# Configuring the Embedded Event Manager

This chapter describes how to configure the Embedded Event Manager (EEM) to detect and handle critical events on Cisco NX-OS devices.

This chapter includes the following sections:



## Finding Feature Information

Your software release might not support all the features documented in this module. For the latest caveats and feature information, see the Bug Search Tool at https://tools.cisco.com/bugsearch/ and the release notes for your 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 "New and Changed Information"chapter or the Feature History table in this chapter.

## About EEM

EEM monitors events that occur on your device and takes 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.

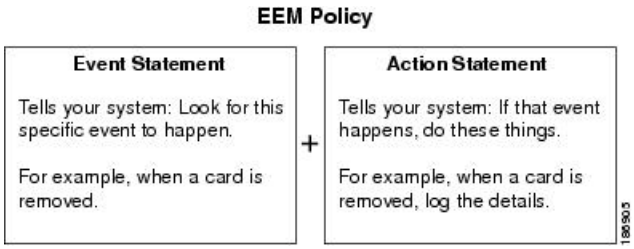
- Action statements—An action that EEM can take, such as sending an e-mail, or disabling an interface, to recover from an event.
- Policies—An event paired with one or more actions to troubleshoot or recover from the event.

# 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.

This figure shows the two basic statements in an EEM policy.

***Figure 1: EEM Policy Statements***




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. You configure EEM policies on the supervisor, and EEM pushes the policy to the correct module based on the event type. EEM takes any actions for a triggered event either locally on the module or on the supervisor (the default option).

EEM maintains event logs on the supervisor.

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 (__).

You can create user policies to suit your network. If you create a user policy, any actions in your policy occur after EEM triggers any system policy actions related to the same event as your policy.

You can also override some system policies. The overrides that you configure take the place of the system policy. You can override the event or the actions.

Use the **show event manager system-policy** command to view the preconfigured system policies and determine which policies that you can override.

**Note** You should use the **show running-config eem** command to check the configuration of each policy. An override policy that consists of an event statement and no action statement triggers no action and no notification of failures.

**Note** Your override policy should always include an event statement. An override policy without an event statement overrides all possible events in the system policy.

The table below lists the system policies that can be completely overridden and policies that are only augmented.

| System Policy | Can be completely overridden?<br>Note Policies with default actions that cannot be completely overridden will be augmented. |
|---|---|
| __BootupPortLoopback | No |
| __FIPS | No |
| __IntPortLoopback | No |
| __PortLoopback | No |
| __RewriteEngineLoopback | No |
| __SnakeLoopback | No |
| __SwPortLoopback | No |
| __asic_register_check | Yes |
| __compact_flash | Yes |
| __eobc_port_loopback | Yes |
| __ethpm_debug_1 | No |
| __ethpm_debug_2 | No |
| __ethpm_debug_3 | No |
| __ethpm_debug_4 | No |
| __ethpm_link_flap | No |
| __external_compact_flash | Yes |
| __gold_obfl | Yes |
| __lcm_module_failure | Yes |

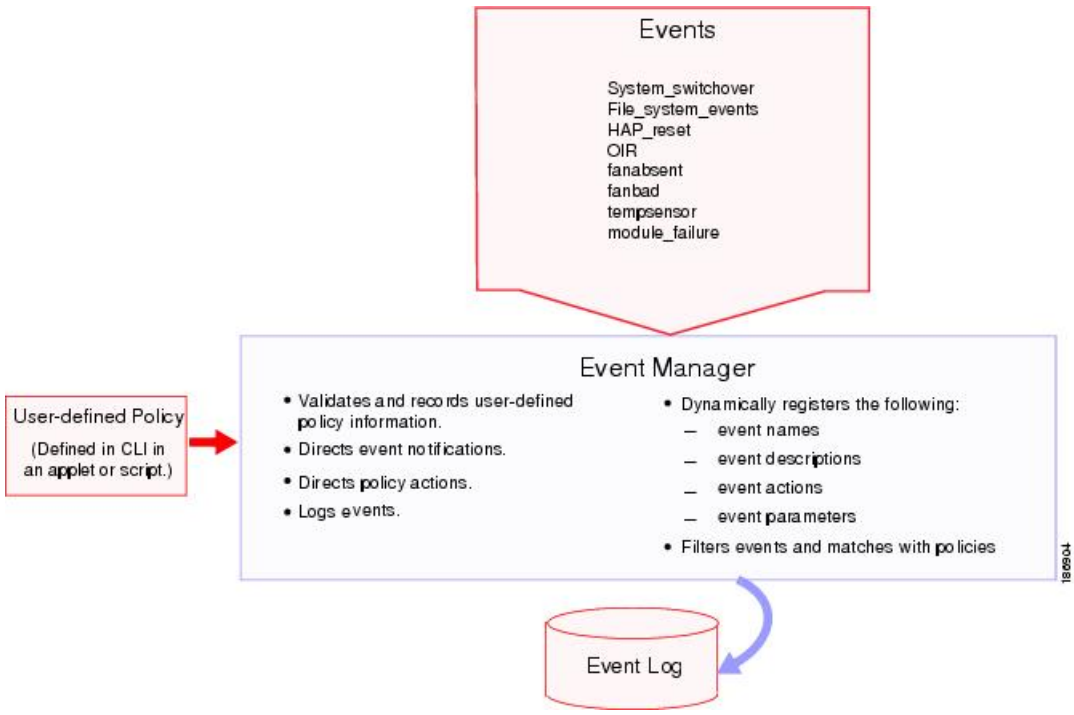
# Event Statements

An event is any device activity for which some action, such as a workaround or a notification, should be taken. In many cases, these events are related to faults in the device such as when an interface or a fan malfunctions.

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

This figure shows events that are handled by EEM.

*Figure 2: EEM Overview*




Event statements specify the event that triggers a policy to run. You can configure multiple event triggers.

EEM schedules and runs policies on the basis of event statements. EEM examines the event and action commands and runs them as defined.

**Note** If you want to allow the triggered event to process any default actions, you must configure the EEM policy to allow the event default action statement.

# Action Statements

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

EEM supports the following actions in action statements:

- Execute any CLI commands.
- Update a counter.
- Log an exception.
- Force the shutdown of any module.
- Reload the device.
- Shut down specified modules because the power is over budget.
- Generate a syslog message.

- Generate a Call Home event.
- Generate an SNMP notification.
- Use the default action for the system policy.

**Note** If you want to allow the triggered event to process any 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 will not allow the CLI command to execute.

**Note** Verify that your action statements within your user policy or overriding policy do not negate each other or adversely affect the associated system policy.

# VSH Script Policies

You can also write policies in a VSH script, using a text editor. These policies have an event statement and action statement(s) just as other policies, and these policies can either augment or override system policies. After you write your VSH script policy, copy it to the device and activate it.

# Environment Variables

You can define environment variables for EEM that are available for all policies. Environment variables are useful for configuring common values that you can use in multiple policies. For example, you can create an environment variable for the IP address of an external e-mail server.

You can use an environment variable in action statements by using the parameter substitution format.

This example shows a sample action statement to force a module 1 shutdown, with a reset reason of "EEM action."

```
switch (config-eem-policy)# action 1.0 forceshut module 1 reset-reason “EEM action.”
```

If you define an environment variable for the shutdown reason, called default-reason, you can replace that reset reason with the environment variable, as shown in the following example.

```
switch (config-eem-policy)# action 1.0 foreshut module 1 reset-reason $default-reason
```

You can reuse this environment variable in any policy.

# EEM Event Correlation

You can trigger an EEM policy based on a combination of events. First, you use the **tag** keyword to create and differentiate multiple events in the EEM policy. Then using a set of boolean operators (**and**, **or**, **andnot**), along with the count and time, you can define a combination of these events to trigger a custom action.

## High Availability

Cisco NX-OS supports stateless restarts for EEM. After a reboot or supervisor switchover, Cisco NX-OS applies the running configuration.

## Virtualization Support

You configure EEM in the virtual device context (VDC) that you are logged into. By default, Cisco NX-OS places you in the default VDC. You must be in this VDC to configure policies for module-based events.

Not all actions or events are visible in all VDCs. You must have network-admin or vdc-admin privileges to configure policies.

See the *Cisco Nexus 7000 Series NX-OS Virtual Device Context Configuration Guide* for more information on VDCs.

# Prerequisites for EEM

EEM has the following prerequisites:

- The username: admin (with network-admin or vdc-admin user privileges) is required to configure EEM on a nondefault VDC.

# Guidelines and Limitations for EEM

EEM has the following configuration guidelines and limitations:

- 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.
- 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 CLI command in a match statement, you must add the event-default action statement to the EEM policy or EEM will not allow the CLI command to execute.
- 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.
- The following rules apply to regular command expressions: all keywords must be expanded, and only the * symbol can be used for argument replacement.
- EEM event correlation is supported only on the supervisor module.
- EEM event correlation is not supported across different modules within a single policy.
- 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, module, module-failure, oir, snmp, and syslog.

- 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.
- While usign an EEM applet, the **copy r bootflash:last_config** command prompts for overriding the configuration file if the same file name is present. You need to add the **terminal dont-ask** if you are prompted to proceed with overwrtting a file in an EEM applet. Refer to the example given below.

```
event manager applet test
            event cli match "rollback *"
            action 1.0 cli command "terminal dont-ask"
            action 2.0 cli command "copy running-config bootflash:last_config"
            action 3.0 cli command "no terminal dont-ask"
```

- You can invoke EEM from Python. For more information about Python, see the *Cisco Nexus 7000 Series NX-OS Programmability Guide*.

# Default Settings for EEM

This table lists the default settings for EEM parameters.

| Parameters | Default |
|---|---|
| System policies | Active |

# Configuring EEM

You can create policies that contain actions to take based on system policies. To display information about the system policies, use the **show event manager system-policy** command.

## Defining an Environment Variable

You can define a variable to serve as a parameter in an EEM policy.

**Procedure**

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **configure terminal**<br>**Example:**<br>`switch# configure terminal`<br>`switch(config)#` | Enters global configuration mode. |
| **Step 2** | **event manager environment** *variable-name variable-value*<br>**Example:** | 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 alphanumeric string up to 39 characters. |

| | Command or Action | Purpose |
|---|---|---|
| | `switch(config)# event manager environment emailto "admin@anyplace.com"` | |
| **Step 3** | (Optional) **show event manager environment** {*variable-name* \| **all**}<br>**Example:**<br>`switch(config)# show event manager environment all` | Displays information about the configured environment variables. |
| **Step 4** | (Optional) **copy running-config startup-config**<br>**Example:**<br>`switch(config)# copy running-config startup-config` | Copies the running configuration to the startup configuration. |

# Defining a User Policy Using the CLI

You can define a user policy using the CLI to the device.

### Procedure

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **configure terminal**<br>**Example:**<br>`switch# configure terminal`<br>`switch(config)#` | Enters global configuration mode. |
| **Step 2** | **event manager applet** *applet-name*<br>**Example:**<br>`switch(config)# event manager applet monitorShutdown`<br>`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*<br>**Example:**<br>`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*<br>**Example:**<br>`switch(config-applet)# event cli match "conf t ; interface * ; shutdown"` | Configures the event statement for the policy. Repeat this step for multiple event statements. See Configuring Event Statements, on page 9. |
| **Step 5** | (Optional) **tag** *tag* {**and** \| **andnot** \| **or**} *tag* [**and** \| **andnot** \| **or** {*tag*}] {**happens** *occurs in seconds*}<br>**Example:** | Correlates multiple events in the policy.<br>The range for the *occurs* argument is from 1 to 4294967295. The range for the *seconds* argument is from 0 to 4294967295 seconds. |

| | Command or Action | Purpose |
|---|---|---|
| | `switch(config-applet)# tag one or two happens 1 in 10000` | |
| **Step 6** | **action** *label action-statement*<br>**Example:**<br>`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. See Configuring Action Statements, on page 14. |
| **Step 7** | (Optional) **show event manager policy-state** *name* [**module** *module-id*]<br>**Example:**<br>`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**<br>**Example:**<br>`switch(config)# copy running-config startup-config` | Copies the running configuration to the startup configuration. |

## Configuring Event Statements

Use one of the following commands in Applet Configuration mode to configure an event statement:

| Command | Purpose |
|---|---|
| **event application** [**tag** *tag*] **sub-system** *sub-system-id* **type** *event-type*<br>**Example:**<br>`switch(config-applet)# event application sub-system 798 type 1` | Triggers an event when an event specification matches the subsystem ID and application event type.<br>The range for the *sub-system-id* and for the *event-type* is from 1 to 4294967295.<br>The **tag** *tag* keyword-argument pair identifies this specific event when multiple events are included in the policy.<br>**Note** To use this command, you must first enable the **feature evmed** command to enable generic event detectors. |
| **event cli** [**tag** *tag*] **match** *expression* [**count** *repeats* \| **time** *seconds*]<br>**Example:**<br>`switch(config-applet)# event cli match "conf t ; interface * ; shutdown"` | Triggers an event if you enter a command that matches the regular expression.<br>The **tag** *tag* keyword-argument pair identifies this specific event when multiple events are included in the policy.<br>The *repeats* range is from 1 to 65000. The time range, in seconds, is from 0 to 4294967295, where 0 indicates no time limit. |

| Command | Purpose |
|---|---|
| **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**}]<br>**Example:**<br>`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.<br>The **tag** *tag* keyword-argument pair identifies this specific event when multiple events are included in the policy.<br>The *counter* name can be any case-sensitive, alphanumeric string up to 28 characters. The *entry* and *exit* value ranges are from 0 to 2147483647. |
| **event fanabsent** [**fan** *number*] **time** *seconds*<br>**Example:**<br>`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 module dependent. The *seconds* range is from 10 to 64000. |
| **event fanbad** [**fan** *number*] **time** *seconds*<br>**Example:**<br>`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. |
| **event fib** {**adjacency extra** \| **resource tcam usage** \| **route** {**extra** \| **inconsistent** \| **missing**}}<br>**Example:**<br>`switch(config-applet)# event fib adjacency extra` | Triggers an event for one of the following:<br>• **adjacency extra**—If there is an extra route in the unicast FIB.<br>• **resource tcam usage**—Each time the TCAM utilization percentage becomes a multiple of 5, in either direction.<br>• **route** {**extra** \| **inconsistent** \| **missing**}—If a route is added, changed, or deleted in the unicast FIB. |

| Command | Purpose |
|---|---|
| **event gold** [**failure-type** {**sup** \| **fabric** \| **lc** \| **port**}] **module** {*module* \| **all**} **test** {*test-name* \| *test-id*} [**severity** {**major** \| **minor** \| **moderate**}] **testing-type** {**bootup** \| **monitoring** \| **ondemand** \| **scheduled**} **consecutive-failure** *count*<br>**Example:**<br>`switch(config-applet)# event gold failure-type module 2`<br>`test 7 ASICRegisterCheck testing-type ondemand consecutive-failure 2` | Triggers an event if the named online diagnostic test experiences the configured failure severity for the configured number of consecutive failures.<br>The *module* specifies the number of the module that needs to be monitored.<br>The *test-name* is the name of a configured online diagnostic test. The *test-id* specifies the test ID of the event criteria. The range is from 1 to 30.<br>The *count* range is from 1 to 1000. |
| **event interface** [**tag** *tag*] {**name** *interface slot*/*port* **parameter**}<br>**Example:**<br>`switch(config-applet)# event interface ethernet 2/2 parameter` | Triggers an event if the counter is exceeded for the specified interface.<br>The **tag** *tag* keyword-argument pair identifies this specific event when multiple events are included in the policy.<br>**Note** To use this command, you must first enable the **feature evmed** command to enable generic event detectors. |
| **event memory** {**critical** \| **minor** \| **severe**}<br>**Example:**<br>`switch(config-applet)# event memory critical` | Triggers an event if a memory threshold is crossed. See also Configuring Memory Thresholds, on page 25. |
| **event module** [**tag** *tag*] **status** {**online** \| **offline** \| **any**} **module** {**all** \| *module-num*}<br>**Example:**<br>`switch(config-applet)# event module status offline module all` | Triggers an event if the specified module enters the selected status.<br>The **tag** *tag* keyword-argument pair identifies this specific event when multiple events are included in the policy. |
| **event module-failure** [**tag** *tag*] **type** *failure-type* **module** {*slot* \| **all**} **count** *repeats* [**time** *seconds*]<br>**Example:**<br>`switch(config-applet)# event module-failure type lc-failed module 3 count 1` | Triggers an event if a module experiences the failure type configured.<br>The **tag** *tag* keyword-argument pair identifies this specific event when multiple events are included in the policy.<br>The *repeats* range is from 0 to 4294967295. The *seconds* range is from 0 to 4294967295, where 0 indicates no time limit. |

| Command | Purpose |
|---|---|
| **event none**<br>**Example:**<br>`switch(config-applet)# event none` | Manually runs the policy event without any events specified.<br>**Note** To use this command, you must first enable the **feature evmed** command to enable generic event detectors. |
| **event oir** [**tag** *tag*] {**fan** \| **module** \| **powersupply**} {**anyoir** \| **insert** \| **remove**} [*number*]<br>**Example:**<br>`switch(config-applet)# event oir fan remove 4` | Triggers an event if the configured device element (fan, module, or power supply) is inserted or removed from the device.<br>The **tag** *tag* keyword-argument pair identifies this specific event when multiple events are included in the policy.<br>You can optionally configure a specific fan, module, or power supply number. The *number* range is as follows:<br>• Fan number—Module dependent.<br>• Module number—Device dependent.<br>• Power supply number—The range is from 1<br>to 3. |
| **event policy-default count** *repeats* [**time** *seconds*]<br>**Example:**<br>`switch(config-applet)# event policy-default count 3` | Uses the event configured in the system policy. Use this option for overriding policies.<br>The *repeats* range is from 1 to 65000. The *seconds* range is from 0 to 4294967295, where 0 indicates no time limit. |
| **event poweroverbudget**<br>**Example:**<br>`switch(config-applet)# event poweroverbudget` | Triggers an event if the power budget exceeds the capacity of the configured power supplies. |

| Command | Purpose |
|---|---|
| **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*<br>**Example:**<br>`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.<br>The **tag** *tag* keyword-argument pair identifies this specific event when multiple events are included in the policy.<br>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 1 to 2147483647. |
| **event storm-control**<br>**Example:**<br>`switch(config-applet)# event storm-control` | Triggers an event if traffic on a port exceeds the configured storm control threshold. |
| **event syslog** [**occurs** *count*] {**pattern** *string* \| **period** *time* \| **priority** *level* \| **tag** *tag*}<br>**Example:**<br>`switch(config-applet)# event syslog period 500` | Triggers an event if the specified syslog threshold is exceeded. The range for the count is from 1 to 65000, and the range for the time is from 1 to 4294967295. The priority range is from 0 to 7.<br>The **tag** *tag* keyword-argument pair identifies this specific event when multiple events are included in the policy. |
| **event sysmgr memory** [**module** *module-num*] **major** *major-percent* **minor** *minor-percent* **clear** *clear-percent*<br>**Example:**<br>`switch(config-applet)# event sysmgr memory minor 80` | Triggers an event if the specified system manager memory threshold is exceeded. The range for the percentage is from 1 to 99. |
| **event sysmgr switchover count** *count* **time** *interval*<br>**Example:**<br>`switch(config-applet)# event sysmgr switchover count 10 time 1000` | Triggers an event if the specified switchover count is exceeded within the time interval specified. The switchover count is from 1 to 65000. The time interval is from 0 to 2147483647. |
| **event temperature** [**module** *slot*] [*sensor-number*] **threshold** {**any** \| **major** \| **minor**}<br>**Example:**<br>`switch(config-applet)# event temperature module 2 threshold any` | Triggers an event if the temperature sensor exceeds the configured threshold. The sensor range is from 1 to 18. |

| Command | Purpose |
|---|---|
| **event timer** {**absolute time** *time* **name** *name* \| **countdown time** *time* **name** *name* \| **cron cronentry** *string* \| **tag** *tag* \| **watchdog time** *time* **name** *name*}<br>**Example:**<br>`switch(config-applet)# event timer absolute time 100 name abtimer` | Triggers an event if the specified time is reached. The range for the time is from 1 to 4294967295.<br>• **absolute time**—Triggers an event when the specified absolute time of day occurs.<br>• **countdown time**—Triggers an event when when the specified time counts down to zero. The timer does not reset.<br>• **cron cronentry**—Triggers an event when the CRON string specification matches the current time.<br>• **watchdog time**—Triggers an event when the specified time counts down to zero. The timer automatically resets to the initial value and continues to count down.<br>The **tag** *tag* keyword-argument pair identifies this specific event when multiple events are included in the policy.<br>**Note** To use this command, you must first enable the **feature evmed** command to enable generic event detectors. |
| **event track** [**tag** *tag*] *object-number* **state** {**any** \| **down** \| **up**}<br>**Example:**<br>`switch(config-applet)# event track 1 state down` | Triggers an event if the tracked object is in the configured state.<br>The **tag** *tag* keyword-argument pair identifies this specific event when multiple events are included in the policy.<br>The *object-number* range is from 1 to 500. |

## Configuring Action Statements

Use any of the following commands in Applet configuration (config-applet) mode to configure action statements:

| Command | Purpose |
|---|---|
| **action** *label* **cli** *command1* [*command2...*] [**local**]<br>**Example:**<br>`switch(config-applet)# action 1.0 cli "show interface e 3/1"` | Runs the configured CLI commands. You can optionally run the commands on the module where the event occurred.<br>The action label is in the format number1.number2. *number1* can be any number up to 16 digits. The range for *number2* is from 0 to 9. |
| **action** *label* **counter name** *counter* **value** *val* **op** {**dec** \| **inc** \| **nop** \| **set**}<br>**Example:**<br>`switch(config-applet)# action 2.0 counter name mycounter value 20 op inc` | Modifies the counter by the configured value and operation.<br>The action label is in the format number1.number2. *number1* can be any number up to 16 digits. The range for *number2* is from 0 to 9.<br>The counter name 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. |
| **action** *label* **event-default**<br>**Example:**<br>`switch(config-applet)# action 1.0 event-default` | Executes the default action for the associated event.<br>The action label is in the format number1.number2. *number1* can be any number up to 16 digits. The range for *number2* is from 0 to 9. |
| **action** *label* **forceshut** [**module** *slot* \| **xbar** *xbar-number*] **reset-reason** *seconds*<br>**Example:**<br>`switch(config-applet)# action 1.0 forceshut module 2 reset-reason "flapping links"` | Forces a module, crossbar, or the entire system to shut down.<br>The action label is in the format number1.number2. *number1* can be any number up to 16 digits. The range for *number2* is from 0 to 9.<br>The reset reason is a quoted alphanumeric string up to 80 characters. |
| **action** *label* **overbudgetshut** [**module** *slot*[-*slot*]]<br>**Example:**<br>`switch(config-applet)# action 1.0 overbudgetshut module 3-5` | Forces one or more modules or the entire system to shut down because of a power overbudget issue.<br>The action label is in the format number1.number2. *number1* can be any number up to 16 digits. The range for *number2* is from 0 to 9. |

| Command | Purpose |
|---|---|
| **action** *label* **policy-default**<br>**Example:**<br>`switch(config-applet)# action 1.0 policy-default` | Executes the default action for the policy that you are overriding.<br>The action label is in the format number1.number2. *number1* can be any number up to 16 digits. The range for *number2* is from 0 to 9. |
| **action** *label* **publish-event**<br>**Example:**<br>`switch(config-applet)# action 1.0 publish-event` | Forces the publication of an application-specific event.<br>The action label is in the format number1.number2. *number1* can be any number up to 16 digits. The range for *number2* is from 0 to 9. |
| **action** *label* **reload** [**module** *slot*[*-slot*]]<br>**Example:**<br>`switch(config-applet)# action 1.0 reload module 3-5` | Forces one or more modules or the entire system to reload.<br>The action label is in the format number1.number2. *number1* can be any number up to 16 digits. The range for *number2* is from 0 to 9. |
| **action** *label* **snmp-trap** {[**intdata1** *data* [**intdata2** *data*]] [**strdata** *string*]}<br>**Example:**<br>`switch(config-applet)# action 1.0 snmp-trap strdata "temperature problem"` | Sends an SNMP trap with the configured data.<br>The action label is in the format number1.number2 *number1* can be any number up to 16 digits. The range for *number2* is from 0 to 9.<br>The *data* arguments can by any number up to 80 digits. The *string* can be any alphanumeric string up to 80 characters. |
| **action** *label* **syslog** [**priority** *prio-val*] **msg** *error-message*<br>**Example:**<br>`switch(config-applet)# action 1.0 syslog priority notifications msg "cpu high"` | Sends a customized syslog message at the configured priority.<br>The action label is in the format number1.number2. *number1* can be any number up to 16 digits. The range for *number2* is from 0 to 9.<br>The *error-message* can be any quoted alphanumeric string up to 80 characters. |
| **action** *label* **end**<br>**Example:**<br>`switch(config-applet)# action 1.0 end` | Identifies the end of a conditional action block like if/else and while.<br>The action label is in the format number1.number2. *number1* can be any number up to 16 digits. The range for *number2* is from 0 to 9. |

| Command | Purpose |
|---|---|
| **action** *label* **exit** [*result* ]<br>**Example:**<br>`switch(config-applet)# action 1.0 exit 25` | Exits from the applet configuration mode that is currently running.<br>The action label is in the format number1.number2. *number* can be any number up to 16 digits. The range for *number2* is from 0 to 9. |
| **action** *label* **else**<br>**Example:**<br>`switch(config-applet)# action 1.0 else` | Identifies the beginning of an *else* conditional action block in an *if*/*else* action block.<br>The action label is in the format number1.number2. *number1* can be any number up to 16 digits. The range for *number2* is from 0 to 9. |
| **action** *label* **elseif** *string-1* {**eq** \| **gt** \| **ge** \| **lt** \| **le** \| **ne**} *string-2*<br>`switch(config-applet)# action 1.0 elseif $x ge 10` | Identifies the beginning of an *elseif* conditional action block in an *else/if* action block.<br>The action label is in the format number1.number2. *number1* can be any number up to 16 digits. The range for *number2* is from 0 to 9. |
| **action** *label* **if** *string-1* {**eq** \| **gt** \| **ge** \| **lt** \| **le** \| **ne**} *string-2*<br>`switch(config-applet)# action 1.0 if $x lt 10` | Identifies the beginning of an *if* conditional action block.<br>The action label is in the format number1.number2. *number1* can be any number up to 16 digits. The range for *number2* is from 0 to 9. |
| **action** *label* **if** *string-1* {**eq** \| **gt** \| **ge** \| **lt** \| **le** \| **ne**} *string-2* **goto** *label*<br>`switch(config-applet)# action 2.0 if $x lt 10 goto 1.0` | Instructs the applet to jump to a given label if the specified condition is true.<br>The action label is in the format number1.number2. *number1* can be any number up to 16 digits. The range for *number2* is from 0 to 9. |
| **action** *label* **puts** *string*<br>**Example:**<br>`switch(config-applet)# action 2.0 puts "Hello world"` | Enables the action of printing data directly to the terminal.<br>The action label is in the format number1.number2. *number1* can be any number up to 16 digits. The range for *number2* is from 0 to 9. |

| Command | Purpose |
|---|---|
| **action** *label* **add** {*long-integer* \| *variable-name*} {*long-integer* \| *variable-name*}<br>**Example:**<br>`switch(config-applet)# action 2.0 add $var1 10` | Specifies the action of adding two variables.<br>The action label is in the format number1.number2. *number1* can be any number up to 16 digits. The range for *number2* is from 0 to 9. |
| **action** *label* **decrement** *variable-name long-integer*<br>**Example:**<br>`switch(config-applet)# action 1.0 decrement $varname 12` | Specifies the action of decrementing the value of a variable.<br>The action label is in the format number1.number2. *number1* can be any number up to 16 digits. The range for *number2* is from 0 to 9. |
| **action** *label* **increment** *variable-name long-integer*<br>**Example:**<br>`switch(config-applet)# action 2.0 increment $varname 12` | Specifies the action of incrementing the value of a variable.<br>The action label is in the format number1.number2. *number1* can be any number up to 16 digits. The range for *number2* is from 0 to 9. |
| **action** *label* **multiply** {*long-integer1* \| *variable-name1*} {*long-integer2* \| *variable-name2*}<br>`switch(config-applet)# action 2.0 multiply 12 35` | Specifies the action of multiplying a variable value with a long integer value.<br>The action label is in the format number1.number2. *number1* can be any number up to 16 digits. The range for *number2* is from 0 to 9. |
| **action** *label* **subtract** {*long-integer1* \| *variable-name1*} {*long-integer2* \| *variable-name2*}<br>**Example:**<br>`switch(config-applet)# action 2.0 subtract $var1 $var2` | Specifies the action of subtracting the value of a variable from another one.<br>The action label is in the format number1.number2. *number1* can be any number up to 16 digits. The range for *number2* is from 0 to 9. |
| **action** *label* **comment** *string*<br>**Example:**<br>`switch(config-applet)# action 2.0 comment keyvalue` | Adds comments to applets.<br>The action label is in the format number1.number2. *number1* can be any number up to 16 digits. The range for *number2* is from 0 to 9. |
| **action** *label* **break**<br>**Example:**<br>`switch(config-applet)# action 2.0 break` | Specifies the action of exiting from a loop of actions.<br>The action label is in the format number1.number2. *number1* can be any number up to 16 digits. The range for *number2* is from 0 to 9. |

| Command | Purpose |
|---|---|
| **action** *label* **continue**<br>**Example:**<br>`switch(config-applet)# action 2.0 continue` | Specifies the action of continuing with a loop of actions.<br>The action label is in the format number1.number2. *number1* can be any number up to 16 digits. The range for *number2* is from 0 to 9. |
| **action** *label* **foreach** *string-iterator string-input* [*string-delimiter*]<br>**Example:**<br>`switch(config-applet)# action 3.1 foreach _iterator "orange blue green"` | Specifies the iteration of an input string using the delimiter as the tokenizing pattern.<br>The action label is in the format number1.number2. *number1* can be any number up to 16 digits. The range for *number2* is from 0 to 9. |
| **action** *label* **while** *string-op-1 operator string-op-2*<br>**Example:**<br>`switch(config-applet)# action 3.2 while $i lt 10` | Identifies the beginning of a loop action block.<br>The action label is in the format number1.number2. *number1* can be any number up to 16 digits. The range for *number2* is from 0 to 9.<br>Valid values for *operator* are: ge, gt, eq, ne, lt, le. |

Use any of the following action commands in Applet Configuration (config-applet) mode to enable string operations.

| Command | Purpose |
|---|---|
| **action** *label* **append** *var-name* [*var-value*]<br>`switch(config-applet)# action 4.2 append $var 12` | Specifies the action of appending the string value to the current value of a variable.<br>The action label is in the format number1.number2. *number1* can be any number up to 16 digits. The range for *number2* is from 0 to 9. If the variable does not exist, it will be created and set to the given value. |
| **action** *label* **regexp** *string-pattern string-input* [*string-match* [*string-submatch1*] [*string-submatch2*] [*string-submatch3*]]<br>`switch(config-applet)# action 4.3 regexp "(.*) (.*) (.*)" "one two three" _match _sub1` | Matches the regular expression in *string-pattern* on the *string-input*. *string-match* and *string-submatch* store the results of the match.<br>The action label is in the format number1.number2.<br>*number1* can be any number up to 16 digits. The range for *number2* is from 0 to 9. |

| Command | Purpose |
|---|---|
| **action** *label* **string compare** [**nocase**] [**length** *integer*] *string1 string2*<br>`switch(config-applet)# action 4.5 string compare nocase length 3` | Compares two unequal strings. The result is stored in the inbuilt variable *$_string_result*.<br>The action label is in the format number1.number2. *number* can be any number up to 16 digits. The range for *number2* is from 0 to 9. |
| **action** *label* **string equal** [**nocase**] [**length** *integer*] *string1 string2*<br>`switch(config-applet)# action 4.5 string equal "contains" "data"` | Compares two strings and returns 1 if the two strings are equal. The result is stored in the inbuilt variable *$_string_result*.<br>The action label is in the format number1.number2. *number1* can be any number up to 16 digits. The range for *number2* is from 0 to 9. |
| **action** *label* **string first** *string1 string2* [*index-value*]<br>`switch(config-applet)# action 4.6 string first "contains" $str` | Returns the index of the first occurrence of *string1* within *string2*. *index-value* is optional and indicates the position to start the first test.<br>The action label is in the format number1.number2. *number1* can be any number up to 16 digits. The range for *number2* is from 0 to 9. |
| **action** *label* **string index** *string* [*value* \| *end*]<br>`switch(config-applet)# action 4.7 string index "this is a test" 6` | Returns the characters specified at the given index-value. *end* denotes the last character of the string. The characters are stored in the inbuilt variable *$_string_result*.<br>The action label is in the format number1.number2. *number1* can be any number up to 16 digits. The range for *number2* is from 0 to 9. |
| **action** *label* **string last** *string1 string2* [*index-value*]<br>`switch(config-applet)# action 4.9 string last "contains" $str` | Returns the index of the last occurrence of *string1* within *string2*.<br>The action label is in the format number1.number2. *number1* can be any number up to 16 digits. The range for *number2* is from 0 to 9. |
| **action** *label* **string length** *string*<br>`switch(config-applet)# action 5.0 string length "contains"` | Returns the number of characters in a string. The result is stored in the inbuilt variable *$_string_result*.<br>The action label is in the format number1.number2. *number1* can be any number up to 16 digits. The range for *number2* is from 0 to 9. |
| **action** *label* **string match** [**nocase**] *string-pattern string*<br>`switch(config-applet)# action 5.2 string match "*Bl*"  $str` | Matches *string* with a specified pattern, *string-pattern*. If they match, the result 1 is stored in the inbuilt variable *$_string_result*.<br>The action label is in the format number1.number2. *number1* can be any number up to 16 digits. The range for *number2* is from 0 to 9. |

| Command | Purpose |
|---|---|
| **action** *label* **string range** *string start-index end-index*<br>`switch(config-applet)# action 5.2 string range "$data" 4 9` | Stores a range of characters in a string, starting from the *start-index* and ending at *end-index*. The resultant characters are stored in the inbuilt variable *$_string_result*.<br>The action label is in the format number1.number2. *number1* can be any number up to 16 digits. The range for *number2* is from 0 to 9. |
| **action** *label* **string replace** *string start-index end-index* [*new-string*]<br>`switch(config-applet)# action 5.4 string replace $str 1 4 "test"` | Forms a new string by replacing specific characters of a string. If *new-string* is not specified, it replaces the characters with whitespace. The newly formed string is stored in the inbuilt variable *$_string_result*.<br>The action label is in the format number1.number2. *number* can be any number up to 16 digits. The range for *number2* is from 0 to 9. |
| **action** *label* **string tolower** *string* [*start-index*] [*end-index*]<br>`switch(config-applet)# action 5.5 string tolower "$string" 11 16` | Stores a specific range of characters of a string in lowercase. The characters are stored in the inbuilt variable *$_string_result*.<br>The action label is in the format number1.number2. *number1* can be any number up to 16 digits. The range for *number2* is from 0 to 9. |
| **action** *label* **string toupper** *string* [*start-index*] [*end-index*]<br>`switch(config-applet)# action 5.6 string toupper "$string" 0 7` | Stores a specific range of characters of a string in uppercase. The characters are stored in the inbuilt variable *$_string_result*<br>The action label is in the format number1.number2. *number* can be any number up to 16 digits. The range for *number2* is from 0 to 9. |
| **action** *label* **string trim** *string1* [*string2*]<br>`switch(config-applet)# action 5.7 string trim "$string"` | Trims the characters in *string2* from both ends of *string1*. By default, *string2* corresponds to whitespace.<br>The action label is in the format number1.number2. *number1* can be any number up to 16 digits. The range for *number2* is from 0 to 9. |
| **action** *label* **string trimleft** *string1* [*string2*]<br>`switch(config-applet)# action 5.7 string trimleft "$string" "Hello"` | Trims the characters in *string2* from the left end of *string1*. By default, *string2* corresponds to whitespace.<br>The action label is in the format number1.number2. *number1* can be any number up to 16 digits. The range for *number2* is from 0 to 9. |
| **action** *label* **string trimright** *string1* [*string2*]<br>`switch(config-applet)# action 5.7 string trimright "this is a testtest" "test"` | Trims the characters in *string2* from the right end of *string1*. By default, *string2* corresponds to whitespace.<br>The action label is in the format number1.number2. *number1* can be any number up to 16 digits. The range for *number2* is from 0 to 9. |

| Command | Purpose |
|---|---|
| **action** *label* **set** *variable-name variable-value*<br>`switch(config-applet)# action 6.0 set $string "Container"` | Sets the value of a variable.<br>The action label is in the format number1.number2. *number1* can be any number up to 16 digits. The range for *number2* is from 0 to 9. |

**Note** If you want to allow the triggered event to process any 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 will not allow the CLI command to execute. You can use the **terminal event-manager bypass** command to allow all EEM policies with CLI matches to execute the CLI command.

# Defining a Policy Using a VSH Script

You can define a policy using a VSH script.

### Before you begin

Ensure that you are logged in with administrator privileges.

Ensure that your script name is the same name as the script filename.

### Procedure

**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.

# Registering and Activating a VSH Script Policy

You can register and activate a policy defined in a VSH script.

### Procedure

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **configure terminal**<br>**Example:**<br>`switch# configure terminal`<br>`switch(config)#` | Enters global configuration mode. |

| | Command or Action | Purpose |
|---|---|---|
| **Step 2** | **event manager policy** *policy-script*<br>**Example:**<br>`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** | **show event manager policy internal** *name*<br>**Example:**<br>`switch(config)# show event manager policy internal moduleScript` | (Optional) Displays information about the configured policy. |
| **Step 4** | **copy running-config startup-config**<br>**Example:**<br>`switch(config)# copy running-config startup-config` | (Optional) Copies the running configuration to the startup configuration. |

# Scheduling an EEM Policy

You can schedule an EEM policy that is registered and set the policy scheduling options.

### Procedure

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **configure terminal**<br>**Example:**<br>`switch# configure terminal`<br>`switch(config)#` | Enters the global configuration mode. |
| **Step 2** | **event manager scheduler applet thread class** *class-options* **number** *thread-number*<br>**Example:**<br>`switch(config)# event manager scheduler applet thread class default number 2` | Schedules an EEM policy and sets the policy scheduling options like class and thread number for execution. |
| **Step 3** | **event manager scheduler script thread class** *class-options* **range** *class-range* **number** *thread-number*<br>**Example:**<br>`switch(config)# event manager scheduler script thread class A B range D-E number 1` | Schedules an EEM policy and sets the script scheduling options. |
| **Step 4** | **event manager scheduler clear** {**all** \| **policy** *job-id* \| **queue-type applet** [**class** *class-options*]} [**processor {rp_primary \| rp_standby}**]<br>**Example:** | Clears the EEM policies that are currently executing or pending execution. |

| | Command or Action | Purpose |
|---|---|---|
| | `switch# event manager scheduler clear policy 2` | |
| **Step 5** | **event manager scheduler hold** {**all** \| **policy** *job-id* \| **queue-type applet** [**class** *class-options*]}<br>**Example:**<br>`switch# event manager scheduler hold policy 2` | Holds a scheduled EEM policy event or event queue in the EEM scheduler. |
| **Step 6** | **event manager scheduler modify** {**all** \| **policy** *job-id* \| **queue-type applet**} {**class** *class-options* [**queue-priority** {**high** \| **last** \| **low** \| **normal**}] \| **queue-priority** {**high** \| **last** \| **low** \| **normal**} [**class** *class-options*]}<br>**Example:**<br>`switch# event manager scheduler modify all class A` | Modifies the scheduling parameters of the EEM policy. |
| **Step 7** | **event manager scheduler release** {**all** \| **policy** *policy-id* \| **queue-type applet** [**class** *class-options*]}<br>**Example:**<br>`switch# event manager scheduler release all` | Releases the EEM policies held through the event manger scheduler hold command. |

# Overriding a Policy

You can override a system policy.

### Procedure

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **configure terminal**<br>**Example:**<br>`switch# configure terminal`<br>`switch(config)#` | Enters global configuration mode. |
| **Step 2** | (Optional) **show event manager policy-state** *system-policy*<br>**Example:**<br>`switch(config-applet)# show event manager policy-state __ethpm_link_flap`<br>`Policy __ethpm_link_flap`<br>`Cfg count : 5`<br>`Cfg time interval : 10.000000 (seconds)`<br>`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. For information about system policies, see Embedded Event Manager System Events and Configuration Examples. |

| | Command or Action | Purpose |
|---|---|---|
| **Step 3** | **event manager applet** *applet-name* **override** *system-policy*<br>**Example:**<br>`switch(config)# 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 29 characters. The *system-policy* must be one of the existing system policies. |
| **Step 4** | (Optional) **description** *policy-description*<br>**Example:**<br>`description "Overrides link flap policy."` | 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 5** | Required: **event** *event-statement*<br>**Example:**<br>`switch(config-applet)# event policy-default count 2 time 1000` | Configures the event statement for the policy. |
| **Step 6** | Required: **action** *number action-statement*<br>**Example:**<br>`switch(config-applet)# action 1.0 syslog priority warnings msg "Link is flapping."` | Configures an action statement for the policy.<br>Repeat this step for multiple action statements. |
| **Step 7** | (Optional) **show event manager policy-state** *name*<br>**Example:**<br>`switch(config-applet)# show event manager policy-state ethport` | Displays information about the configured policy. |
| **Step 8** | (Optional) **copy running-config startup-config**<br>**Example:**<br>`switch(config)# copy running-config startup-config` | Copies the running configuration to the startup configuration. |

# Configuring Memory Thresholds

You can set the memory thresholds used to trigger events and set whether the operating system should kill processes if it cannot allocate memory.

### Before you begin

Ensure that you are logged in with administrator privileges.

### Procedure

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **configure terminal**<br>**Example:**<br>`switch# configure terminal`<br>`switch(config)#` | Enters global configuration mode. |
| **Step 2** | **system memory-thresholds minor** *minor* **severe** *severe* **critical** *critical*<br>**Example:**<br>`switch(config)# system memory-thresholds minor 60 severe 70 critical 80` | Configures the system memory thresholds that generate EEM memory events. The default values are as follows:<br>• Minor-85<br>• Severe-90<br>• Critical-95<br>When these memory thresholds are exceeded, the system generates the following syslogs:<br>• 2013 May 7 17:06:30 switch %$ %PLATFORM-2-MEMORY_ALERT: Memory Status Alert : MINOR<br>• 2013 May 7 17:06:30 switch %$ %PLATFORM-2-MEMORY_ALERT: Memory Status Alert : SEVERE<br>• 2013 May 7 17:06:30 switch %$ %PLATFORM-2-MEMORY_ALERT: Memory Status Alert : CRITICAL<br>• 2013 May 7 17:06:35 switch %$ %PLATFORM-2-MEMORY_ALERT: Memory Status Alert : MINOR ALERT RECOVERED<br>• 2013 May 7 17:06:35 switch %$ %PLATFORM-2-MEMORY_ALERT: Memory Status Alert : SEVERE ALERT RECOVERED<br>• 2013 May 7 17:06:35 switch %$ %PLATFORM-2-MEMORY_ALERT: Memory Status Alert : CRITICAL ALERT RECOVERED |
| **Step 3** | (Optional) **system memory-thresholds threshold critical no-process-kill**<br>**Example:**<br>`switch(config)# system memory-thresholds threshold critical no-process-kill` | Configures the system to not kill processes when the memory cannot be allocated. The default value is to allow the system to kill processes, starting with the one that consumes the most memory. |
| **Step 4** | (Optional) **show running-config \| include "system memory"**<br>**Example:** | Displays information about the system memory configuration. |

| | Command or Action | Purpose |
|---|---|---|
| | `switch(config-applet)# show running-config \| include "system memory"` | |
| **Step 5** | (Optional) **copy running-config startup-config**<br>**Example:**<br>`switch(config)# copy running-config startup-config` | Copies the running configuration to the startup configuration. |

# Configuring Syslog as EEM Publisher

You can monitor syslog messages from the switch.

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

### Before you begin

EEM should be available for registration by syslog.

The syslog daemon must be configured and executed.

### Procedure

| | Command or Action | Purpose |
|---|---|---|
| **Step 1** | **configure terminal**<br>**Example:**<br>`switch# configure terminal`<br>`switch(config)#` | Enters global configuration mode. |
| **Step 2** | **event manager applet** *applet-name*<br>**Example:**<br>`switch(config)# event manager applet abc`<br>`switch(config-applet)#` | 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*}<br>**Example:**<br>`switch(config-applet)# event syslog occurs 10` | Monitors syslog messages and invokes the policy based on the search string in the policy.<br>• The **tag** *tag* keyword-argument pair identifies this specific event when multiple events are included in the policy.<br>• The **occurs** *number* keyword-argument pair specifies the number of occurrences. The range is from 1 to 65000.<br>• The **period** *seconds* keyword-argument pair specifies the interval during which the |

| | Command or Action | Purpose |
|---|---|---|
| | | event occurs. The range is from 1 to 4294967295.<br>• The **pattern** *msg-text* keyword-argument pair specifies the matching regular expression. The pattern can contain character text, an environment variable, or a combination of the two. If the string contains embedded blanks, it is enclosed in quotation marks.<br>• The **priority** *priority* keyword-argument pair specifies the priority of the syslog messages. If this keyword is not selected, all syslog messages are set at the informational priority level. |
| **Step 4** | (Optional) **copy running-config startup-config**<br>**Example:**<br>`switch(config)# copy running-config startup-config` | Copies the running configuration to the startup configuration. |

# Verifying the EEM Configuration

To display EEM configuration information, use one of the following commands:

| Command | Purpose |
|---|---|
| **show event manager environment** [*variable-name* \| **all**] | Displays information about the event manager environment variables. |
| **show event manager event-types** [*event* \| **all** \| **module** *slot*] | Displays information about the event manager event types. |
| **show event manager history events** [**detail**] [**maximum** *num-events*] [**severity** {**catastrophic** \| **minor** \| **moderate** \| **severe**}] | Displays the history of events for all policies. |
| **show event manager policy internal** [*policy-name*] [**inactive**] | Displays information about the configured policies. |
| **show event manager policy-state** *policy-name* | Displays information about the policy state, including thresholds. |
| **show event manager script system** [*policy-name* \| **all**] | Displays information about the script policies. |
| **show event manager system-policy** [**all**] | Displays information about the predefined system policies. |

| Command | Purpose |
|---|---|
| **show running-config eem** | Displays information about the running configuration for EEM. |
| **show startup-config eem** | Displays information about the startup configuration for EEM. |
| **show event manager policy active** [**class** *class-options* \| [**detailed**] [**queue-type** [**applet**] ] | Displays the EEM policies that are executing. |
| **show event manager policy pending** [**class** *class-options* \| [**detailed**] [**queue-type applet** [**detailed**] ] | Displays the policies that are pending for execution. |
| **show event manager scheduler thread detailed** | Displays the scheduled activities of the EEM policies. |

# Configuration Examples for EEM

This example shows how to override the __lcm_module_failure system policy by changing the threshold for just module 3 hitless upgrade failures. This example also sends a syslog message. The settings in the system policy, __lcm_module_failure, apply in all other cases.

```
event manager applet example2 override __lcm_module_failure
event module-failure type hitless-upgrade-failure module 3 count 2
action 1 syslog priority errors msg module 3 “upgrade is not a hitless upgrade!”
action 2 policy-default
```

This example shows how to override the __ethpm_link_flap system policy and shuts down the interface:

```
event manager applet ethport override __ethpm_link_flap
event policy-default count 2 time 1000
action 1 cli conf t
action 2 cli int et1/1
action 3 cli no shut
```

This example creates an EEM policy that allows the CLI command to execute but triggers an SNMP notification when a user enters configuration mode on the device:

```
event manager applet TEST
event cli match "conf t"
action 1.0 snmp-trap strdata "Configuration change"
action 2.0 event-default
```

**Note** You must add the **event-default** action statement to the EEM policy or EEM will not allow the CLI command to execute.

This example shows how to correlate multiple events in an EEM policy and execute the policy based on a combination of the event triggers. In this example, the EEM policy is triggered if one of the specified syslog patterns occurs within 120 seconds.

```
event manager applet eem-correlate
event syslog tag one pattern "copy bootflash:.* running-config.*"
event syslog tag two pattern "copy run start"
event syslog tag three pattern "hello"
tag one or two or three happens 1 in 120
action 1.0 reload module 1
```

**Note** For additional EEM configuration examples, see Embedded Event Manager System Events and Configuration Examples.

This example shows how to monitor an interface shutdown with an EEM applet.

```
Devicek# sh run eem

!Command: show running-config eem
!Time: Thu Aug 24 00:21:17 2017

version 8.2(0)SK(1)
event manager applet E1
  event cli match "conf t ; interface * ; shutdown"
  action 1 syslog priority critical msg ""tracked interface shutdown" "
```

# Related Documents

| Related Topic | Document Title |
|---|---|
| EEM commands | *Cisco Nexus 7000 Series NX-OS System Management Command Reference* |
| VDCs | *Cisco Nexus 7000 Series NX-OS Virtual Device Context Configuration Guide* |

# Feature History for EEM

Your software release might not support all the features in this document. For the latest caveats and feature information, see the Bug Search Tool at https://tools.cisco.com/bugsearch/ and the release notes for your software release.

***Table 1: Feature History for EEM***

| Feature Name | Releases | Feature Information |
|---|---|---|
| EEM event correlation | 5.2(1) | Added support for multiple event triggers in a single EEM policy. |
| Syslog as EEM publisher | 5.1(1) | Added support to monitor syslog messages from the switch. |
| Memory thresholds configuration | 4.1(3) | Added a configuration section for memory thresholds. |