_EBADADDRESS  (bad API control block address)
```

This error means that a control block address that was passed in the EEM API was incorrect.

```
(_cerr_sub_err = 22)    FH_ENULLPTR    (event detector internal error - ptr is null)
```

This error means that an internal EEM event detector pointer was null when it should have contained a value.

```
(_cerr_sub_err = 25)    FH_ESUBSEXCEED (number of subscribers exceeded)
```

This error means that the number of timer or counter subscribers exceeded the maximum.

```
(_cerr_sub_err = 26)    FH_ESUBSIDXINV (invalid subscriber index)
```

This error means that the subscriber index was invalid.

```
(_cerr_sub_err = 54)    FH_EFDUNAVAIL (connection to event detector unavailable)
```

This error means that the event detector was unavailable.

```
(_cerr_sub_err = 56)    FH_EFDCONNERR (event detector connection error)
```

This error means that the EEM event detector that handles this request is not available.

# register\_timer

Registers a timer and returns a timer event ID. This Tcl command extension is used by a timer publisher to perform this registration before using the event ID to manipulate the timer if it does not use the **event\_register\_timer** command extension to register as a publisher and subscriber.

## Syntax

```
register_timer watchdog|countdown|absolute|cron name ?
```

## Arguments

name	(Mandatory) The name of the timer to be manipulated.
------	--

## Result String

```
event_id %u
```

Where `event_id` is the timer event ID for the specified timer (can be used to manipulate the timer by the **timer\_arm** or **timer\_cancel** command extensions).

## Set\_cerrno

Yes

```
(_cerr_sub_err = 2)    FH_ESYSERR    (generic/unknown error from OS/system)
```

This error means that the operating system reported an error. The POSIX `errno` value that is reported with the error should be used to determine the cause of the operating system error.

```
(_cerr_sub_err = 4)    FH_EINITONCE  (Init() is not yet done, or done twice.)
```

This error means that the request to register the specific event was made before the EEM event detector had completed its initialization.

```
(_cerr_sub_err = 6)    FH_EBADEVENTTYPE (unknown EEM event type)
```

This error means that the event type specified in the internal event specification was invalid.

```
(_cerr_sub_err = 9)    FH_EMEMORY    (insufficient memory for request)
```

This error means that an internal EEM request for memory failed.

```
(_cerr_sub_err = 10)   FH_ECORRUPT  (internal EEM API context is corrupt)
```

This error means that the internal EEM API context structure is corrupt.

```
(_cerr_sub_err = 11)   FH_ENOSUCHESID (unknown event specification ID)
```

This error means that the event specification ID could not be matched when the event was being registered or that an event detector internal event structure is corrupt.

```
(_cerr_sub_err = 16)   FH_EBADFMPPTR (bad ptr to fh_p data structure)
```

This error means that the context pointer that is used with each EEM API call is incorrect.

```
(_cerr_sub_err = 17)   FH_EBADADDRESS (bad API control block address)
```

This error means that a control block address that was passed in the EEM API was incorrect.



```
(_cerr_sub_err = 22)    FH_ENULLPTR    (event detector internal error - ptr is null)
```

This error means that an internal EEM event detector pointer was null when it should have contained a value.

```
(_cerr_sub_err = 25)    FH_ESUBSEXCEED    (number of subscribers exceeded)
```

This error means that the number of timer or counter subscribers exceeded the maximum.

```
(_cerr_sub_err = 26)    FH_ESUBSIDXINV    (invalid subscriber index)
```

This error means that the subscriber index was invalid.

```
(_cerr_sub_err = 54)    FH_EFDUNAVAIL    (connection to event detector unavailable)
```

This error means that the event detector was unavailable.

```
(_cerr_sub_err = 56)    FH_EFDCONNERR    (event detector connection error)
```

This error means that the EEM event detector that handles this request is not available.

# timer\_arm

Arms a timer. The type could be CRON, watchdog, countdown, or absolute.

## Syntax

```
timer_arm event_id ? cron_entry ?|time ?
```

## Arguments

event_id	(Mandatory) The timer event ID returned by the <b>register_timer</b> command extension. Must be an integer between 0 and 4294967295, inclusive.
cron_entry	(Mandatory) Must exist if the timer type is CRON. Must not exist for other types of timer. CRON timer specification uses the format of the CRON table entry.
time	(Mandatory) Must exist if the timer type is not CRON. Must not exist if the timer type is CRON. For watchdog and countdown timers, the number of seconds and milliseconds until the timer expires; for an absolute timer, the calendar time of the expiration time (specified in SSSSSSSSS[.MMM] format, where SSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). An absolute expiration date is the number of seconds and milliseconds since January 1, 1970. If the date specified has already passed, the timer expires immediately.

## Result String

```
sec_remain %ld msec_remain %ld
```

Where sec\_remain and msec\_remain are the remaining time before the next expiration of the timer.



### Note

A value of 0 will be returned for the sec\_remain and msec\_remain arguments if the timer type is CRON.

## Set\_cerrno

Yes

```
(_cerr_sub_err = 2)    FH_ESYSERR    (generic/unknown error from OS/system)
```

This error means that the operating system reported an error. The POSIX errno value that is reported with the error should be used to determine the cause of the operating system error.

```
(_cerr_sub_err = 6)    FH_EBADEVENTTYPE    (unknown EEM event type)
```

This error means that the event type specified in the internal event specification was invalid.

```
(_cerr_sub_err = 9)    FH_EMEMORY    (insufficient memory for request)
```

This error means that an internal EEM request for memory failed.

```
(_cerr_sub_err = 11)   FH_ENOSUCHESID    (unknown event specification ID)
```

This error means that the event specification ID could not be matched when the event was being registered or that an event detector internal event structure is corrupt.

```
(_cerr_sub_err = 12)    FH_ENOSUCHEID (unknown event ID)
```

This error means that the event ID could not be matched when the event was being registered or that an event detector internal event structure is corrupt.

```
(_cerr_sub_err = 22)    FH_ENULLPTR (event detector internal error - ptr is null)
```

This error means that an internal EEM event detector pointer was null when it should have contained a value.

```
(_cerr_sub_err = 27)    FH_ETMDELAYZR (zero delay time)
```

This error means that the time specified to arm a timer was zero.

```
(_cerr_sub_err = 42)    FH_ENOTREGISTERED (request for event spec that is unregistered)
```

This error means that the event was not registered.

```
(_cerr_sub_err = 54)    FH_EFDUNAVAIL (connection to event detector unavailable)
```

This error means that the event detector was unavailable.

```
(_cerr_sub_err = 56)    FH_EFDCONNERR (event detector connection error)
```

This error means that the EEM event detector that handles this request is not available.

# timer\_cancel

Cancels a timer.

## Syntax

```
timer_cancel event_id ?
```

## Arguments

event_id	(Mandatory) The timer event ID returned by the <b>register_timer</b> command extension. Must be an integer between 0 and 4294967295, inclusive.
----------	---

## Result String

```
sec_remain %ld msec_remain %ld
```

Where sec\_remain and msec\_remain are the remaining time before the next expiration of the timer.



### Note

A value of 0 will be returned for sec\_remain and msec\_remain if the timer type is CRON.

## Set\_cerrno

Yes

```
(_cerr_sub_err = 2)   FH_ESYSERR   (generic/unknown error from OS/system)
```

This error means that the operating system reported an error. The POSIX errno value that is reported with the error should be used to determine the cause of the operating system error.

```
(_cerr_sub_err = 6)   FH_EBADEVENTTYPE (unknown EEM event type)
```

This error means that the event type specified in the internal event specification was invalid.

```
(_cerr_sub_err = 7)   FH_ENOSUCHKEY  (could not find key)
```

This error means that the application event detector info key or other ID was not found.

```
(_cerr_sub_err = 11)  FH_ENOSUCHESID (unknown event specification ID)
```

This error means that the event specification ID could not be matched when the event was being registered or that an event detector internal event structure is corrupt.

```
(_cerr_sub_err = 12)  FH_ENOSUCHEID (unknown event ID)
```

This error means that the event ID could not be matched when the event was being registered or that an event detector internal event structure is corrupt.

```
(_cerr_sub_err = 22)  FH_ENULLPTR  (event detector internal error - ptr is null)
```

This error means that an internal EEM event detector pointer was null when it should have contained a value.

```
(_cerr_sub_err = 54)  FH_EFDUNAVAIL (connection to event detector unavailable)
```

This error means that the event detector was unavailable.

```
(_cerr_sub_err = 56)  FH_EFDCONNERR (event detector connection error)
```

This error means that the EEM event detector that handles this request is not available.

# unregister\_counter

Unregisters a counter. This Tcl command extension is used by a counter publisher to unregister a counter that was previously registered with the **register\_counter** Tcl command extension.

## Syntax

```
unregister_counter event_id ? event_spec_id ?
```

## Arguments

event_id	(Mandatory) Counter event ID returned by the <b>register_counter</b> command extension. Must be an integer between 0 and 4294967295, inclusive.
event_spec_id	(Mandatory) Counter event specification ID for the specified counter returned by the <b>register_counter</b> command extension. Must be an integer between 0 and 4294967295, inclusive.

## Result String

None

## Set\_cerrno

Yes

```
(_cerr_sub_err = 2)    FH_ESYSERR    (generic/unknown error from OS/system)
```

This error means that the operating system reported an error. The POSIX errno value that is reported with the error should be used to determine the cause of the operating system error.

```
(_cerr_sub_err = 9)    FH_EMEMORY    (insufficient memory for request)
```

This error means that an internal EEM request for memory failed.

```
(_cerr_sub_err = 11)   FH_ENOSUCHESID (unknown event specification ID)
```

This error means that the event specification ID could not be matched when the event was being registered or that an event detector internal event structure is corrupt.

```
(_cerr_sub_err = 22)   FH_ENULLPTR    (event detector internal error - ptr is null)
```

This error means that an internal EEM event detector pointer was null when it should have contained a value.

```
(_cerr_sub_err = 26)   FH_ESUBSIDXINV (invalid subscriber index)
```

This error means that the subscriber index was invalid.

```
(_cerr_sub_err = 54)   FH_EFDUNAVAIL (connection to event detector unavailable)
```

This error means that the event detector was unavailable.

```
(_cerr_sub_err = 56)   FH_EFDCONNERR (event detector connection error)
```

This error means that the EEM event detector that handles this request is not available.

## EEM System Information Tcl Command Extensions

- [sys\\_reqinfo\\_cli\\_freq](#), page 191
- [sys\\_reqinfo\\_cli\\_history](#), page 192
- [sys\\_reqinfo\\_cpu\\_all](#), page 193
- [sys\\_reqinfo\\_crash\\_history](#), page 194
- [sys\\_reqinfo\\_mem\\_all](#), page 195
- [sys\\_reqinfo\\_proc](#), page 197
- [sys\\_reqinfo\\_proc\\_all](#), page 199
- [sys\\_reqinfo\\_routename](#), page 200
- [sys\\_reqinfo\\_snmp](#), page 201
- [sys\\_reqinfo\\_syslog\\_freq](#), page 202
- [sys\\_reqinfo\\_syslog\\_history](#), page 204

**Note**

---

All EEM system information commands—**sys\_reqinfo\_xxx**—have the Set \_cerrno section set to yes.

---

# sys\_reqinfo\_cli\_freq

Queries the frequency information of all command-line interface (CLI) events.

**Syntax**

```
sys_reqinfo_cli_freq
```

**Arguments**

None

**Result String**

```
rec_list {{CLI frequency string 0},{CLI frequency str 1}, ...}
```

Where each CLI frequency string is:

```
time_sec %ld time_msec %ld match_count %u raise_count %u occurs %u period_sec %ld
period_msec %ld pattern {%s}
```

rec_list	Marks the start of the CLI event frequency list.
time_sec time_msec	Last time when this CLI event was raised.
match count	Number of times that a CLI command matches the pattern specified by this CLI event specification.
raise_count	Number of times that this CLI event was raised. The following fields are information about the CLI event specification: <ul style="list-style-type: none"> <li>• sync—A “yes” means that event publish should be performed synchronously. The event detector will be notified when the Event Manager Server has completed publishing the event. The Event Manager Server will return a code that indicates whether or not the CLI command should be executed.</li> <li>• skip—A “yes” means that the CLI command should not be executed if the sync flag is not set.</li> </ul>
occurs	Number of occurrences before an event is raised; if this argument is not specified, an event is raised on the first occurrence.
period_sec period_msec	Number of occurrences must occur within this number of POSIX timer units in order to raise event; if this argument is not specified, it does not apply.
pattern	Regular expression used to perform CLI command pattern matching.

**Set\_cerrno**

Yes

# sys\_reqinfo\_cli\_history

Queries the history of command-line interface (CLI) commands.

## Syntax

```
sys_reqinfo_cli_history
```

## Arguments

None

## Result String

```
rec_list {{CLI history string 0}, {CLI history str 1},...}
```

Where each CLI history string is:

```
time_sec %ld time_msec %ld cmd {%s}
```

rec_list	Marks the start of the CLI command history list.
time_sec time_msec	Time when the CLI command was run.
cmd	Text of the CLI command.

## Set\_cerrno

Yes



# sys\_reqinfo\_cpu\_all

Queries the CPU utilization of the top processes (both POSIX processes and IOS processes) during a specified time period and in a specified order. This Tcl command extension is supported only in Software Modularity images.

### Syntax

```
sys_reqinfo_cpu_all order cpu_used [sec ?] [msec ?] [num ?]
```

### Arguments

order	(Mandatory) Order used for sorting the CPU utilization of processes.
cpu_used	(Mandatory) Specifies that the average CPU utilization, for the specified time window, will be sorted in descending order.
sec msec	(Optional) The time period, in seconds and milliseconds, during which the average CPU utilization is calculated. Must be integers in the range from 0 to 4294967295. If not specified, or if both sec and msec are specified as 0, the most recent CPU sample is used.
num	(Optional) Number of entries from the top of the sorted list of processes to be displayed. Must be an integer in the range from 1 to 4294967295. Default value is 5.

### Result String

```
rec_list {{process CPU info string 0},{process CPU info string 1}, ...}
```

Where each process CPU info string is:

```
pid %u name {%s} cpu_used %u
```

rec_list	Marks the start of the process CPU information list.
pid	Process ID.
name	Process name.
cpu_used	Specifies that if sec and msec are specified with a number greater than zero, the average percentage is calculated from the process CPU utilization during the specified time period. If sec and msec are both zero or not specified, the average percentage is calculated from the process CPU utilization in the latest sample.

### Set\_cerrno

Yes

## sys\_reqinfo\_crash\_history

Queries the crash information of all processes that have ever crashed. This Tcl command extension is supported only in Software Modularity images.

### Syntax

```
sys_reqinfo_crash_history
```

### Arguments

None

### Result String

```
rec_list {{crash info string 0},{crash info string 1}, ...}
```

Where each crash info string is:

```
job_id %u name {%s} respawn_count %u fail_count %u dump_count %u
inst_id %d exit_status 0x%x exit_type %d proc_state {%s} component_id 0x%x
crash_time_sec %ld crash_time_msec %ld
```

job_id	System manager assigned job ID for the process. An integer between 1 and 4294967295, inclusive.
name	Process name.
respawn_count	Total number of restarts for the process.
fail_count	Number of restart attempts of the process. This count is reset to zero when the process is successfully restarted.
dump_count	Number of core dumps performed.
inst_id	Process instance ID.
exit_status	Last exit status of the process.
exit_type	Last exit type.
proc_state	Sysmgr process states. One of the following: error, forced_stop, hold, init, ready_to_run, run, run_rnode, stop, waitEOltimer, wait_rnode, wait_spawnntimer, wait_tpl.
component_id	Version manager assigned component ID for the component to which the process belongs.
crash_time_sec crash_time_msec	Seconds and milliseconds since January 1, 1970, which represent the last time the process crashed.

### Set\_cerrno

Yes

# sys\_reqinfo\_mem\_all

Queries the memory usage of the top processes (both POSIX and IOS) during a specified time period and in a specified order. This Tcl command extension is supported only in Software Modularity images.

### Syntax

```
sys_reqinfo_mem_all order allocates|increase|used [sec ?] [msec ?] [num ?]
```

### Arguments

order	(Mandatory) Order used for sorting the memory usage of processes.
allocates	(Mandatory) Specifies that the memory usage is sorted by the number of process allocations during the specified time window, and in descending order.
increase	(Mandatory) Specifies that the memory usage is sorted by the percentage of process memory increase during the specified time window, and in descending order.
used	(Mandatory) Specifies that the memory usage is sorted by the current memory used by the process.
sec msec	(Optional) The time period, in seconds and milliseconds, during which the process memory usage is calculated. Must be integers in the range from 0 to 4294967295. If both sec and msec are specified and are nonzero, the number of allocations is the difference between the number of allocations in the oldest and latest samples collected in the time period. The percentage is calculated as the the percentage difference between the memory used in the oldest and latest samples collected in the time period. If not specified, or if both sec and msec are specified as 0, the first sample ever collected is used as the oldest sample; that is, the time period is set to be the time from startup until the current moment.
num	(Optional) Number of entries from the top of the sorted list of processes to be displayed. Must be an integer in the range from 1 to 4294967295. Default value is 5.

### Result String

```
rec_list {{process mem info string 0},{process mem info string 1}, ...}
```

Where each process mem info string is:

```
pid %u name {%s} delta_allocs %d initial_alloc %u current_alloc %u percent_increase %d
```

rec_list	Marks the start of the process memory usage information list.
pid	Process ID.
name	Process name.
delta_allocs	Specifies the difference between the number of allocations in the oldest and latest samples collected in the time period.
initial_alloc	Specifies the amount of memory, in kilobytes, used by the process at the start of the time period.

current_alloc	Specifies the amount of memory, in kilobytes, currently used by the process.
percent_increase	Specifies the percentage difference between the memory used in the oldest and latest samples collected in the time period. The percentage difference can be expressed as $\text{current\_alloc} - \text{initial\_alloc}$ times 100 and divided by $\text{initial\_alloc}$ .

**Set\_cerrno**

Yes

# sys\_reqinfo\_proc

Queries the information about a single POSIX process. This Tcl command extension is supported only in Software Modularity images.

### Syntax

```
sys_reqinfo_proc job_id ?
```

### Arguments

job_id	(Mandatory) System manager assigned job ID for the process. Must be an integer between 1 and 4294967295, inclusive.
--------	---

### Result String

```
job_id %u component_id 0x%x name {%s} helper_name {%s} helper_path {%s} path {%s}
node_name {%s} is_respawn %u is_mandatory %u is_hold %u dump_option %d
max_dump_count %u respawn_count %u fail_count %u dump_count %u
last_respawn_sec %ld last_respawn_msec %ld inst_id %u proc_state %s
level %d exit_status 0x%x exit_type %d
```

job_id	System manager assigned job ID for the process. An integer between 1 and 4294967295, inclusive.
component_id	Version manager assigned component ID for the component to which the process belongs.
name	Process name.
helper_name	Helper process name.
helper_path	Executable path of the helper process.
path	Executable path of the process.
node_name	System manager assigned node name for the node to which the process belongs.
is_respawn	Flag that specifies that the process can be respawned.
is_mandatory	Flag that specifies that the process must be alive.
is_hold	Flag that specifies that the process is spawned until called by the API.
dump_option	Core dumping options.
max_dump_count	Maximum number of core dumping permitted.
respawn_count	Total number of restarts for the process.
fail_count	Number of restart attempts of the process. This count is reset to zero when the process is successfully restarted.
dump_count	Number of core dumps performed.
last_respawn_sec last_respawn_msec	Seconds and milliseconds in POSIX timer units since January 1, 1970, which represent the last time the process was started.
inst_id	Process instance ID.
proc_state	Sysmgr process states. One of the following: error, forced_stop, hold, init, ready_to_run, run, run_rnode, stop, waitEOLtimer, wait_rnode, wait_spawntimer, wait_tpl.

level	Process run level.
exit_status	Last exit status of the process.
exit_type	Last exit type.

**Set\_cerrno**

Yes

# sys\_reqinfo\_proc\_all

Queries the information of all POSIX processes. This Tcl command extension is supported only in Software Modularity images.

**Syntax**

```
sys_reqinfo_proc_all
```

**Arguments**

None

**Result String**

```
rec_list {{process info string 0}, {process info string 1},...}
```

Where each process info string is the same as the result string of the **sysreq\_info\_proc** Tcl command extension.

**Set\_cerrno**

Yes

## sys\_reqinfo\_routername

Queries the router name.

### Syntax

```
sys_reqinfo_routername
```

### Arguments

None

### Result String

```
routername %s
```

Where routername is the name of the router.

### Set\_cerrno

Yes



# sys\_reqinfo\_snmp

Queries the value of the entity specified by a Simple Network Management Protocol (SNMP) object ID.

### Syntax

```
sys_reqinfo_snmp oid ? get_type exact|next
```

### Arguments

oid	(Mandatory) SNMP OID in dot notation (for example, 1.3.6.1.2.1.2.1.0).
get_type	(Mandatory) Type of SNMP get operation that needs to be applied to the specified oid. If the get_type is “exact,” the value of the specified oid is retrieved; if the get_type is “next,” the value of the lexicographical successor to the specified oid is retrieved.

### Result String

```
oid {%s} value {%s}
```

oid	SNMP OID.
value	Value string of the associated SNMP data element.

### Set\_cerrno

Yes

```
(_cerr_sub_err = 2) FH_ESYSERR (generic/unknown error from OS/system)
```

This error means that the operating system reported an error. The POSIX errno value that is reported with the error should be used to determine the cause of the operating system error.

```
(_cerr_sub_err = 22) FH_ENULLPTR (event detector internal error - ptr is null)
```

This error means that an internal EEM event detector pointer was null when it should have contained a value.

```
(_cerr_sub_err = 37) FH_ENOSNMPDATA (can't retrieve data from SNMP)
```

This error means that there was no data for the SNMP object type.

```
(_cerr_sub_err = 51) FH_ESTATSTYP (invalid statistics data type)
```

This error means that the SNMP statistics data type was invalid.

```
(_cerr_sub_err = 54) FH_EFDUNAVAIL (connection to event detector unavailable)
```

This error means that the event detector was unavailable.

## sys\_reqinfo\_syslog\_freq

Queries the frequency information of all syslog events.

### Syntax

```
sys_reqinfo_syslog_freq
```

### Arguments

None

### Result String

```
rec_list {{event frequency string 0}, {log freq str 1}, ...}
```

Where each event frequency string is:

```
time_sec %ld time_msec %ld match_count %u raise_count %u occurs %u
period_sec %ld period_msec %ld pattern {%s}
```

time_sec time_msec	Seconds and milliseconds in POSIX timer units since January 1, 1970, which represent the time the last event was raised.
match_count	Number of times that a syslog message matches the pattern specified by this syslog event specification since event registration.
raise_count	Number of times that this syslog event was raised.
occurs	Number of occurrences needed in order to raise the event; if not specified, the event is raised on the first occurrence.
period_sec period_msec	Number of occurrences must occur within this number of POSIX timer units in order to raise the event; if not specified, the period check does not apply.
pattern	Regular expression used to perform syslog message pattern matching.

### Set\_cerrno

Yes

```
(_cerr_sub_err = 2)    FH_ESYSERR  (generic/unknown error from OS/system)
```

This error means that the operating system reported an error. The POSIX errno value that is reported with the error should be used to determine the cause of the operating system error.

```
(_cerr_sub_err = 9)    FH_EMEMORY  (insufficient memory for request)
```

This error means that an internal EEM request for memory failed.

```
(_cerr_sub_err = 22)   FH_ENULLPTR  (event detector internal error - ptr is null)
```

This error means that an internal EEM event detector pointer was null when it should have contained a value.

```
(_cerr_sub_err = 45)   FH_ESEQNUM  (sequence or workset number out of sync)
```

This error means that the event detector sequence or workset number was invalid.

```
(_cerr_sub_err = 46)   FH_EREGEMPTY  (registration list is empty)
```

This error means that the event detector registration list was empty.

```
(_cerr_sub_err = 54)    FH_EFDUNAVAIL (connection to event detector unavailable)
```

This error means that the event detector was unavailable.

# sys\_reqinfo\_syslog\_history

Queries the history of the specified syslog message.

## Syntax

```
sys_reqinfo_syslog_history
```

## Arguments

None

## Result String

```
rec_list {{log hist string 0}, {log hist str 1}, ...}
```

Where each log hist string is:

```
time_sec %ld time_msec %ld msg {%s}
```

time_sec	Seconds and milliseconds since January 1, 1970, which represent the
time_msec	time the message was logged.
msg	Syslog message.

## Set\_cerrno

Yes

```
(_cerr_sub_err = 2)    FH_ESYSERR  (generic/unknown error from OS/system)
```

This error means that the operating system reported an error. The POSIX errno value that is reported with the error should be used to determine the cause of the operating system error.

```
(_cerr_sub_err = 22)   FH_ENULLPTR  (event detector internal error - ptr is null)
```

This error means that an internal EEM event detector pointer was null when it should have contained a value.

```
(_cerr_sub_err = 44)   FH_EHISTEMPTY (history list is empty)
```

This error means that the history list was empty.

```
(_cerr_sub_err = 45)   FH_ESEQNUM  (sequence or workset number out of sync)
```

This error means that the event detector sequence or workset number was invalid.

```
(_cerr_sub_err = 54)   FH_EFDUNAVAIL (connection to event detector unavailable)
```

This error means that the event detector was unavailable.

## EEM Library Debug Command Extensions

- [cli\\_debug](#), page 206
- [smtp\\_debug](#), page 207

# cli\_debug

Prints a command-line interface (CLI) debug statement to syslog. This Tcl command extension is used to print a CLI debug statement to syslog if the **debug event manager tcl cli\_library** Cisco IOS CLI command is in effect.

## Syntax

```
cli_debug spec_string debug_string
```

## Arguments

spec_string	(Mandatory) The spec_string argument is used to indicate the type of debug statement.
debug_string	(Mandatory) The debug_string argument is used to indicate the debugging text.

## Result String

None

## Set\_cerrno

No

# smtp\_debug

Prints a Simple Mail Transfer Protocol (SMTP) debug statement to syslog. This Tcl command extension prints a SMTP debug statement to syslog if the **debug event manager tcl smtp\_library** Cisco IOS command-line interface (CLI) command is in effect.

## Syntax

```
smtp_debug spec_string debug_string
```

## Arguments

spec_string	(Mandatory) The spec_string argument is used to indicate the type of debug statement.
debug_string	(Mandatory) The debug_string argument is used to indicate the debugging text.

## Result String

None

## Set\_cerrno

No

## SMTP Library Command Extensions

All Simple Mail Transfer Protocol (SMTP) library command extensions belong to the `::cisco::lib` namespace.

To use this library, the user needs to provide an e-mail template file. The template file can include Tcl global variables so that the e-mail service and the e-mail text can be configured through the **event manager environment Cisco IOS** command-line interface (CLI) configuration command. There are commands in this library to substitute the global variables in the e-mail template file and to send the desired e-mail context with the To address, CC address, From address, and Subject line properly configured using the configured e-mail server.

### E-Mail Template

The e-mail template file has the following format:



#### Note

Based on RFC 2554, the SMTP e-mail server name—Mailservername— can be in any one of the following template formats: `username:password@host`, `username@host`, or `host`.

```
Mailservername:<space><the list of candidate SMTP server addresses>
From:<space><the e-mail address of sender>
To:<space><the list of e-mail addresses of recipients>
Cc:<space><the list of e-mail addresses that the e-mail will be copied to>
Sourceaddr:<space><the IP addresses of the recipients>
Subject:<subject line>
<a blank line>
<body>
```



#### Note

Note that the template normally includes Tcl global variables to be configured.

Below is a sample e-mail template file:

```
Mailservername: $_email_server
From: $_email_from
To: $_email_to
Cc: $_email_cc
Sourceaddr: $_email_ipaddr
Subject: From router $routername: Process terminated

process name: $process_name
subsystem: $sub_system
exit status: $exit_status
respawn count: $respawn_count
```

### Exported Tcl Command Extensions

- [smtp\\_send\\_email](#), page 209
- [smtp\\_subst](#), page 210



# smtp\_send\_email

Given the text of an e-mail template file with all global variables already substituted, sends the e-mail out using Simple Mail Transfer Protocol (SMTP). The e-mail template specifies the candidate mail server addresses, To addresses, CC addresses, From address, subject line, and e-mail body.



## Note

A list of candidate e-mail servers can be provided so that the library will try to connect the servers on the list one by one until it can successfully connect to one of them.

## Syntax

```
smtp_send_email text
```

## Arguments

text	(Mandatory) The text of an e-mail template file with all global variables already substituted.
------	--

## Result String

None

## Set\_cerrno

- Wrong 1st line format—Mailservername:list of server names.
- Wrong 2nd line format—From:from-address.
- Wrong 3rd line format—To:list of to-addresses.
- Wrong 4th line format—CC:list of cc-addresses.
- Error connecting to mail server:—\$sock closed by remote server (where \$sock is the name of the socket opened to the mail server).
- Error connecting to mail server:—\$sock reply code is \$k instead of the service ready greeting (where \$sock is the name of the socket opened to the mail server; \$k is the reply code of \$sock).
- Error connecting to mail server:—cannot connect to all the candidate mail servers.
- Error disconnecting from mail server:—\$sock closed by remote server (where \$sock is the name of the socket opened to the mail server).

## Sample Scripts

After all needed global variables in the e-mail template are defined:

```
if [catch {smtp_subst [file join $tcl_library email_template_sm]} result] {
    puts stderr $result
    exit 1
}
if [catch {smtp_send_email $result} result] {
    puts stderr $result
    exit 1
}
```

## smtp\_subst

Given an e-mail template file `e-mail_template`, substitutes each global variable in the file by its user-defined value. Returns the text of the file after substitution.

### Syntax

```
smtp_subst e-mail_template
```

### Arguments

<code>e-mail_template</code>	(Mandatory) Name of an e-mail template file in which global variables need to be substituted by a user-defined value. An example filename could be <code>/disk0://example.template</code> which represents a file named <code>example.template</code> in a top-level directory on an ATA flash disk in slot 0.
------------------------------	--

### Result String

The text of the e-mail template file with all the global variables substituted.

### Set\_cerrno

- cannot open e-mail template file
- cannot close e-mail template file

## CLI Library Command Extensions

All command-line interface (CLI) library command extensions belong to the `::cisco::eem` namespace.

This library provides users the ability to run CLI commands and get the output of the commands in Tcl. Users can use commands in this library to spawn an exec and open a virtual terminal channel to it, write the command to execute to the channel so that the command will be executed by exec, and read back the output of the command.

There are two types of CLI commands: interactive commands and non-interactive commands.

For interactive commands, after the command is entered, there will be a “Q&A” phase in which the router will ask for different user options, and the user is supposed to enter the answer for each question. Only after all the questions have been answered properly will the command run according to the user’s options until completion.

For noninteractive commands, once the command is entered, the command will run to completion. To run different types of commands using an EEM script, different CLI library command sequences should be used, which are documented in the [“Using the CLI Library to Run a Noninteractive Command” section on page 224](#) and in the [“Using the CLI Library to Run an Interactive Command” section on page 224](#).

The vty lines are allocated from the pool of vty lines that are configured using the **line vty** CLI configuration command. EEM will use a vty line when a vty line is not being used by EEM and there are available vty lines. EEM will also use a vty line when EEM is already using a vty line and there are three or more vty lines available. Be aware that the connection will fail when fewer than three vty lines are available, preserving the remaining vty lines for Telnet use.

In Cisco IOS Release 12.4(22)T, and later releases, XML-PI support was introduced. For details about the XML-PI support, the new CLI library command extensions, and some examples of how to implement XML-PI, see [“CLI Library XML-PI Support” section on page 227](#).

### Exported Tcl Command Extensions

- [cli\\_close](#), page 212
- [cli\\_exec](#), page 213
- [cli\\_get\\_ttyname](#), page 214
- [cli\\_open](#), page 215
- [cli\\_read](#), page 216
- [cli\\_read\\_drain](#), page 217
- [cli\\_read\\_line](#), page 218
- [cli\\_read\\_pattern](#), page 219
- [cli\\_run](#), page 220
- [cli\\_run\\_interactive](#), page 221
- [cli\\_write](#), page 223

# cli\_close

Closes the exec process and releases the vty and the specified channel handler connected to the command-line interface (CLI).

## Syntax

```
cli_close fd tty_id
```

## Arguments

fd	(Mandatory) The CLI channel handler.
tty_id	(Mandatory) The TTY ID returned from the <b>cli_open</b> command extension.

## Result String

None

## Set\_cerrno

Cannot close the channel.

# cli\_exec

Writes the command to the specified channel handler to execute the command. Then reads the output of the command from the channel and returns the output.

## Syntax

```
cli_exec fd cmd
```

## Arguments

fd	(Mandatory) The command-line interface (CLI) channel handler.
cmd	(Mandatory) The CLI command to execute.

## Result String

The output of the CLI command executed.

## Set\_cerrno

Error reading the channel.

# cli\_get\_ttyname

Returns the real and pseudo TTY names for a given TTY ID.

## Syntax

```
cli_get_ttyname tty_id
```

## Arguments

tty_id	(Mandatory) The TTY ID returned from the <b>cli_open</b> command extension.
--------	---

## Result String

```
pty %s tty %s
```

## Set\_cerrno

None

# cli\_open

Allocates a vty, creates an EXEC command-line interface (CLI) session, and connects the vty to a channel handler. Returns an array including the channel handler.



## Note

Each call to **cli\_open** initiates a Cisco IOS EXEC session that allocates a Cisco IOS vty line. The vty remains in use until the **cli\_close** routine is called. The vty lines are allocated from the pool of vty lines that are configured using the **line vty** CLI configuration command. EEM will use a vty line when a vty line is not being used by EEM and there are available vty lines. EEM will also use a vty line when EEM is already using a vty line and there are three or more vty lines available. Be aware that the connection will fail when fewer than three vty lines are available, preserving the remaining vty lines for Telnet use

## Syntax

```
cli_open
```

## Arguments

None

## Result String

```
"tty_id {s} pty {d} tty {d} fd {d}"
```

Event Type	Description
<b>tty_id</b>	TTY ID.
<b>pty</b>	PTY device name.
<b>tty</b>	TTY device name.
<b>fd</b>	CLI channel handler.

## Set\_cerrno

- Cannot get pty for EXEC.
- Cannot create an EXEC CLI session.
- Error reading the first prompt.

# cli\_read

Reads the command output from the specified command-line interface (CLI) channel handler until the pattern of the router prompt occurs in the contents read. Returns all the contents read up to the match.

## Syntax

```
cli_read fd
```

## Arguments

fd	(Mandatory) The CLI channel handler.
----	--------------------------------------

## Result String

All the contents read.

## Set\_cerrno

Cannot get router name.



## Note

This Tcl command extension will block waiting for the router prompt to show up in the contents read.



# cli\_read\_drain

Reads and drains the command output of the specified command-line interface (CLI) channel handler. Returns all the contents read.

## Syntax

```
cli_read_drain fd
```

## Arguments

fd	(Mandatory) The CLI channel handler.
----	--------------------------------------

## Result String

All the contents read.

## Set\_cerrno

None

## cli\_read\_line

Reads one line of the command output from the specified command-line interface (CLI) channel handler. Returns the line read.

### Syntax

```
cli_read_line fd
```

### Arguments

fd	(Mandatory) The CLI channel handler.
----	--------------------------------------

### Result String

The line read.

### Set\_cerrno

None

**Note**

This Tcl command extension will block waiting for the end of line to show up in the contents read.

## cli\_read\_pattern

Reads the command output from the specified command-line interface (CLI) channel handler until the pattern that is to be matched occurs in the contents read. Returns all the contents read up to the match.



### Note

The pattern matching logic attempts a match by looking at the command output data as it is delivered from the Cisco IOS command. The match is always done on the most recent 256 characters in the output buffer unless there are fewer characters available, in which case the match is done on fewer characters. If more than 256 characters in the output buffer are required for the match to succeed, the pattern will not match.

### Syntax

```
cli_read_pattern fd ptn
```

### Arguments

fd	(Mandatory) The CLI channel handler.
ptn	(Mandatory) The pattern to be matched when reading the command output from the channel.

### Result String

All the contents read.

### Set\_cerrno

None



### Note

This Tcl command extension will block waiting for the specified pattern to show up in the contents read.

# cli\_run

Iterates over the items in the `clist` and assumes that each one is a command-line-interface (CLI) command to be executed in the enable mode. On success, returns the output of all executed commands and on failure, returns error from the failure.

## Syntax

```
cli_run clist
```

## Arguments

<code>clist</code>	(Mandatory) The list of commands to be executed.
--------------------	--

## Result String

Output of all the commands that are executed or an error message.

## Set\_cerrno

None.

## Sample Usage

The following example shows how to use the `cli_run` command extension.

```
set clist [list {sh run} {sh ver} {sh event man pol reg}]
cli_run { clist }
```

## cli\_run\_interactive

Provides a sublist to the clist which has four items. On success, returns the output of all executed commands and on failure, returns error from the failure. Also uses arrays when possible as a way of making things easier to read later by keeping expect and reply separated.

### Syntax

```
cli_run_interactive clist
```

### Arguments

clist	<p>(Mandatory) Sublist which has four items and each item has four subitems:</p> <ul style="list-style-type: none"> <li>• command <ul style="list-style-type: none"> <li>– expect</li> <li>– an expected question</li> <li>– reply</li> <li>– reply to this question</li> </ul> </li> <li>• a command to run <ul style="list-style-type: none"> <li>– expect</li> <li>– an expected question</li> <li>– reply</li> <li>– reply to this question</li> </ul> </li> <li>• responses <ul style="list-style-type: none"> <li>– expect</li> <li>– an expected question</li> <li>– reply</li> <li>– reply to this question</li> </ul> </li> <li>• a list of what to expect and what to reply. <ul style="list-style-type: none"> <li>– expect</li> <li>– an expected question</li> <li>– reply</li> <li>– reply to this question</li> </ul> </li> </ul>
-------	--

### Result String

Output of all the commands that are executed or an error message.

### Set\_cerrno

None.

### Sample Usage

The following example shows how to use the cli\_ru\_ interactive command extension.

```
set cmd1 "first command"
set cmd1_exp1 {[confirm]}
```

```
set cmd1_rep1 {y}
set cmd1_response [list [list expect $cmd1_exp1 reply $cmd1_rep1]]

set cmd2 "second command"
set cmd2_exp1 {save config}
set cmd2_rep1 {no}
set cmd2_exp2 {[confirm]}
set cmd2_rep2 {y}
set cmd2_response [list [list expect $cmd2_exp1 reply $cmd2_rep1] [list expect $cmd2_exp2
reply $cmd2_rep2]]

set cmd3 "third command"
set cmd3_exp1 {are you sure}
set cmd3_rep1 {yes}
set cmd3_exp2 {destination file}
set cmd3_rep2 {test.txt}
set cmd2_response [list [list expect $cmd3_exp1 reply $cmd3_rep1] [list expect $cmd3_exp2
reply $cmd3_rep2]]

set clist [list " command $cmd1 responses $cmd1_response" " command $cmd2 responses
$cmd2_response" " command $cmd3 responses $cmd3_response"]
cli_run_interactive { clist }
```

# cli\_write

Writes the command that is to be executed to the specified CLI channel handler. The CLI channel handler executes the command.

## Syntax

```
cli_write fd cmd
```

## Arguments

fd	(Mandatory) The CLI channel handler.
cmd	(Mandatory) The CLI command to execute.

## Result String

None

## Set\_cerrno

None

## Sample Usage

As an example, use configuration CLI commands to bring up Ethernet interface 1/0:

```
if [catch {cli_open} result] {
  puts stderr $result
  exit 1
} else {
  array set cli1 $result
}
if [catch {cli_exec $cli1(fd) "en"} result] {
  puts stderr $result
  exit 1
}
if [catch {cli_exec $cli1(fd) "config t"} result] {
  puts stderr $result
  exit 1
}
if [catch {cli_exec $cli1(fd) "interface Ethernet1/0"} result] {
  puts stderr $result
  exit 1
}
if [catch {cli_exec $cli1(fd) "no shut"} result] {
  puts stderr $result
  exit 1
}
if [catch {cli_exec $cli1(fd) "end"} result] {
  puts stderr $result
  exit 1
}
if [catch {cli_close $cli1(fd) $cli1(tty_id)} result] {
  puts stderr $result
  exit 1
}
```

### Using the CLI Library to Run a Noninteractive Command

To run a noninteractive command, use the **cli\_exec** command extension to issue the command, and then wait for the complete output and the router prompt. For example, the following shows the use of configuration CLI commands to bring up Ethernet interface 1/0:

```
if [catch {cli_open} result] {
error $result $errorInfo
} else {
set fd $result
}
if [catch {cli_exec $fd "en"} result] {
error $result $errorInfo
}
if [catch {cli_exec $fd "config t"} result] {
error $result $errorInfo
}
if [catch {cli_exec $fd "interface Ethernet1/0"} result] {
error $result $errorInfo
}
if [catch {cli_exec $fd "no shut"} result] {
error $result $errorInfo
}
if [catch {cli_exec $fd "end"} result] {
error $result $errorInfo
}
if [catch {cli_close $fd} result] {
error $result $errorInfo
}
```

### Using the CLI Library to Run an Interactive Command

To run interactive commands, three phases are needed:

- Phase 1: Issue the command using the **cli\_write** command extension.
- Phase 2: Q&A Phase. Use the **cli\_read\_pattern** command extension to read the question (the regular pattern that is specified to match the question text) and the **cli\_write** command extension to write back the answers alternately.
- Phase 3: Noninteractive phase. All questions have been answered, and the command will run to completion. Use the **cli\_read** command extension to wait for the complete output of the command and the router prompt.

For example, use CLI commands to do squeeze bootflash: and save the output of this command in the Tcl variable `cmd_output`.

```
if [catch {cli_open} result] {
error $result $errorInfo
} else {
array set cli1 $result
}
if [catch {cli_exec $cli1(fd) "en"} result] {
error $result $errorInfo
}

# Phase 1: issue the command
if [catch {cli_write $cli1(fd) "squeeze bootflash:"} result] {
error $result $errorInfo
}

# Phase 2: Q&A phase
# wait for prompted question:
# All deleted files will be removed. Continue? [confirm]
if [catch {cli_read_pattern $cli1(fd) "All deleted"} result] {
```



```

error $result $errorInfo
}
# write a newline character
if [catch {cli_write $cli1(fd) "\n"} result] {
error $result $errorInfo
}
# wait for prompted question:
# Squeeze operation may take a while. Continue? [confirm]
if [catch {cli_read_pattern $cli1(fd) "Squeeze operation"} result] {
error $result $errorInfo
}
# write a newline character
if [catch {cli_write $cli1(fd) "\n"} result] {
error $result $errorInfo
}

# Phase 3: noninteractive phase
# wait for command to complete and the router prompt
if [catch {cli_read $cli1(fd) } result] {
error $result $errorInfo
} else {
set cmd_output $result
}
if [catch {cli_close $cli1(fd) $cli1(tty_id)} result] {
error $result $errorInfo
}

```

The following example causes a router to be reloaded using the CLI **reload** command. Note that the EEM **action\_reload** command accomplishes the same result in a more efficient manner, but this example is presented to illustrate the flexibility of the CLI library for interactive command execution.

```

# 1. execute the reload command
if [catch {cli_open} result] {
    error $result $errorInfo
} else {
    array set cli1 $result
}
if [catch {cli_exec $cli1(fd) "en"} result] {
    error $result $errorInfo
}
if [catch {cli_write $cli1(fd) "reload"} result] {
    error $result $errorInfo
} else {
    set cmd_output $result
}
if [catch {cli_read_pattern $cli1(fd) ".*(System configuration has been modified. Save\\?
\\[yes/no\\]: )"} result] {
    error $result $errorInfo
} else {
    set cmd_output $result
}
if [catch {cli_write $cli1(fd) "no"} result] {
    error $result $errorInfo
} else {
    set cmd_output $result
}
if [catch {cli_read_pattern $cli1(fd) ".*(Proceed with reload\\? \\[confirm\\])"}
result] {
    error $result $errorInfo
} else {
    set cmd_output $result
}

```

```
if [catch {cli_write $cli1(fd) "y"} result] {
    error $result $errorInfo
} else {
    set cmd_output $result
}
if [catch {cli_close $cli1(fd) $cli1(tty_id)} result] {
    error $result $errorInfo
}
```

## CLI Library XML-PI Support

XML Programmatic Interface (XML-PI) was introduced in Cisco IOS Release 12.4(22)T. XML-PI provides a programmable interface which encapsulates IOS command-line interface (CLI) show commands in XML format in a consistent way across different Cisco products. Customers using XML-PI will be able to parse IOS show command output from within Tcl scripts using well-known keywords instead of having to depend on the use of regular expression support to “screen-scrape” output.

The benefit of using the XML-PI command extensions is to facilitate the extraction of specific output information that is generated using a CLI **show** command. Most show commands return many fields within the output and currently a regular expression has to be used to extract specific information that may appear in the middle of a line. XML-PI support provides a set of Tcl library functions to facilitate the parsing of output from the IOS CLI format extension in the form of:

```
show <show-command> | format {spec-file}
```

where a spec-file is a concatenation of all Spec File Entries (SFE) for each **show** command currently supported. As part of the XML-PI project a default spec-file will be included in the IOS Release 12.4(22)T images. The default spec-file will have a small set of commands and the SFE for the commands will have a subset of the possible tags. If no spec-file is provided with the format command, the default spec-file is used.

For examples of implementing the XML-PI feature, see the [“Sample Usage of the XML-PI feature” section on page 231](#).

For more general details about XML-PI, see the [“XML-PI” module](#).

The following Tcl command extensions were introduced to support XML-PI:

- [xml\\_pi\\_exec, page 228](#)
- [xml\\_pi\\_parse, page 229](#)
- [xml\\_pi\\_read, page 230](#)
- [xml\\_pi\\_write, page 231](#)

## xml\_pi\_exec

Writes the XML-PI command specified using the `cmd` argument to the channel whose handler is specified using the `fd` argument and the spec-file specified by the `spec_file` argument to execute the command. The raw XML output data of the command is then read from the channel and the XML output is returned.

### Syntax

```
xml_pi_show fd cmd [spec_file]
```

### Arguments

<code>fd</code>	(Mandatory) The CLI library file descriptor obtained from <code>cli_open</code> .
<code>cmd</code>	(Mandatory) IOS show command.
<code>spec_file</code>	(Optional) IOS CLI show command <code>spec_file</code> .

### Result String

Result of IOS show command in XML format.

### Set\_cerrno

Possible error raised:

1. error reading the channel

## xml\_pi\_parse

Processes the XML show command raw output passed into this function as `xml_data` and retrieve those fields that are specified by `xml_tags_list`. The following processing occurs:

Step 1: The XML tag list is validated as a Tcl list. An XML tag can be specified as the low order XML tag name or as a fully qualified XML tag name in case the low order name is ambiguous for a given command.

Example tags:

```
<Interface>
```

```
<ShowIpInterfaceBrief><IPInterfaces><entry><Interface>
```

Step 2: The `xml_data` is validated as valid XML and parsed into an XML parse tree.

Step 3: A walk is made through the XML parse tree and each tag is compared with entries in the XML tag list. When a match occurs it is determined if the tag name matches a Tcl procedure defined within the current Tcl scope. If so, that Tcl procedure will be called with the current result. If not, the tag name and the data associated with that tag name will be appended to the current result.

### Syntax

```
xml_pi_parse fd xml_show_cmd_output xml_tags_list
```

### Arguments

<code>fd</code>	(Mandatory) The CLI library file descriptor obtained from <code>cli_open</code> .
<code>xml_show_cmd_output</code>	(Mandatory) Output of <code>xml_pi_show</code> command extension in xml format.
<code>xml_tags_list</code>	(Mandatory) List of interesting tags.

### Result String

Data in a Tcl array indexed by XML tag name.



### Note

The current result is reset after Tcl procedure calls.

### Set\_cerrno

Possible errors raised:

1. error splitting the XML tags list
2. null XML tag list specified
3. XML tag tree exceeds 20 levels
4. called Tcl procedure returned an error
5. memory allocation failure
6. XML parse failure
7. failed to create XML domain

## xml\_pi\_read

Reads the XML-PI command output (from the specified show command) from the CLI channel whose handler is given by the file descriptor until the pattern of the router prompt occurs in the contents that are read. Returns all the contents read up to the match in XML format.

### Syntax

```
xml_pi_read fd
```

### Arguments

fd	(Mandatory) The CLI library file descriptor obtained from cli_open.
----	---

### Result String

All the contents that are read in XML format.

### Set\_cerrno

Possible errors raised:

1. cannot get router name
2. command error

# xml\_pi\_write

Writes the XML-PI command specified using the `cmd` argument to the channel whose handler is given by the `fd` argument and the spec file specified by the `spec_file` argument.

## Syntax

```
xml_pi_write fd cmd spec_file
```

## Arguments

<code>fd</code>	(Mandatory) The CLI library file descriptor obtained from <code>cli_open</code> .
<code>cmd</code>	(Mandatory) IOS show command.
<code>spec_file</code>	(Optional) IOS CLI show command <code>spec_file</code> .

## Result String

None

## Set\_cerrno

None

## Sample Usage of the XML-PI feature

The following EEM policy (`sample.tcl`) presents one example that illustrates five different implementations of the new EEM XML-PI functionality. The `odm` spec-file (required for Example 2) follows this policy.

```
::cisco::eem::event_register_none maxrun 60
namespace import ::cisco::eem::*
namespace import ::cisco::lib::*
# open the cli_lib.tcl channel
if [catch {cli_open} result] {
    error $result $errorInfo
} else {
    array set cli1 $result
}
# enter "enable" privilege mode
if [catch {cli_exec $cli1(fd) "en"} result] {
    error $result $errorInfo
}
# Example 1:
#
# Detect if XML-PI is present in this image
# Invoke xml_pi_exec with the default spec file for the "show inventory"
# command. After the command executes $result contains the raw XML data if
# the command is successful.
if [catch {xml_pi_exec $cli1(fd) "show inventory" ""} result] {
    puts "Example 1: XML-PI support is not present in this image - exiting"
    exit
} else {
    puts "Example 1: XML-PI support is present in this image"
}
# Example 2:
#
# In the next example we demonstrate how to extract two data elements
# from the "show version" command using the specified XML-PI spec file.
# The raw output from this command is as follows:
#
```

```

# router#show version | format disk2:speceemtest.odm
# <?xml version="1.0" encoding="UTF-8"?>
# <ShowVersion>
# <Version>12.4(20071029:194217)</Version>
# <Compiled>Thu 08-Nov-07 11:28</Compiled>
# <ROM>System Bootstrap, Version 12.2(20030826:190624) [BLD-npeg1_rommon_r11 102],
DEVELOPMENT</ROM>
# <uptime>17 minutes</uptime>
# <processor>NPE-G1</processor>
# <bytesofmemory>983040K/65536K</bytesofmemory>
# <CPU>700MHz</CPU>
# <L2Cache>0.2</L2Cache>
# <GigabitEthernetinterfaces>3</GigabitEthernetinterfaces>
# <bytesofNVRAM>509K</bytesofNVRAM>
# <bytesofATAPCMCIAcard>125952K</bytesofATAPCMCIAcard>
# <Sectorsize>512 bytes</Sectorsize>
# <bytesofFlashinternalSIMM>16384K</bytesofFlashinternalSIMM>
# <Configurationregister>0x2100</Configurationregister>
# </ShowVersion>
#
# Invoke xml_pi_exec with the spec file "disk2:speceemtest.odm" for the
# "show version" command. After the command executes $result contains
# the raw XML data.
if [catch {xml_pi_exec $cli1(fd) "show version" "disk2:speceemtest.odm"} result] {
error $result $errorInfo
} else {
# Pass the raw XML data to the xml_pi_parse routine to extract fields
# of interest:
# we ask that only the <processor> and <CPU> fields be returned.
array set xml_result [xml_pi_parse $cli1(fd) $result "<processor> <CPU>"]
puts "Example 2: Processor is $xml_result(<processor>) CPU is $xml_result(<CPU>)"
}
# Example 3:
#
# In the next example we demonstrate how to extract two data elements
# from the multi-record "show inventory" command using the default built-in
# XML-PI spec file. Sample raw output from this command is as follows:
#
# router#show inventory | format
# <?xml version="1.0" encoding="UTF-8"?>
# <ShowInventory>
# <SpecVersion>built-in</SpecVersion>
# <InventoryEntry>
# <ChassisName>&quot;Chassis&quot;</ChassisName>
# <Description>&quot;Cisco 7206VXR, 6-slot chassis&quot;</Description>
# <PID>CISCO7206VXR</PID>
# <VID>
# </VID>
# <SN>31413378 </SN>
# </InventoryEntry>
# <InventoryEntry>
# <ChassisName>&quot;NPE-G1 0&quot;</ChassisName>
# <Description>&quot;Cisco 7200 Series Network Processing Engine
NPE-G1&quot;</Description>
# <PID>NPE-G1</PID>
# <VID>
# </VID>
# <SN>31493825 </SN>
# </InventoryEntry>
# <InventoryEntry>
# <ChassisName>&quot;disk2&quot;</ChassisName>
# <Description>&quot;128MB Compact Flash Disk for NPE-G1&quot;</Description>
# <PID>MEM-NPE-G1-FLD128</PID>
# <VID>

```



```

# </VID>
# <SN>NAME: &quot;module 1&quot;;</SN>
# </InventoryEntry>
# <InventoryEntry>
# <ChassisName>&quot;module 1&quot;;</ChassisName>
# <Description>&quot;Dual Port FastEthernet (RJ45)&quot;;</Description>
# <PID>PA-2FE-TX</PID>
# <VID>
# </VID>
# <SN>JAE0827NGKX</SN>
# </InventoryEntry>
# <InventoryEntry>
# <ChassisName>&quot;Power Supply 2&quot;;</ChassisName>
# <Description>&quot;Cisco 7200 AC Power Supply&quot;;</Description>
# <PID>PWR-7200-AC</PID>
# <VID>
# </VID>
# </InventoryEntry>
# </ShowInventory>
#
# Define a procedure to be called every time the <InventoryEntry> tag
# is processed. Since this tag precedes each new output record, the data
# that is passed into this procedure contains the fields that have been
# requested via xml_pi_parse since the previous time this procedure was
# called.
proc <InventoryEntry> {xml_line} {
    global num
    # The first time that this function is called there is no data and
    # xml_line will be null.
    if [string length $xml_line] {
        array set xml_result $xml_line
        incr num
        set output [format "Example 3: Item %2d %-18s %s" \
            $num $xml_result(<PID>) $xml_result(<Description>)]
        puts $output
    }
    set num 0
    # Invoke xml_pi_exec with the default built-in spec file for the
    # "show inventory" command. After the command executes $result contains
    # the raw XML data.
    if [catch {xml_pi_exec $cli1(fd) "show inventory"} result] {
        error $result $errorInfo
    } else {
        # Pass the raw XML data to the xml_pi_parse routine to extract fields
        # of interest:
        # we ask that only the <PID> and <Description> fields be returned.
        # If an XML tag name is requested and a Tcl proc exists with that name,
        # the Tcl proc will be called every time that tag is encountered in the
        # output data. Specify the <InventoryEntry> tag and define the proc
        # before executing the xml_pi_parse statement.
        array set xml_result [xml_pi_parse $cli1(fd) $result \
            "<InventoryEntry> <PID> <Description>"]
        # Display the data from the last record.
        incr num
        set output [format "Example 3: Item %2d %-18s %s" \
            $num $xml_result(<PID>) $xml_result(<Description>)]
        puts $output
    }
    # Example 4:
    #
    # In the next example we demonstrate how to extract two data elements
    # from the multi-record "show ip interface brief" command using the default
    # built-in XML-PI spec file. Sample raw output from this command is as

```

```

# follows:
#
# router#show ip interface brief | format
# <?xml version="1.0" encoding="UTF-8"?>
# <ShowIpInterfaceBrief>
# <SpecVersion>built-in</SpecVersion>
# <IPInterfaces>
# <entry>
# <Interface>GigabitEthernet0/1</Interface>
# <IP-Address>172.19.209.34</IP-Address>
# <OK>YES</OK>
# <Method>NVRAM</Method>
# <Status>up</Status>
# <Protocol>up</Protocol>
# </entry>
# <entry>
# <Interface>GigabitEthernet0/2</Interface>
# <IP-Address>unassigned</IP-Address>
# <OK>YES</OK>
# <Method>NVRAM</Method>
# <Status>administratively down</Status>
# <Protocol>down</Protocol>
# </entry>
# <entry>
# <Interface>GigabitEthernet0/3</Interface>
# <IP-Address>unassigned</IP-Address>
# <OK>YES</OK>
# <Method>NVRAM</Method>
# <Status>administratively down</Status>
# <Protocol>down</Protocol>
# </entry>
# <entry>
# <Interface>FastEthernet1/0</Interface>
# <IP-Address>unassigned</IP-Address>
# <OK>YES</OK>
# <Method>NVRAM</Method>
# <Status>administratively down</Status>
# <Protocol>down</Protocol>
# </entry>
# <entry>
# <Interface>FastEthernet1/1</Interface>
# <IP-Address>unassigned</IP-Address>
# <OK>YES</OK>
# <Method>NVRAM</Method>
# <Status>administratively down</Status>
# <Protocol>down</Protocol>
# </entry>
# </IPInterfaces>
# </ShowIpInterfaceBrief>
#
# Define a procedure to be called every time the fully qualified name
# <ShowIpInterfaceBrief><IPInterfaces><entry> tag is processed. Since
# this tag precedes each new output record, the data that is passed into
# this procedure contains the fields that have been requested via
# xml_pi_parse since the previous time this procedure was called.
proc <ShowIpInterfaceBrief><IPInterfaces><entry> {xml_line} {
global num
# The first time that this function is called there is no data and
# xml_line will be null.
if [string length $xml_line] {
array set xml_result $xml_line
incr num
set output [format "Example 4: Interface %2d %-30s %s" \
$num $xml_result(<Interface>) $xml_result(<Status>)]

```

```

puts $output
} else {
puts "Example 4: Display All Interfaces"
}
}
set num 0
# Invoke xml_pi_exec with the default built-in spec file for the
# "show ip interface brief" command. After the command executes $result
# contains the raw XML data.
if [catch {xml_pi_exec $cli1(fd) "show ip interface brief"} result] {
error $result $errorInfo
} else {
# Pass the raw XML data to the xml_pi_parse routine to extract fields
# of interest:
# we ask that only the <Interface> and <Status> fields be returned.
# If an XML tag name is requested and a Tcl proc exists with that name,
# the Tcl proc will be called every time that tag is encountered in the
# output data. Specify the <entry> tag and define the proc
# before executing the xml_pi_parse statement.
array set xml_result [xml_pi_parse $cli1(fd) $result \
"<ShowIpInterfaceBrief><IPInterfaces><entry> <Interface> <Status>"]
# Display the data from the last record.
incr num
set output [format "Example 4: Interface %2d %-30s %s" \
$num $xml_result(<Interface>) $xml_result(<Status>)]
puts $output
}
# Example 5:
#
# In the next example we demonstrate how to extract two data elements
# from the multi-record "show ip interface brief" command using the default
# built-in XML-PI spec file. Sample raw output from this command is as
# follows:
#
# router#show ip interface brief | format
# <?xml version="1.0" encoding="UTF-8"?>
# <ShowIpInterfaceBrief>
# <SpecVersion>built-in</SpecVersion>
# <IPInterfaces>
# <entry>
# <Interface>GigabitEthernet0/1</Interface>
# <IP-Address>172.19.209.34</IP-Address>
# <OK>YES</OK>
# <Method>NVRAM</Method>
# <Status>up</Status>
# <Protocol>up</Protocol>
# </entry>
# <entry>
# <Interface>GigabitEthernet0/2</Interface>
# <IP-Address>unassigned</IP-Address>
# <OK>YES</OK>
# <Method>NVRAM</Method>
# <Status>administratively down</Status>
# <Protocol>down</Protocol>
# </entry>
# <entry>
# <Interface>GigabitEthernet0/3</Interface>
# <IP-Address>unassigned</IP-Address>
# <OK>YES</OK>
# <Method>NVRAM</Method>
# <Status>administratively down</Status>
# <Protocol>down</Protocol>
# </entry>
# <entry>

```

```

# <Interface>FastEthernet1/0</Interface>
# <IP-Address>unassigned</IP-Address>
# <OK>YES</OK>
# <Method>NVRAM</Method>
# <Status>administratively down</Status>
# <Protocol>down</Protocol>
# </entry>
# <entry>
# <Interface>FastEthernet1/1</Interface>
# <IP-Address>unassigned</IP-Address>
# <OK>YES</OK>
# <Method>NVRAM</Method>
# <Status>administratively down</Status>
# <Protocol>down</Protocol>
# </entry>
# </IPInterfaces>
# </ShowIpInterfaceBrief>
#
# Note: This example is the same as Example 4 with the exception that
# the new record procedure is called by the un-qualified tag name. The
# ability to specify the un-qualified tag names is simpler but only works
# if the un-qualified name is used once per Tcl program. In this example
# the unqualified new record tag name is "<entry>" which is a very
# common name in the Cisco spec file.
# Define a procedure to be called every time the <entry> tag
# is processed. Since this tag precedes each new output record, the data
# that is passed into this procedure contains the fields that have been
# requested via xml_pi_parse since the previous time this procedure was
# called.
proc <entry> {xml_line} {
    global num
    # The first time that this function is called there is no data and
    # xml_line will be null.
    if [string length $xml_line] {
        array set xml_result $xml_line
        incr num
        if ([string equal $xml_result(<Status>) "up"]) {
            set output [format "Example 5: Interface %2d %-30s %s" \
                $num $xml_result(<Interface>) $xml_result(<Status>)]
            puts $output
        }
        } else {
        puts "Example 5: Display All Interfaces That Are Up"
        }
    }
    set num 0
    # Invoke xml_pi_exec with the default built-in spec file for the
    # "show ip interface brief" command. After the command executes $result
    # contains the raw XML data.
    if [catch {xml_pi_exec $cli1(fd) "show ip interface brief"} result] {
        error $result $errorInfo
    } else {
        # Pass the raw XML data to the xml_pi_parse routine to extract fields
        # of interest:
        # we ask that only the <Interface> and <Status> fields be returned.
        # If an XML tag name is requested and a Tcl proc exists with that name,
        # the Tcl proc will be called every time that tag is encountered in the
        # output data. Specify the <entry> tag and define the proc
        # before executing the xml_pi_parse statement.
        array set xml_result [xml_pi_parse $cli1(fd) $result \
            "<entry> <Interface> <Status>"]
        # Display the data from the last record.
        incr num
        if ([string equal $xml_result(<Status>) "up"]) {

```

```

set output [format "Example 5: Interface %2d %-30s %s" \
$num $xml_result(<Interface>) $xml_result(<Status>)]
puts $output
}
}

```

### Sample XML-PI spec eemtest.odm ODM File:

```

###
show version
<?xml version='1.0' encoding='utf-8'?>
<ODMSpec>
<Command>
<Name>show version</Name>
</Command>
<OS>ios</OS>
<DataModel>
<Container name="ShowVersion">
<Property name="Version" distance = "1.0" length = "1" type = "IpAddress"/>
<Property name="Technical Support" distance = "1.0" length = "1" type = "IpAddress"/>
<Property name="Compiled" distance = "1.0" length = "3" type = "String"/>
<Property name="ROM" distance = "1.0" length = "7" type = "IpAddress"/>
<Property name="uptime" distance = "2" length = "8" type = "String"/>
<Property name="image" distance = "4" length = "1" type = "IpAddress"/>
<Property name="processor" distance = "-1" length = "1" type = "String"/>
<Property name="bytes of memory" distance = "-1" length = "1" type = "Port"/>
<Property name="CPU" distance = "2" length = "1" end-delimiter = "," type = "String"/>
<Property name="L2 Cache" distance = "-2" length = "1" end-delimiter = "," type =
"String"/>
<Property name="Gigabit Ethernet interfaces" distance = "-1" length = "1" type =
"Integer"/>
<Property name="bytes of NVRAM" distance = "-1" length = "1" type = "String"/>
<Property name="bytes of ATA PCMCIA card" distance = "-1" length = "1" type = "String"/>
<Property name="Sector size" distance = "1.0" length = "2" end-delimiter = ")" type =
"String"/>
<Property name="bytes of Flash internal SIMM" distance = "-1" length = "1" type =
"String"/>
<Property name="Configuration register" distance = "2" length = "1" type = "String"/>
</Container>
</DataModel>
</ODMSpec>

```

### Example sample.tcl Run:

```

router#config t
Enter configuration commands, one per line. End with CNTL/Z.
router(config)#event manager policy sample.tcl
router(config)#end
router#
Oct 10 20:21:26: %SYS-5-CONFIG_I: Configured from console by console
router#event manager run sample.tcl
Example 1: XML-PI support is present in this image
Example 2: Processor is NPE-G1 CPU is 700MHz
Example 3: Item 1 CISCO7206VXR "Cisco 7206VXR, 6-slot chassis"
Example 3: Item 2 NPE-G1 "Cisco 7200 Series Network Processing Engine NPE-G1"
Example 3: Item 3 MEM-NPE-G1-FLD128 "128MB Compact Flash Disk for NPE-G1"
Example 3: Item 4 PA-2FE-TX "Dual Port FastEthernet (RJ45)"
Example 3: Item 5 PWR-7200-AC "Cisco 7200 AC Power Supply"
Example 4: Display All Interfaces
Example 4: Interface 1 GigabitEthernet0/1 up
Example 4: Interface 2 GigabitEthernet0/2 administratively down
Example 4: Interface 3 GigabitEthernet0/3 administratively down
Example 4: Interface 4 FastEthernet1/0 administratively down
Example 4: Interface 5 FastEthernet1/1 administratively down

```

Example 4: Interface 6 SSLVPN-VIF0 up  
Example 5: Display All Interfaces That Are Up  
Example 5: Interface 1 GigabitEthernet0/1 up  
Example 5: Interface 6 SSLVPN-VIF0 up

## Tcl Context Library Command Extensions

All the Tcl context library command extensions belong to the `::cisco::eem` namespace.

### Exported Commands

- [context\\_retrieve](#), page 240
- [context\\_save](#), page 244

# context\_retrieve

Retrieves Tcl variable(s) identified by the given context name, and possibly the scalar variable name, the array variable name, and the array index. Retrieved information is automatically deleted.



## Note

Once saved information is retrieved, it is automatically deleted. If that information is needed by another policy, the policy that retrieves it (using the **context\_retrieve** command extension) should also save it again (using the **context\_save** command extension).

## Syntax

```
context_retrieve ctxt [var] [index_if_array]
```

## Arguments

ctxt	(Mandatory) Context name.
var	(Optional) Scalar variable name or array variable name. Defaults to a null string if this argument is not specified.
index_if_array	(Optional) The array index.



## Note

The `index_if_array` argument will be ignored when the `var` argument is a scalar variable.

If `var` is unspecified, retrieves the whole variable table saved in the context.

If `var` is specified and `index_if_array` is not specified, or if `index_if_array` is specified but `var` is a scalar variable, retrieves the value of `var`.

If `var` is specified, and `index_if_array` is specified, and `var` is an array variable, retrieves the value of the specified array element.

## Result String

Resets the Tcl global variables to the state that they were in when the save was performed.

## Set\_cerrno

- A string displaying `_cerrno`, `_cerr_sub_num`, `_cerr_sub_err`, `_cerr_posix_err`, `_cerr_str` due to `appl_reqinfo` error.
- Variable is not in the context.

## Sample Usage

The following examples show how to use the **context\_save** and **context\_retrieve** command extension functionality to save and retrieve data. The examples are shown in save and retrieve pairs.

### Example 1: Save

If `var` is unspecified or if a pattern is specified, saves multiple variables to the context.

```
::cisco::eem::event_register_none

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



```

set testvara 123
set testvarb 345
set testvarc 789
if {[catch {context_save TESTCTX "testvar*"} errmsg]} {
    action_syslog msg "context_save failed: $errmsg"
} else {
    action_syslog msg "context_save succeeded"
}

```

**Example 1: Retrieve**

If var is unspecified, retrieves multiple variables from the context.

```

::cisco::eem::event_register_none

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

if {[catch {foreach {var value} [context_retrieve TESTCTX] {set $var $value}} errmsg]} {
    action_syslog msg "context_retrieve failed: $errmsg"
} else {
    action_syslog msg "context_retrieve succeeded"
}
if {[info exists testvara]} {
    action_syslog msg "testvara exists and is $testvara"
} else {
    action_syslog msg "testvara does not exist"
}
if {[info exists testvarb]} {
    action_syslog msg "testvarb exists and is $testvarb"
} else {
    action_syslog msg "testvarb does not exist"
}
if {[info exists testvarc]} {
    action_syslog msg "testvarc exists and is $testvarc"
} else {
    action_syslog msg "testvarc does not exist"
}

```

**Example 2: Save**

If var is specified, saves the value of var.

```

::cisco::eem::event_register_none

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

set testvar 123
if {[catch {context_save TESTCTX testvar} errmsg]} {
    action_syslog msg "context_save failed: $errmsg"
} else {
    action_syslog msg "context_save succeeded"
}

```

**Example 2: Retrieve**

If var is specified and index\_if\_array is not specified, or if index\_if\_array is specified but var is a scalar variable, retrieves the value of var.

```

::cisco::eem::event_register_none

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

```

```

if {[catch {set testvar [context_retrieve TESTCTX testvar]} errmsg]} {
    action_syslog msg "context_retrieve failed: $errmsg"
} else {
    action_syslog msg "context_retrieve succeeded"
}
if {[info exists testvar]} {
    action_syslog msg "testvar exists and is $testvar"
} else {
    action_syslog msg "testvar does not exist"
}

```

**Example 3: Save**

If var is specified, saves the value of var even if it is an array.

```

::cisco::eem::event_register_none

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

array set testvar "testvar1 ok testvar2 not_ok"
if {[catch {context_save TESTCTX testvar} errmsg]} {
    action_syslog msg "context_save failed: $errmsg"
} else {
    action_syslog msg "context_save succeeded"
}

```

**Example 3: Retrieve**

If var is specified, and index\_if\_array is not specified, and var is an array variable, retrieves the entire array.

```

::cisco::eem::event_register_none

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

if {[catch {array set testvar [context_retrieve TESTCTX testvar]} errmsg]} {
    action_syslog msg "context_retrieve failed: $errmsg"
} else {
    action_syslog msg "context_retrieve succeeded"
}
if {[info exists testvar]} {
    action_syslog msg "testvar exists and is [array get testvar]"
} else {
    action_syslog msg "testvar does not exist"
}

```

**Example 4: Save**

If var is specified, saves the value of var even if it is an array.

```

::cisco::eem::event_register_none

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

array set testvar "testvar1 ok testvar2 not_ok"
if {[catch {context_save TESTCTX testvar} errmsg]} {
    action_syslog msg "context_save failed: $errmsg"
} else {
    action_syslog msg "context_save succeeded"
}

```

**Example 4: Retrieve**

If var is specified, and index\_if\_array is specified, and var is an array variable, retrieves the specified array element value.

```
::cisco::eem::event_register_none

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

if {[catch {set testvar [context_retrieve TESTCTX testvar testvar1]} errmsg]} {
    action_syslog msg "context_retrieve failed: $errmsg"
} else {
    action_syslog msg "context_retrieve succeeded"
}

if {[info exists testvar]} {
    action_syslog msg "testvar exists and is $testvar"
} else {
    action_syslog msg "testvar doesn't exist"
}
```

## context\_save

Saves Tcl variables that match a given pattern in current and global namespaces with the given context name as identification. Use this Tcl command extension to save information outside of a policy. Saved information can be retrieved by a different policy using the **context\_retrieve** command extension.



### Note

Once saved information is retrieved, it is automatically deleted. If that information is needed by another policy, the policy that retrieves it (using the **context\_retrieve** command extension) should also save it again (using the **context\_save** command extension).

### Syntax

```
context_save ctxt [pattern]
```

### Arguments

ctxt	(Mandatory) Context name.
pattern	(Optional) The glob-style pattern as used by the <b>string match</b> Tcl command. If this argument is not specified, the pattern defaults to the wildcard *.  There are three constructs used in glob patterns: <ul style="list-style-type: none"> <li>• * = all characters</li> <li>• ? = 1 character</li> <li>• [abc] = match one of a set of characters</li> </ul>

### Result String

None

### Set \_cerrno

A string displaying \_cerrno, \_cerr\_sub\_num, \_cerr\_sub\_err, \_cerr\_posix\_err, \_cerr\_str due to appl\_setinfo error.

### Sample Usage

For examples showing how to use the **context\_save** and **context\_retrieve** command extension functionality to save and retrieve data, see the “[Sample Usage](#)” section on page 240.

## Event Registration Tcl Command Extensions for EEM 3.2

- [event\\_register\\_identity](#), page 246
- [event\\_register\\_mat](#), page 249
- [event\\_register\\_neighbor\\_discovery](#), page 251

## event\_register\_identity

Registers for an identity event. Use this Tcl command extension to generate an event when AAA authentication or authorization is successful or failure or after normal user traffic on the port is allowed to flow.

### Syntax

```
event_register_identity [tag ?] interface ?
[aaa-attribute ?]
[authc {all | fail | success}]
[authz {all | fail | success}]
[authz-complete]
[mac-address ?]
[queue_priority {normal | low | high | last}]
[maxrun ?] [nice {0 | 1}]
```

### Arguments

tag	(Optional) String identifying a tag that can be used with the trigger Tcl command extension to support multiple event statements within a Tcl script.
interface	A regular expression pattern to match against interface names.
aaa-attribute	(Optional) A regular expression that can be used to filter events by specific AAA attributes.
authc	(Optional) Triggers events on successful, failed or both successful and failed authentication.
authz	(Optional) Triggers events on successful, failed or both successful and failed authorization.
authz-complete	(Optional) Triggers events once the device connected to the interface is fully authenticated, authorized and normal traffic has begun to flow on that interface.
mac-address	(Optional) A regular expression pattern that can be used to filter events by mac addresses of the remote device.
maxrun	(Optional) Maximum run time of the script (specified in SSSSSSSSS[.MMM] format, where SSSSSSSSS must be an integer representing seconds between 0 and 31536000, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the default 20-second run-time limit is used.

queue_priority	<p>(Optional) Priority level at which the script will be queued:</p> <ul style="list-style-type: none"> <li>• queue_priority low—Specifies that the script is to be queued at the lowest of the three priority levels.</li> <li>• queue_priority normal—Specifies that the script is to be queued at a priority level greater than low priority but less than high priority.</li> <li>• queue_priority high—Specifies that the script is to be queued at the highest of the three priority levels.</li> <li>• queue_priority last—Specifies that the script is to be queued at the lowest priority level.</li> </ul> <p>If more than one script is registered with the “queue_priority_last” argument set, these scripts will execute in the order in which the events are published.</p> <p>The queue_priority argument specifies the queuing priority, but not the execution priority, of the script being registered.</p> <p>If this argument is not specified, the default queuing priority is normal.</p>
nice	<p>(Optional) Policy run-time priority setting. When the nice argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.</p>

**Result String**

None

**Set\_cerrno**

No

**Event\_reqinfo For EEM\_EVENT\_IDENTITY**

```
"event_id %u event_type %u event_type_string {%s} event_pub_sec %u event_pub_msec %u
event_severity %u identity_stage %u identity_status %u interface %u identity_mac %u
identity_attribute {%s}"
```

Event Type	Description
event_id	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
event_type	Type of event.
event_type_string	An ASCII string that represents the name of the event for this event type.
event_pub_sec event_pub_msec	The time, in seconds and milliseconds, at which the event was published to the EEM.
event_severity	The severity of the event.
identity_stage	One among authentication, authorization or authorization-complete stages.
identity_status	Success or one of these failure types: fail_authc, fail_aaa_server, fail_no_response, fail_timeout, fail_authz. For authorization-complete it is always success.

<b>interface</b>	The interface for the event.
<b>identity_mac</b>	The MAC address of the remote device for the event.
<b>identity_&lt;attribute&gt;</b>	For each AAA attribute, a set a dynamic variable to the value corresponding to that AAA attribute in the attribute or value list.



# event\_register\_mat

Registers for a MAT event. Use this Tcl command extension to generate an event when a mac-address is learned in the mac-address-table.

## Syntax

```
event_register_identity [tag ?] interface ?
[mac-address ?]
[type {add | delete}]
[hold-down ?]
[maxrun ?]
```

## Arguments

tag	(Optional) String identifying a tag that can be used with the trigger Tcl command extension to support multiple event statements within a Tcl script.
interface	A regular expression pattern to match against interface names.
mac-address	Mandatory if the interface parameter is not specified. A regular expression pattern that can be used to filter events by mac addresses of the remote device.
type	(Optional) Filter based on a mac-address-table event type of add or delete. If not specified, the event type is not used in determining whether the event should be triggered.
hold-down	(Optional) When a mac-address-table event comes in, the hold-down timer can be set to make the event to wait between 1 and 4294967295 seconds before processing the policy. If not set then the policy is not delayed in being processed.
maxrun	(Optional) Maximum run time of the script (specified in SSSSSSSSS[.MMM] format, where SSSSSSSSS must be an integer representing seconds between 0 and 31536000, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the default 20-second run-time limit is used.

## Result String

None

## Set\_cerrno

No

## Event\_reqinfo For EEM\_EVENT\_MAT

```
"event_id %u event_type %u event_type_string {%s} event_pub_sec %u event_pub_msec %u
event_severity %u notification %u intf_name %u mac_address {%s}"
```

Event Type	Description
event_id	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
event_type	Type of event.
event_type_string	An ASCII string that represents the name of the event for this event type.

<b>event_pub_sec</b> <b>event_pub_msec</b>	The time, in seconds and milliseconds, at which the event was published to the EEM.
<b>event_severity</b>	The severity of the event.
<b>notification</b>	Notification type—add or delete.
<b>intf_name</b>	The interface name for the address table entry.
<b>mac_address</b>	The mac-address for the address table entry.

## event\_register\_neighbor\_discovery

Registers for a neighbor discover event. Use this Tcl command extension to generate an event when a Cisco Discovery Protocol (CDP) or Link Layer Discovery Protocol (LLDP) cache entry or a interface link status changes.

### Syntax

```
event_register_neighbor_discovery [tag ?] interface ?
[cdp {add | update | delete | all}]
[lldp {add | update | delete | all}]
[link-event]
[line-event]
[queue_priority {normal | low | high | last}]
[maxrun ?] [nice {0 | 1}]
```

### Arguments

tag	(Optional) String identifying a tag that can be used with the trigger Tcl command extension to support multiple event statements within a Tcl script.
interface	A regular expression pattern to match against interface names.
cdp	Trigger an event when a matching CDP event occurs. One of the following options should be specified. <ul style="list-style-type: none"> <li>• add—Trigger events only when a new CDP cache entry is created in the CDP table.</li> <li>• all—Trigger an event when a CDP cache entry is added or deleted from the CDP cache table and when a remote CDP device sends a keepalive to update the CDP cache entry.</li> <li>• delete—trigger events only when a CDP cache entry is deleted from the CDP table.</li> <li>• update—trigger an event when a CDP cache entry is added to the CDP table or when the remote CDP device sends a CDP keepalive to update the CDP cache entry.</li> </ul>
lldp	Trigger an event when a matching lldp event occurs. One of the following options should be specified. <ul style="list-style-type: none"> <li>• add—Trigger events only when a new cdp cache entry is created in the cdp table.</li> <li>• all—Trigger an event when a cdp cache entry is added or deleted from the cdp cache table and when a remote cdp device sends a keepalive to update the cdp cache entry.</li> <li>• delete—trigger events only when a cdp cache entry is deleted from the cdp table.</li> <li>• update—trigger an event when a cdp cache entry is added to the cdp table or when the remote cdp device sends a cdp keepalive to update the cdp cache entry.</li> </ul>
line-event	Trigger an event when the interface line protocol status changes.
link-event	Trigger an event when the interface link status changes.

queue_priority	<p>(Optional) Priority level at which the script will be queued:</p> <ul style="list-style-type: none"> <li>• queue_priority low—Specifies that the script is to be queued at the lowest of the three priority levels.</li> <li>• queue_priority normal—Specifies that the script is to be queued at a priority level greater than low priority but less than high priority.</li> <li>• queue_priority high—Specifies that the script is to be queued at the highest of the three priority levels.</li> <li>• queue_priority last—Specifies that the script is to be queued at the lowest priority level.</li> </ul> <p>If more than one script is registered with the “queue_priority_last” argument set, these scripts will execute in the order in which the events are published.</p> <p>The queue_priority argument specifies the queuing priority, but not the execution priority, of the script being registered.</p> <p>If this argument is not specified, the default queuing priority is normal.</p>
maxrun	<p>(Optional) Maximum run time of the script (specified in SSSSSSSSS[.MMM] format, where SSSSSSSSS must be an integer representing seconds between 0 and 31536000, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). If this argument is not specified, the default 20-second run-time limit is used.</p>
nice	<p>(Optional) Policy run-time priority setting. When the nice argument is set to 1, the policy is run at a run-time priority that is less than the default priority. The default value is 0.</p>

**Result String**

None

**Set\_cerrno**

No

**Event\_reqinfo For EEM\_EVENT\_NEIGHBOR\_DISCOVERY**

```
"event_id %u event_type %u event_type_string {%s} event_pub_sec %u event_pub_msec %u
event_severity %u nd_notification {%s}"
```

Event Type	Description
event_id	Unique number that indicates the ID for this published event. Multiple policies may be run for the same event, and each policy will have the same event_id.
event_type	Type of event.
event_type_string	An ASCII string that represents the name of the event for this event type.
event_pub_sec event_pub_msec	The time, in seconds and milliseconds, at which the event was published to the EEM.
event_severity	The severity of the event.
<b>Common Event_Reqinfo</b>	
nd_notification	The type of notification—cdp-add, cdp-update, cdp-delete, lldp-add, lldp-update, lldp-delete, link, line.

<b>nd_intf_linkstatus</b>	The current interface link status, up or down.
<b>nd_intf_linestatus</b>	The current interface line status, down, goingdown, init, testing, up, reset, admin down, deleted.
<b>nd_local_intf_name</b>	The local interface name for the event.
<b>nd_short_local_intf_name</b>	The short name of the local interface for the event.
<b>nd_port_id</b>	The port id as identified by either the cdp or lldp protocol. This is not set for link or line protocol events.
<b>CDP-specific Event_reqinfo</b>	
<b>nd_protocol</b>	Identifies which protocol triggered the event, for CDP it will always be set to cdp.
<b>nd_proto_notif</b>	Identifies which type of protocol event triggered the event, add, update or delete.
<b>nd_proto_new_entry</b>	If set to 1, the event was triggered because the cache entry is new, otherwise it will be set to 0.
<b>nd_cdp_entry_name</b>	The name of the cdp cache entry in the cdp table.
<b>nd_cdp_hold_time</b>	The time remaining until the cdp cache entry expires and is deleted from the cdp table. This time will be reset to some maximum by an update from the cdp neighbor. It is usually set to 0 for new entries.
<b>nd_cdp_mgmt_domain</b>	The CDP VTP management domain.
<b>nd_cdp_platform</b>	The platform name reported by the remote device.
<b>nd_cdp_version</b>	The version of code running on the remote device.
<b>nd_cdp_capabilities_string</b>	The contents of the CDP capabilities field in a string format: Router, Trans-Bridge, Source-Route-Bridge, Switch, Host, IGMP, Repeater, Phone, Remotely-Managed device, CVTA phone port, Two-port Mac Relay or any combination of these separated by commas.
<b>nd_cdp_capabilities_bits</b>	The CDP capabilities bits in a hexadecimal number preceded with 0x.
<b>nd_cdp_capabilities_bit_[0-31]</b>	A series of values that will be set to YES if that bit in the capabilities field is set or NO if it is not set.
<b>LLDP-specific Event_reqinfo</b>	
<b>nd_protocol</b>	Identifies which protocol triggered the event, for LLDP it will always be set to lldp.
<b>nd_proto_notif</b>	Identifies which type of protocol event triggered the event, add, update or delete.
<b>nd_proto_new_entry</b>	If set to 1, the event was triggered because the cache entry is new, otherwise it will be set to 0.
<b>nd_lldp_chassis_id</b>	The chassis id field from the LLDP cache entry.
<b>nd_lldp_system_name</b>	The system name from the LLDP cache entry.
<b>nd_lldp_system_description</b>	The system description field from the LLDP cache entry.
<b>nd_lldp_ttl</b>	The LLDP time to live field from the LLDP cache entry.
<b>nd_lldp_port_description</b>	The port description field from the LLDP cache entry.

<b>nd_ldap_system_capabilities_string</b>	The LLDP system capabilities field from the LLDP cache entry. Provided as a string that can contain O, P, B, W, R, T, C, S or any combination of these separated by commas.
<b>nd_ldap_enabled_capabilities_string</b>	The LLDP enabled system capabilities field from the LLDP cache entry. Provided as a string that can contain O, P, B, W, R, T, C, S or any combination of these separated by commas.
<b>nd_ldap_system_capabilities_bits</b>	The LLDP system capabilities bits field from the LLDP cache entry. Provided as a hexadecimal number preceded by 0x.
<b>nd_ldap_enabled_capabilities_bits</b>	The LLDP enabled capabilities bits field from the LLDP cache entry. Provided as a hexadecimal number preceded by 0x.
<b>nd_ldap_capabilities_bits</b>	The LLDP capabilities bits field from the LLDP cache entry. Provided as a hexadecimal number preceded by 0x.
<b>nd_ldap_capabilities_bit_[0-31]</b>	A series of values that will be set to YES if that bit in the capabilities field is set or NO if it is not set.

# Feature Information for Writing Embedded Event Manager Policies Using Tcl

Table 18 lists the features in this module and provides links to specific configuration information. Only features that were introduced or modified in Cisco IOS Releases 12.3(14)T, 12.2(25)S, 12.0(26)S, 12.2(18)SXF4, 12.2(28)SB, 12.2(33)SRA, 12.2(33)SXH, 12.2(33)SXI, 12.4(20)T, 12.4(22)T, 15.0(1)M, 12.2(33)SRE or a later release appear in the table.

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

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



## Note

Table 18 lists only the Cisco IOS software release that introduced support for a given feature in a given Cisco IOS software release train. Unless noted otherwise, subsequent releases of that Cisco IOS software release train also support that feature.

**Table 18** Feature Information for Writing Embedded Event Manager Policies Using Tcl

Feature Name	Releases	Feature Configuration Information
Embedded Event Manager 2.1	12.2(18)SXF5 12.2(28)SB 12.2(33)SRA 12.3(14)T	This document was introduced to support the ability to create policies using Tool Command Language (Tcl) that was introduced in the Embedded Event Manager 2.1 feature in Cisco IOS Release 12.3(14)T.
Embedded Event Manager 2.1 (Software Modularity)	12.2(18)SXF4 Cisco IOS Software Modularity images	<p>EEM 2.1 for Software Modularity images introduced the GOLD, system manager, and WDSysMon (Cisco IOS Software Modularity watchdog) event detectors and the ability to display Cisco IOS Software Modularity processes and process matrix. Six new sample policies were also introduced.</p> <p>The following sections provide information about this feature:</p> <ul style="list-style-type: none"> <li>• <a href="#">EEM Policy Tcl Command Extension Categories, page 3</a></li> <li>• <a href="#">Displaying Software Modularity Process Reliability Metrics Using EEM, page 18</a></li> <li>• <a href="#">Modifying the Sample EEM Policies, page 20</a></li> </ul> <p>The following commands were introduced by this feature: <b>event gold</b>, <b>event process</b>, <b>show event manager metric process</b>.</p> <p><b>Note</b> EEM 2.1 for Software Modularity images also supports the resource and RF event detectors introduced in EEM 2.2, but it does not support the enhanced object tracking event detector or the actions to read and set tracked objects.</p>

Table 18 Feature Information for Writing Embedded Event Manager Policies Using Tcl (continued)

Feature Name	Releases	Feature Configuration Information
Embedded Event Manager 2.2	12.2(31)SB3 12.2(33)SRB 12.4(2)T	<p>EEM 2.2 introduced the enhanced object tracking, resource, and RF event detectors. The actions of reading and setting the state of a tracked object were also introduced.</p> <p>The following sections provide information about this, feature:</p> <ul style="list-style-type: none"> <li>• <a href="#">EEM Policy Tcl Command Extension Categories, page 3</a></li> <li>• <a href="#">Modifying the Sample EEM Policies, page 20</a></li> </ul> <p>The following commands were introduced or modified by this feature: <b>action track read</b>, <b>action track set</b>, <b>default-state</b>, <b>event resource</b>, <b>event rf</b>, <b>event track</b>, <b>show track</b>, <b>track stub-object</b>.</p>
SNMP event detector delta environment variable <sup>1</sup>	12.4(11)T	<p>A new SNMP event detector environment variable, <code>delta_val</code>, was introduced.</p>
Embedded Event Manager 2.3	12.2(33)SB 12.2(33)SXH	<p>EEM 2.3 introduced some new features relative to the Generic Online Diagnostics (GOLD) Event Detector on the Cisco Catalyst 6500 Series switches.</p> <p>The <b>event gold</b> command was enhanced in addition to the Tcl keywords—<b>action-notify</b>, <b>testing-type</b>, <b>test-name</b>, <b>test-id</b>, <b>consecutive-failure</b>, <b>platform-action</b>, and <b>maxrun</b>—for improved reaction to GOLD test failures and conditions.</p> <p>The following updates were made to the <b>event gold</b> command:</p> <ul style="list-style-type: none"> <li>• Optional arguments were added to the <b>event_register_gold</b> EEM Event Registration Tcl command extension to support additional event configuration options.</li> <li>• Event types were added under the <b>event_reqinfo</b> EEM Event Information Tcl command extension to provide access to platform-wide and test-specific GOLD EEM Tcl policy information for a detected event.</li> </ul>



**Table 18**      *Feature Information for Writing Embedded Event Manager Policies Using Tcl (continued)*

Feature Name	Releases	Feature Configuration Information
Embedded Event Manager 2.4	12.2(33)SRE 12.2(33)SXI 12.4(20)T	<p>EEM 2.4 is supported in Cisco IOS Release 12.4(20)T and later releases, and introduced several new features.</p> <p>The following sections were updated:</p> <ul style="list-style-type: none"> <li>• <a href="#">EEM Policy Tcl Command Extension Categories, page 3</a></li> <li>• <a href="#">Modifying the Sample EEM Policies, page 20</a></li> </ul> <p>The following commands were introduced by this feature:</p> <p><b>attribute (EEM), correlate, event manager detector rpc, event manager directory user repository, event manager update user policy, event manager scheduler clear, event manager update user policy, event owner, event rpc, event snmp-notification, show event manager detector, show event manager version, trigger (EEM).</b></p>

Table 18 Feature Information for Writing Embedded Event Manager Policies Using Tcl (continued)

Feature Name	Releases	Feature Configuration Information
Embedded Event Manger 3.0	12.2(33)SRE 12.4(22)T	<p>EEM 3.0 is supported in Cisco IOS Release 12.4(22)T and later releases, and introduced several new features.</p> <p>The following sections provide information about this feature:</p> <ul style="list-style-type: none"> <li>• <a href="#">EEM Policy Tcl Command Extension Categories, page 3</a></li> <li>• <a href="#">Modifying the Sample EEM Policies, page 20</a></li> </ul> <p>The following commands were introduced or modified by this feature:</p> <p><b>action add, action append, action break, action comment, action context retrieve, action context save, action continue, action decrement, action divide, action else, action elseif, action end, action exit, action foreach, action gets, action if, action if goto, action increment, action info type interface-names, action info type snmp getid, action info type snmp inform, action info type snmp oid, action info type snmp trap, action info type snmp var, action multiply, action puts, action regexp, action set (EEM), action string compare, action string equal, action string first, action string index, action string last, action string length, action string match, action string range, action string replace, action string tolower, action string toupper, action string trim, action string trimleft, action string trimright, action subtract, action while, event cli, event ipsla, event manager detector routing, event manager scheduler, event manager scheduler clear, event manager scheduler hold, event manager scheduler modify, event manager scheduler release, event nf, event routing, show event manager policy active, show event manager policy pending, and show event manager scheduler.</b></p>
Embedded Event Manager 3.1	15.0(1)M	<p>EEM 3.1 is supported in Cisco IOS Release 15.0(1)M and later releases, and introduced several new features.</p> <p>The following sections provide information about this feature:</p> <ul style="list-style-type: none"> <li>• <a href="#">EEM Policy Tcl Command Extension Categories, page 3</a></li> <li>• <a href="#">Modifying the Sample EEM Policies, page 20</a></li> </ul> <p>The following commands were introduced or modified by this feature:</p> <p><b>action syslog, description, event manager applet, event manager policy, event snmp-notification, event snmp-object, show event manager policy registered, and show event manager policy available.</b></p>

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

Cisco and the Cisco Logo are trademarks of Cisco Systems, Inc. and/or its affiliates in the U.S. and other countries. A listing of Cisco's trademarks can be found at [www.cisco.com/go/trademarks](http://www.cisco.com/go/trademarks). Third party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1005R)

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

© 2005–2010 Cisco Systems, Inc. All rights reserved.

