Configuring Embedded Event Manager

This chapter contains the following sections:

About Embedded Event Manager

The ability to detect and handle critical events in the Cisco NX-OS system is important for high availability. The Embedded Event Manager (EEM) provides a central, policy-driven framework to detect and handle events in the system by monitoring events that occur on your device and taking action to recover or troubleshoot these events, based on your configuration..

EEM consists of three major components:

Event statements

Events to monitor from another Cisco NX-OS component that may require some action, workaround, or notification.

Action statements

An action that EEM can take, such as sending an e-mail or disabling an interface, to recover from an event.

Policies

An event paired with one or more actions to troubleshoot or recover from the event.

Without EEM, each individual component is responsible for detecting and handling its own events. For example, if a port flaps frequently, the policy of "putting it into errDisable state" is built into ETHPM.

Embedded Event Manager Policies

An EEM policy consists of an event statement and one or more action statements. The event statement defines the event to look for as well as the filtering characteristics for the event. The action statement defines the action EEM takes when the event occurs.

For example, you can configure an EEM policy to identify when a card is removed from the device and log the details related to the card removal. By setting up an event statement that tells the system to look for all instances of card removal and an then with an action statement that tells the system to log the details.

You can configure EEM policies using the command line interface (CLI) or a VSH script.

EEM gives you a device-wide view of policy management. Once EEM policies are configured, the corresponding actions are triggered. All actions (system or user-configured) for triggered events are tracked and maintained by the system.

Preconfigured System Policies

Cisco NX-OS has a number of preconfigured system policies. These system policies define many common events and actions for the device. System policy names begin with two underscore characters (__).

Some system policies can be overridden. In these cases, you can configure overrides for either the event or the action. The overrides that you configure take the place of the system policy.

> **Note** Override policies must include an event statement. Override policies without event statements override all possible events for the system policy.

To view the preconfigured system polices and determine which polices you can override, use the **show event manager system-policy** command.

User-Created Policies

User-created policies allow you to customize EEM policies for your network. If a user policy is created for an event, actions in the policy are triggered only after EEM triggers the system policy actions related to the same event.

Log Files

The log file that contains data that is related to EEM policy matches is maintained in the event_archive_1 log file located in the /log/event_archive_1 directory.

Event Statements

Any device activity for which some action, such as a workaround or notification, is taken is considered an event by EEM. In many cases, events are related to faults in the device, such as when an interface or a fan malfunctions.

Event statements specify which event or events triggers a policy to run.

Tip You can configure EEM to trigger an EEM policy that is based on a combination of events by creating and differentiating multiple EEM events in the policy and then defining a combination of events to trigger a custom action.

EEM defines event filters so that only critical events or multiple occurrences of an event within a specified time period trigger an associated action.

Some commands or internal events trigger other commands internally. These commands are not visible, but will still match the event specification that triggers an action. You cannot prevent these commands from triggering an action, but you can check which event triggered an action.

Supported Events

EEM supports the following events in event statements:

- Counter events
- Fan absent events
- Fan bad events
- Memory thresholds events
- Events being used in overridden system policies.
- SNMP notification events
- Syslog events
- System manager events
- Temperature events
- Track events

Action Statements

Action statements describe the action that is triggered by a policy when an event occurs. Each policy can have multiple action statements. If no action is associated with a policy, EEM still observes events but takes no actions.

In order for triggered events to process default actions, you must configure the EEM policy to allow the default action. For example, if you match a CLI command in a match statement, you must add the event-default action statement to the EEM policy or EEM does not allow the command to execute.

Note When configuring action statements within your user policy or overriding policy, it is important that you confirm that action statements do not negate each other or adversely affect the associated system policy.

Supported Actions

EEM supports the following actions in action statements:

- Execute any CLI commands

- Update a counter

- Reload the device

- Generate a syslog message

- Generate an SNMP notification

- Use the default action for the system policy

VSH Script Policies

You can write policies in a VSH script, by using a text editor. Policies that are written using a VSH script have an event statement and action statement(s) just as other policies, and these policies can either augment or override system policies.

After you define your VSH script policy, copy it to the device and activate it.

Prerequisites for Embedded Event Manager

You must have network-admin privileges to configure EEM.

Guidelines and Limitations for Embedded Event Manager

When you plan your EEM configuration, consider the following:

- The maximum number of configurable EEM policies is 500.

- Action statements within your user policy or overriding policy should not negate each other or adversely affect the associated system policy.

- To allow a triggered event to process any default actions, you must configure the EEM policy to allow the default action. For example, if you match a command in a match statement, you must add the event-default action statement to the EEM policy or EEM does not allow the command to execute.

- The following guidelines apply to Event Log Auto-Collection and Backup:

  - By default, enabled log collection on a switch provides between 15 minutes to several hours of event logs depending on size, scale and component activity.

  - To be able to collect relevant logs that span a longer period, only enable event log retention for the specific services/features you need. See "Enabling Extended Log File Retention For a Single Service". You can also export the internal event logs. See "External Log File Storage".

  - When troubleshooting, it is good practice to manually collect a snapshot of internal event logs in real time. See "Generating a Local Copy of Recent Log Files".

- An override policy that consists of an event statement and no action statement triggers no action and no notification of failures.

- An override policy without an event statement overrides all possible events in the system policy.

- In regular command expressions: all keywords must be expanded, and only the asterisk (*) symbol can be used for replace the arguments.

- EEM event correlation supports up to four event statements in a single policy. The event types can be the same or different, but only these event types are supported: cli, counter, snmp, syslog, and track.

- When more than one event statement is included in an EEM policy, each event statement must have a **tag** keyword with a unique tag argument.

- EEM event correlation does not override the system default policies.

- Default action execution is not supported for policies that are configured with tagged events.

- If your event specification matches a CLI pattern, you can use SSH-style wild card characters.

  For example, if you want to match all show commands, enter the **show *** command. Entering the **show . *** command does not work.

- If your event specification is a regular expression for a matching syslog message, you can use a proper regular expression.

  For example, if you want to detect ADMIN_DOWN events on any port where a syslog is generated, use **.ADMIN_DOWN.**. Entering the **ADMIN_DOWN** command does not work.

- In the event specification for a syslog, the regex does not match any syslog message that is generated as an action of an EEM policy.

- If an EEM event matches a **show** command in the CLI and you want the output for that **show** command to display on the screen (and to not be blocked by the EEM policy), you must specify the **event-default** command for the first action for the EEM policy.

- Cisco Nexus 3500 Series switches do not support Embedded Event Manager in Cisco NX-OS Release 7.0(3)I7(2) and the previous releases.

Default Settings for Embedded Event Manager

Table 1: Default EEM Parameters

Parameters	Default
System Policies	Active

Defining an Environment Variable

Defining an environment variable is an optional step but is useful for configuring common values for repeated use in multiple policies.

SUMMARY STEPS

1. **configure terminal**
2. **event manager environment** *variable-name variable-value*
3. (Optional) **show event manager environment** {*variable-name* | **all**}

    4. (Optional) **copy running-config startup-config**

DETAILED STEPS

	Command or Action	Purpose

Step 1	**configure terminal**
	
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**event manager environment** *variable-name variable-value*
	
	Example:
	`switch(config) # event manager`
	`environment emailto "admin@anyplace.com"`
		Creates an environment variable for EEM.
		
		The *variable-name* can be any case-sensitive, alphanumeric string up to 29 characters.
		
		The *variable-value* can be any quoted case-sensitive, alphanumeric string up to 39 characters.

Step 3	(Optional) **show event manager environment** {*variable-name* | **all**}
	
	Example:
	`switch(config) # show event manager`
	`environment all`
		Displays information about the configured environment variables.

Step 4	(Optional) **copy running-config startup-config** **Example:** `switch(config)# copy running-config startup-config`	Saves the change persistently through reboots and restarts by copying the running configuration to the startup configuration.

What to do next

Configure a User Policy.

Defining a User Policy Using the CLI

SUMMARY STEPS

    1. **configure terminal**
    2. **event manager applet** *applet-name*
    3. (Optional) **description** *policy-description*
    4. **event** *event-statement*
    5. (Optional) **tag** *tag* {**and** \| **andnot** \| **or**} *tag* [**and** \| **andnot** \| **or** {*tag*}] {**happens** *occurs* **in** *seconds*}
    6. **action** *number*[.*number2*] *action-statement*
    7. (Optional) **show event manager policy-state** *name* [**module** *module-id*]
    8. (Optional) **copy running-config startup-config**

DETAILED STEPS

	Command or Action	**Purpose**

Step 1	**configure terminal**
	
	Example:
	
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**event manager applet** *applet-name*
	
	Example:
	
	`switch(config)# event manager applet`
	`monitorShutdown`
	`switch(config-applet)#`
		Registers the applet with EEM and enters applet configuration mode.
		
		The applet-name can be any case-sensitive, alphanumeric string up to 29 characters.

Step 3	(Optional) **description** *policy-description*
	
	Example:
	
	`switch(config-applet)# description "Monitors`
	`interface shutdown."`
		Configures a descriptive string for the policy.
		
		The string can be any alphanumeric string up to 80 characters. Enclose the string in quotation marks.

Step 4	**event** *event-statement* **Example:** `switch(config-applet)# event cli match "shutdown"`	Configures the event statement for the policy.

Step 5	(Optional) **tag** *tag* {**and** | **andnot** | **or**} *tag* [**and** | **andnot** | **or** {*tag*}] {**happens** *occurs* **in** *seconds*}
	
	Example:
	
	`switch(config-applet)# tag one or two happens 1 in`
	` 10000`
		Correlates multiple events in the policy.
		
		The range for the *occurs* argument is from 1 to 4294967295.
		
		The range for the *seconds* argument is from 0 to 4294967295 seconds.

Step 6	**action** *number*[.*number2*] *action-statement*
	
	Example:
	
	`switch(config-applet)# action 1.0 cli show`
	`interface e 3/1`
		Configures an action statement for the policy. Repeat this step for multiple action statements.

Step 7	(Optional) **show event manager policy-state** *name* [**module** *module-id*]
	
	Example:
	
	`switch(config-applet)# show event manager`
	`policy-state monitorShutdown`
		Displays information about the status of the configured policy.

Step 8	(Optional) **copy running-config startup-config** **Example:** `switch(config)# copy running-config startup-config`	Saves the change persistently through reboots and restarts by copying the running configuration to the startup configuration.

What to do next

Configure event statements and action statements.

Configuring Event Statements

Use one of the following commands in EEM configuration mode (config-applet) to configure an event statement:

Before you begin

Define a user policy.

SUMMARY STEPS

1. **event cli** [**tag** *tag*] **match** *expression* [**count** *repeats* | **time** *seconds*
2. **event counter** [**tag** *tag*] **name** *counter* **entry-val** *entry* **entry-op** {**eq** | **ge** | **gt** | **le** | **lt** | **ne**} {**exit-val** *exit* **exit-op** {**eq** | **ge** | **gt** | **le** | **lt** | **ne**}
3. **event fanabsent** [**fan** *number*] **time** *seconds*
4. **event fanbad** [**fan** *number*] **time** *seconds*
5. **event memory** {**critical** | **minor** | **severe**}
6. **event policy-default count** *repeats* [**time** *seconds*]
7. **event snmp** [**tag** *tag*] **oid** *oid* **get-type** {**exact** | **next**} **entry-op** {**eq** | **ge** | **gt** | **le** | **lt** | **ne**} **entry-val** *entry* [**exit-comb** {**and** | **or**}]**exit-op** {**eq** | **ge** | **gt** | **le** | **lt** | **ne**} **exit-val** *exit* **exit-time** *time* **polling-interval** *interval*
8. **event sysmgr memory** [**module** *module-num*] **major** *major-percent* **minor** *minor-percent* **clear** *clear-percent*
9. **event temperature** [**module** *slot*] [**sensor** *number*] **threshold** {**any** | **down** | **up**}
10. **event track** [**tag** *tag*] *object-number* **state** {**any** | **down** | **up**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**event cli** [**tag** *tag*] **match** *expression* [**count** *repeats*	**time** *seconds* **Example:** `switch(config-applet) # event cli match "shutdown"`	Triggers an event if you enter a command that matches the regular expression. The **tag** *tag* keyword-argument pair identifies this specific event when multiple events are included in the policy. The *repeats* range is from 1 to 65000. The *time* range is from 0 to 4294967295, where 0 indicates no time limit.
Step 2	**event counter** [**tag** *tag*] **name** *counter* **entry-val** *entry* **entry-op** {**eq**	**ge**	**gt**	**le**	**lt**	**ne**} {**exit-val** *exit* **exit-op** {**eq**	**ge**	**gt**	**le**	**lt**	**ne**} **Example:** `switch(config-applet) # event counter name mycounter entry-val 20 gt`	Triggers an event if the counter crosses the entry threshold based on the entry operation. The event resets immediately. Optionally, you can configure the event to reset after the counter passes the exit threshold. The **tag** *tag* keyword-argument pair identifies this specific event when multiple events are included in the policy. The *counter* name can be any case-sensitive, alphanumeric string up to 28 characters.

	Command or Action	Purpose
		The *entry* and *exit* value ranges are from 0 to 2147483647.
Step 3	**event fanabsent** [**fan** *number*] **time** *seconds* **Example:** `switch(config-applet) # event fanabsent time 300`	Triggers an event if a fan is removed from the device for more than the configured time, in seconds. The *number* range is is from 1 to 1 and is module-dependent. The *seconds* range is from 10 to 64000.
Step 4	**event fanbad** [**fan** *number*] **time** *seconds* **Example:** `switch(config-applet) # event fanbad time 3000`	Triggers an event if a fan fails for more than the configured time, in seconds. The *number* range is module-dependent. The *seconds* range is from 10 to 64000.
Step 5	**event memory** {**critical** | **minor** | **severe**} **Example:** `switch(config-applet) # event memory critical`	Triggers an event if a memory threshold is crossed.
Step 6	**event policy-default count** *repeats* [**time** *seconds*] **Example:** `switch(config-applet) # event policy-default count 3`	Uses the event configured in the system policy. Use this option for overriding policies. The *repeats* range is from 1 to 65000. The *seconds* range is from 0 to 4294967295, where 0 indicates no time limit.
Step 7	**event snmp** [**tag** *tag*] **oid** *oid* **get-type** {**exact** | **next**} **entry-op** {**eq** | **ge** | **gt** | **le** | **lt** | **ne**} **entry-val** *entry* [**exit-comb** {**and** | **or**}]**exit-op** {**eq** | **ge** | **gt** | **le** | **lt** | **ne**} **exit-val** *exit* **exit-time** *time* **polling-interval** *interval* **Example:** `switch(config-applet) # event snmp oid 1.3.6.1.2.1.31.1.1.1.6 get-type next entry-op lt 300 entry-val 0 exit-op eq 400 exit-time 30 polling-interval 300`	Triggers an event if the SNMP OID crosses the entry threshold based on the entry operation. The event resets immediately, or optionally you can configure the event to reset after the counter passes the exit threshold. The OID is in dotted decimal notation. The **tag** *tag* keyword-argument pair identifies this specific event when multiple events are included in the policy. The *entry* and *exit* value ranges are from 0 to 18446744073709551615. The *time*, in seconds, is from 0 to 2147483647. The *interval*, in seconds, is from 0 to 2147483647.
Step 8	**event sysmgr memory** [**module** *module-num*] **major** *major-percent* **minor** *minor-percent* **clear** *clear-percent* **Example:** `switch(config-applet) # event sysmgr memory minor 80`	Triggers an event if the specified system manager memory threshold is exceeded. The *percent* range is from 1 to 99.
Step 9	**event temperature** [**module** *slot*] [**sensor** *number*] **threshold** {**any** | **down** | **up**} **Example:**	Triggers an event if the temperature sensor exceeds the configured threshold. The *sensor* range is from 1 to 18.

	Command or Action	Purpose
	`switch(config-applet) # event temperature module 2 threshold any`	
Step 10	**event track** [**tag** *tag*] *object-number* **state** {**any** | **down** | **up**	Triggers an event if the tracked object is in the configured state.
	Example:	The **tag** *tag* keyword-argument pair identifies this specific event when multiple events are included in the policy.
	`switch(config-applet) # event track 1 state down`	The *object-number* range is from 1 to 500.

What to do next

Configure action statements.

If you have already configured action statements or choose not to, complete any of the optional tasks:

- Define a policy using a VSH script. Then, register and activate a VSH script policy.

- Configure memory thresholds

- Configure the syslog as an EEM publisher.

- Verify your EEM configuration.

Configuring Action Statements

You can configure an action by using one of the following commands in EEM configuration mode (config-applet):

> **Note** If you want to allow a triggered event to process any default actions, you must configure the EEM policy to allow the default action. For example, if you match a command in a match statement, you must add the event-default action statement to the EEM policy or EEM does not allow the command to execute. You can use the **terminal event-manager bypass** command to allow all EEM policies with matches to execute the command.

Before you begin

Define a user policy.

SUMMARY STEPS

1. **action** *number*[*.number2*] **cli** *command1*[*command2.*] [**local**]
2. **action** *number*[*.number2*] **counter name** *counter* **value** *val* **op** {**dec** \| **inc** \| **nop** \| **set**}
3. **action** *number*[*.number2*] **event-default**
4. **action** *number*[*.number2*] **policy-default**
5. **action** *number*[*.number2*] **reload** [**module** *slot* [**-** *slot*]]
6. **action** *number*[*.number2*] **snmp-trap** [**intdata1** *integer-data1*] [**intdata2** *integer-data2*] [**strdata** *string-data*]

7. **action** *number*[.*number2*] **syslog** [**priority** *prio-val*] **msg** *error-message*

DETAILED STEPS

	Command or Action	**Purpose**
Step 1	**action** *number*[.*number2*] **cli** *command1*[*command2.*] [**local**] **Example:** `switch(config-applet) # action 1.0 cli "show interface e 3/1"`	Runs the configured commands. You can optionally run the commands on the module where the event occurred. The action label is in the format number1.number2. The *number* can be any number from 1 to 16 digits. The range for *number2* is from 0 to 9.
Step 2	**action** *number*[.*number2*] **counter name** *counter* **value** *val* **op** {**dec**	**inc**	**nop**	**set**} **Example:** `switch(config-applet) # action 2.0 counter name mycounter value 20 op inc`	Modifies the counter by the configured value and operation. The action label is in the format number1.number2. The *number* can be any number from 1 to 16 digits. The range for *number2* is from 0 to 9. The *counter* can be any case-sensitive, alphanumeric string up to 28 characters. The *val* can be an integer from 0 to 2147483647 or a substituted parameter.
Step 3	**action** *number*[.*number2*] **event-default** **Example:** `switch(config-applet) # action 1.0 event-default`	Completes the default action for the associated event. The action label is in the format number1.number2. The *number* can be any number from 1 to 16 digits. The range for *number2* is from 0 to 9.
Step 4	**action** *number*[.*number2*] **policy-default** **Example:** `switch(config-applet) # action 1.0 policy-default`	Completes the default action for the policy that you are overriding. The action label is in the format number1.number2. The *number* can be any number from 1 to 16 digits. The range for *number2* is from 0 to 9.
Step 5	**action** *number*[.*number2*] **reload** [**module** *slot* [**-** *slot*]] **Example:** `switch(config-applet) # action 1.0 reload module 3-5`	Forces one or more modules to the entire system to reload. The action label is in the format number1.number2. The *number* can be any number from 1 to 16 digits. The range for *number2* is from 0 to 9.
Step 6	**action** *number*[.*number2*] **snmp-trap** [**intdata1** *integer-data1*] [**intdata2** *integer-data2*] [**strdata** *string-data*] **Example:** `switch(config-applet) # action 1.0 snmp-trap strdata "temperature problem"`	Sends an SNMP trap with the configured data. The action label is in the format number1.number2. The *number* can be any number from 1 to 16 digits. The range for *number2* is from 0 to 9. The *data* elements can be any number up to 80 digits.

	Command or Action	Purpose
		The *string* can be any alphanumeric string up to 80 characters.
Step 7	**action** *number*[.*number2*] **syslog** [**priority** *prio-val*] **msg** *error-message* **Example:** `switch(config-applet) # action 1.0 syslog priority notifications msg "cpu high"`	Sends a customized syslog message at the configured priority. The action label is in the format number1.number2. The *number* can be any number from 1 to 16 digits. The range for *number2* is from 0 to 9. The *error-message* can be any quoted alphanumeric string up to 80 characters.

What to do next

Configure event statements.

If you have already configured event statements or choose not to, complete any of the optional tasks:

- Define a policy using a VSH script. Then, register and activate a VSH script policy.

- Configure memory thresholds

- Configure the syslog as an EEM publisher.

- Verify your EEM configuration.

Defining a Policy Using a VSH Script

This is an optional task. Complete the following steps if you are using a VSH script to write EEM policies:

SUMMARY STEPS

1. In a text editor, list the commands that define the policy.
2. Name the text file and save it.
3. Copy the file to the following system directory: bootflash://eem/user_script_policies

DETAILED STEPS

Step 1	In a text editor, list the commands that define the policy.
Step 2	Name the text file and save it.
Step 3	Copy the file to the following system directory: bootflash://eem/user_script_policies

What to do next

Register and activate a VSH script policy.

Registering and Activating a VSH Script Policy

This is an optional task. Complete the following steps if you are using a VSH script to write EEM policies.

Before you begin

Define a policy using a VSH script and copy the file to the system directory.

SUMMARY STEPS

1. **configure terminal**
2. **event manager policy** *policy-script*
3. (Optional) **event manager policy internal** *name*
4. (Optional) **copy running-config startup-config**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** ``` switch# configure terminal switch(config)# ```	Enters global configuration mode.
Step 2	**event manager policy** *policy-script* **Example:** ``` switch(config)# event manager policy moduleScript ```	Registers and activates an EEM script policy. The *policy-script* can be any case-sensitive, alphanumeric string up to 29 characters.
Step 3	(Optional) **event manager policy internal** *name* **Example:** ``` switch(config)# event manager policy internal moduleScript ```	Registers and activates an EEM script policy. The *policy-script* can be any case-sensitive alphanumeric string up to 29 characters.
Step 4	(Optional) **copy running-config startup-config** **Example:** ``` switch(config)# copy running-config startup-config ```	Saves the change persistently through reboots and restarts by copying the running configuration to the startup configuration.

What to do next

Complete any of the following, depending on your system requirements:

- Configure memory thresholds.
- Configure the syslog as an EEM publisher.
- Verify your EEM configuration.

Overriding a System Policy

SUMMARY STEPS

1. **configure terminal**
2. (Optional) **show event manager policy-state** *system-policy*
3. **event manager applet** *applet-name* **override** *system-policy*
4. **description** *policy-description*
5. **event** *event-statement*
6. **section** *number action-statement*
7. (Optional) **show event manager policy-state** *name*
8. (Optional) **copy running-config startup-config**

DETAILED STEPS

	Command or Action	Purpose

Step 1	**configure terminal**
	
	Example:
	
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	(Optional) **show event manager policy-state** *system-policy*
	
	Example:
	
	`switch(config-applet)# show event`
	`manager policy-state __ethpm_link_flap`
	`Policy __ethpm_link_flap`
	`   Cfg count : 5`
	`   Cfg time interval : 10.000000`
	`(seconds)`
	`   Hash default, Count 0`
		Displays information about the system policy that you want to override, including thresholds. Use the **show event manager system-policy** command to find the system policy names.

Step 3	**event manager applet** *applet-name* **override** *system-policy*
	
	Example:
	
	`switch(config-applet)# event manager applet`
	`ethport override __ethpm_link_flap`
	`switch(config-applet)#`
		Overrides a system policy and enters applet configuration mode.
		
		The *applet-name* can be any case-sensitive, alphanumeric string up to 80 characters.
		
		The *system-policy* must be one of the system policies.

Step 4	**description** *policy-description*
	
	Example:
	
	`switch(config-applet)# description`
	`"Overrides link flap policy"`
		Configures a descriptive string for the policy.
		
		The *policy-description* can be any case-sensitive, alphanumeric string up to 80 characters, but it must be enclosed in quotation marks.

Step 5	**event** *event-statement*
	
	Example:
	
	`switch(config-applet)# event`
	`policy-default count 2 time 1000`
		Configures the event statement for the policy.

	Command or Action	Purpose

Step 6	**section** *number action-statement*
	
	Example:
	
	`switch(config-applet)# action 1.0 syslog`
	`priority warnings msg "Link is`
	`flapping."`
		Configures an action statement for the policy. For multiple action statements, repeat this step.

Step 7	(Optional) **show event manager policy-state** *name*
	
	Example:
	
	`switch(config-applet)# show event`
	`manager policy-state ethport`
		Displays information about the configured policy.

Step 8	(Optional) **copy running-config startup-config** **Example:** `switch(config)# copy running-config startup-config`	Saves the change persistently through reboots and restarts by copying the running configuration to the startup configuration.

Configuring Syslog as an EEM Publisher

Configuring syslog as an EEM publisher allows you to monitor syslog messages from the switch.

✎

Note   The maximum number of searchable strings to monitor syslog messages is 10.

Before you begin

- Confirm that EEM is available for registration by the syslog.

- Confirm that the syslog daemon is configured and executed.

SUMMARY STEPS

1. **configure terminal**
2. **event manager applet** *applet-name*
3. **event syslog** [**tag** *tag*] {**occurs** *number* | **period** *seconds* | **pattern** *msg-text* | **priority** *priority*}
4. (Optional) **copy running-config startup-config**

DETAILED STEPS

	Command or Action	Purpose

Step 1	**configure terminal**
	
	Example:
	
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

	Command or Action	Purpose

Step 2	**event manager applet** *applet-name*
	
	Example:
	`switch(config)# event manager applet abc`
	`switch (config-appliet)#`
		Registers an applet with EEM and enters applet configuration mode.

Step 3	**event syslog** [**tag** *tag*] {**occurs** *number* | **period** *seconds* | **pattern** *msg-text* | **priority** *priority*} **Example:** `switch(config-applet)# event syslog occurs 10`	Registers an applet with EEM and enters applet configuration mode.
Step 4	(Optional) **copy running-config startup-config** **Example:** `switch(config)# copy running-config startup-config`	Saves the change persistently through reboots and restarts by copying the running configuration to the startup configuration.

What to do next

Verify your EEM configuration.