



## EEM Scripts

---

Cisco IOS XR Embedded Event Manager (EEM) scripts are also known as event scripts that are triggered automatically in response to events on the router. An event can be any significant occurrence, not limited to errors, that has happened within the system. You can use these scripts to detect issues in the network in real time, program certain conditions in response to the event, detect and generate an action when those conditions are met, and execute policy (script) when an event is generated. The script acts in response to the events and reduces the troubleshooting time involved in resolving the issues. For example, you can enforce LACP dampening if a bundle interface has flapped 5 times in less than 30 secs, and define the script to disable the interface for 2 minutes.

You can programmatically define the event and actions separately and map them using a policy map via CLI or NETCONF RPCs. Whenever the configured event occurs, the action that is mapped to it is executed. The same event and action can be mapped to multiple policy maps. You can map the same event and action in 64 policy maps, and add a maximum of 5 different actions in a policy map.

You can create event scripts using Python 3.5 programming language. For the list of supported Python packages

This chapter gets you started with provisioning your Python automation scripts on the router.



---

**Note** This section does not delve into creating Python scripts, but assumes that you have basic understanding of Python programming language. This section will walk you through the process involved in deploying and using the scripts on the router.

---

- [Workflow to Run Event Scripts, on page 1](#)
- [Example: Shut Inactive Bundle Interfaces Using EEM Script, on page 11](#)

## Workflow to Run Event Scripts

Complete the following tasks to provision eem scripts:

- Download the script—Store the eem script on an HTTP server or copy to the hddisk of the router. Add the eem script from the HTTP server or hddisk to the script management repository on the router using the **script add eem** command.
- Define events—Configure the events with the trigger conditions using the **event manager event-trigger** command.

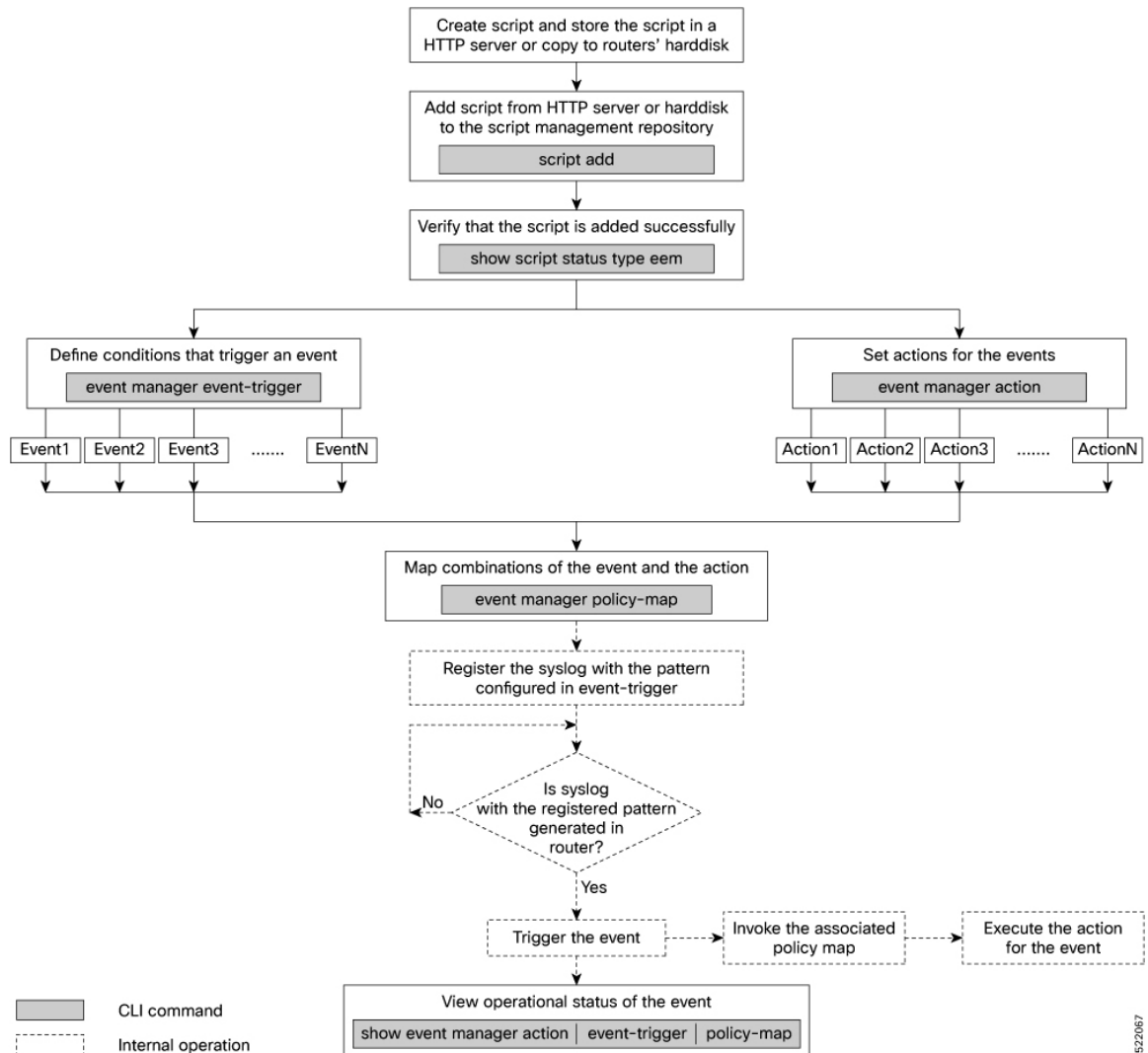
- Define actions to the events—Setup the actions that must be performed in response to an event using **event manager action** command.
- Create policy map—Put together the events and the actions in a policy map using **event manager policy-map** command.



**Note** An eem script is invoked automatically when the event occurs. With the event, the event-trigger invokes the corresponding policy-map to implement the actions in response to the event.

- View operational status of the event—Retrieve the operational data using the **show event-manager action | event-trigger | policy-map** command.

The following image shows a workflow diagram representing the steps involved in using an event script:



522067

## Download the Script to the Router

To manage the scripts, you must add the scripts to the script management repository on the router. A subdirectory is created for each script type. By default, this repository stores the downloaded scripts in the appropriate subdirectory based on script type.

Script Type	Download Location
config	hddisk:/mirror/script-mgmt/config
exec	hddisk:/mirror/script-mgmt/exec
process	hddisk:/mirror/script-mgmt/process
eem	hddisk:/mirror/script-mgmt/eem

The scripts are added to the script management repository using two methods:

- **Method 1:** Add script from a server
- **Method 2:** Copy script from external repository to hddisk using **scp** or **copy** command

In this section, you learn how to add `eem-script.py` script to the script management repository.

**Step 1** Add the script to the script management repository on the router using one of the two options:

- **Add Script From a Server**

Add the script from a configured HTTP server or the hddisk location in the router.

```
Router#script add eem <script-location> <script.py>
```

The following example shows a process script `eem-script.py` downloaded from an external repository `http://192.0.2.0/scripts`:

```
Router#script add eem http://192.0.2.0/scripts eem-script.py
Fri Aug 20 05:03:40.791 UTC
eem-script.py has been added to the script repository
```

You can add a maximum of 10 scripts simultaneously.

```
Router#script add eem <script-location> <script1.py> <script2.py> ... <script10.py>
```

You can also specify the checksum value while downloading the script. This value ensures that the file being copied is genuine. You can fetch the checksum of the script from the server from where you are downloading the script. However, specifying checksum while downloading the script is optional.

```
Router#script add eem http://192.0.2.0/scripts eem-script.py checksum SHA256 <checksum-value>
```

For multiple scripts, use the following syntax to specify the checksum:

```
Router#script add eem http://192.0.2.0/scripts <script1.py> <script1-checksum> <script2.py>
<script2-checksum>
... <script10.py> <script10-checksum>
```

If you specify the checksum for one script, you must specify the checksum for all the scripts that you download.

**Note** Only SHA256 checksum is supported.

- **Copy the Script from an External Repository**

You can copy the script from the external repository to the routers' harddisk and then add the script to the script management repository.

- a. Copy the script from a remote location to harddisk using scp or copy command.

```
Router#scp userx@192.0.2.0:/scripts/eem-script.py /harddisk:/
```

- b. Add the script from the harddisk to the script management repository.

```
Router#script add eem /harddisk:/ eem-script.py
Fri Aug 20 05:03:40.791 UTC
eem-script.py has been added to the script repository
```

**Step 2** Verify that the scripts are downloaded to the script management repository on the router.

**Example:**

```
Router#show script status
Wed Aug 25 23:10:50.453 UTC
=====
Name                | Type      | Status          | Last Action | Action Time
-----
eem-script.py       | eem       | Config Checksum | NEW         | Tue Aug 24 10:44:53 2021
=====
```

Script eem-script.py is copied to harddisk:/mirror/script-mgmt/eem directory on the router.

## Define Trigger Conditions for an Event

You define the event, and create a set of instructions that trigger a match to this event. You can create multiple events.

**Before you begin**

Ensure that the script is added to the script management repository..

**Step 1** Register the event.

**Example:**

```
Router(config)#event manager event-trigger eventT10
```

You can configure more options to trigger an event:

Keyword	Description
occurrence	Number of occurrences before the event is raised. <b>Note</b> The <b>occurrence</b> keyword is supported only for syslog events.
period	Time interval during which configured occurrence should take place. <b>Note</b> The <b>period</b> keyword is supported only for syslog events.

Keyword	Description
type	<p>Configure the type of event.</p> <p><b>Note</b> In Cisco IOS XR Release 7.3.2, you can configure only syslog events. In Cisco IOS XR Release 7.5.1 and later, you can configure the following events:</p> <ul style="list-style-type: none"> <li>• Rate limit—Configure rate limit in seconds or milliseconds. After the event is triggered, the event trigger does not happen even if the event occurs any number of times, till this time has elapsed.</li> <li>• Syslog—Configure syslog pattern, severity.</li> <li>• Timer—Configure watch dog timer in seconds; cron timer as a text string with five fields separated by a space.</li> <li>• Track—Configure event-trigger for track (object tracking), track state (UP, DOWN, or ANY). If event-trigger is configured for track state UP, then it gets triggered when the track state changes from DOWN to UP, and vice-versa.</li> <li>• Telemetry—Define events based on telemetry data. With this feature, you can perform the following operations: <ul style="list-style-type: none"> <li>a. Monitor any operational state such as interface status, and trigger an action when the state changes to a specific value.</li> <li>b. Monitor any counter or statistics in an operational data, and trigger an action when it reaches a threshold.</li> <li>c. Monitor rate of change of any operational attribute, and trigger an action based on threshold.</li> </ul> </li> </ul> <p><b>Note</b> exact match supported on string and threshold or rate limit is supported only for integer type telemetry data</p> <p>Configure sensor path for exact match, threshold or rate depending on the telemetry data type. The exact match is supported on string data type, and threshold and rate limit is supported only for interger data type. Use the following command to verify the sensor path or query before configuring the event trigger.</p> <pre>Router#event manager telemetry sensor-path &lt;sensor-path&gt; json-query &lt;query&gt;</pre> <p>It is mandatory to enable model-driven telemetry using the command:</p> <pre>Router#telemetry model-driven</pre>

**Step 2** Configure the type for the event.

- Syslog:

```
Router(config)#event manager event-trigger eventT10 type syslog pattern
"L2-BM-6-ACTIVE"
```

For syslog, set the pattern to match. In this example, the pattern `L2-BM-6-ACTIVE` is the match value. If a syslog is generated on the router with a pattern that matches this configured pattern, the event gets triggered.

- Timer:

Watchdog timer—

```
Router(config)#event manager event-trigger <event-name>
  type timer watchdog value <countdown-timer-value-in-seconds>
```

Cron timer—

```
Router(config)#event manager event-trigger <event-name>
  type timer cron cron-entry "<cron string>"
```

- Track:

```
Router(config)#event manager event-trigger <event-name>
  type track name <track-name> status {up | down | any}
```

- Telemetry:

Match criteria as exact-match—

```
Router(config)#event manager event-trigger <event-name>
  query json-path <query> match-criteria exact-match value <value>
  type telemetry sensor-path <telemetry-sensor-path>
  sample-interval <sample-interval-in-seconds>
```

Match criteria as threshold—

```
Router(config)#event manager event-trigger <event-name> query
  json-path <query> match-criteria threshold {equal-to | greater-equal-to |
  greater-than | less-equal-to | less-than | not-equal-to} <value>
  type telemetry sensor-path <telemetry-sensor-path> sample-interval <sample-interval-in-seconds>
```

Match criteria as rate—

```
Router(config)#event manager event-trigger <event-name>
  query json-path <query> match-criteria rate direction {any | decreasing | increasing}
  value {equal-to | greater-equal-to | greater-than | less-equal-to | less-than | not-equal-to}
  <value>
  type telemetry sensor-path <telemetry-sensor-path> sample-interval <sample-interval-in-seconds>>
```

## Example

**Example:** The following example shows the configuration for syslog event type. If severity is configured, the event gets triggered only if both the syslog severity and the syslog pattern match with the syslog generated on the router. If severity is not configured, it is set to `all`, where only pattern match is considered for the event to trigger.

```
Router(config)#event manager event-trigger eventT10
  type syslog pattern "<pattern-to-match>" severity <value>

Router(config)#event manager event-trigger eventT10
  rate-limit seconds <time-in-seconds>
  type syslog pattern "<pattern-to-match>" severity <value>
```

The severity values are:

```

alert      Syslog priority 1
critical   Syslog priority 2
debug      Syslog priority 7 (lowest)
emergency  Syslog priority 0 (highest)
error      Syslog priority 3
info       Syslog priority 6
notice     Syslog priority 5
warning    Syslog priority 4

```

The following example shows a syslog pattern `L2-BM-6-ACTIVE` with severity value `critical`:

```

Router(config)#event manager event-trigger eventT10
type syslog pattern "L2-BM-6-ACTIVE" severity info

```

The event gets triggered, if both the syslog pattern `L2-BM-6-ACTIVE` and severity value `info` match.

## Create Actions for Events

Define the actions that must be taken when an event occurs.

### Before you begin

Ensure that the following prerequisites are met before you configure the action:

- [Define Trigger Conditions for an Event, on page 4](#)

**Step 1** Set the event action.

#### Example:

```
Router(config)#event manager action action1
```

**Step 2** Define the type of action. For example, the action is a Python script.

#### Example:

```
Router(config)#event manager action action1 type script action1.py
```

**Step 3** Configure the maximum run time of the script for the event.

#### Example:

```
Router(config)#event manager action action1 type script action1.py maxrun seconds 30
```

The default value is 20 seconds.

**Step 4** Configure the checksum for the script. This configuration is mandatory. Every script is associated with a checksum hash value. This value ensures the integrity of the script, and that the script is not tampered. The checksum is a string of numbers and letters that act as a fingerprint for script.

a) Retrieve the SHA256 checksum hash value for the script from the IOS XR Linux bash shell.

#### Example:

```

Router#run
[node0_RP0_CPU0:~]$sha256sum /harddisk:/mirror/script-mgmt/eem/action1.py
407ce32678a5fc4b0ad49e83acad6453ad1d47e8dad9501cf139daa75d53e3dd
/harddisk:/mirror/script-mgmt/eem/action1.py

```

b) Configure the checksum for the script.

**Example:**

```
Router(config)#event manager action action1 type script action1.py checksum
sha256 407ce32678a5fc4b0ad49e83acad6453ad1d47e8dad9501cf139daa75d53e3dd
```

**Step 5** Enter the username for the script to execute.

**Example:**

```
Router(config)#event manager action action1 username eem_user
```

## Create a Policy Map of Events and Actions

*Table 1: Feature History Table*

Feature Name	Release Information	Description
Add Multiple Events In a Policy Map With a Single EEM Script	Release 7.5.1	With this feature, you can add multiple events to a policy map with boolean (AND or OR) correlation. EEM triggers the script when the correlation defined in the policy map for the events is true. Using EEM scripts, you can create a logical correlation of events in the policy map and configure multiple actions for detectors such as timer, object-tracking, and telemetry events via sensor path.

Create a policy to map events and actions. You can configure a policy that associates multiple actions with an event or use the same action with different events. The policy can be triggered if an event or multiple events occur at a specified number of times within a specified period of time. The `occurrence` and `period` are optional parameters. You can add multiple events to a policy-map with boolean (AND or OR) correlation. EEM triggers the script when correlation defined in the policy-map for the events is true. For example, a multi-event policy-map for `event1` and `event2` with `event1 AND event2` boolean operation is triggered only when both `event1` and `event2` are true.

**Before you begin**

Ensure that the following prerequisites are met before you create a policy map:

- [Define Trigger Conditions for an Event, on page 4](#)
- [Create Actions for Events, on page 7](#)

**Step 1** Create a policy map.

**Example:**

```
Router(config)#event manager policy-map policy1
```



```
Router(config)#event manager policy-map policy1
trigger multi-event ["(<event1> AND <event2>) AND (<event3> OR <event4>)" |
occurrence <count> | period <time in seconds>]
```

**Note** Ensure that the operations when configuring multiple events are within double quotes "".

where,

- **occurrence:** Specifies the number of times the total correlation occurs before an EEM event is raised. If occurrence is not specified, the policy-map gets triggered on every occurrence of the event. The occurrence value ranges from 1 to 32. An occurrence that is configured with multiple events is considered as only one occurrence if the boolean logic operations becomes true.
- **period:** Time interval in seconds, during which the event occurs. The period must be an integer number between 1 to 429496729 seconds.

**Step 2** Define the action that must be implemented when the event occurs. Maximum of 5 actions can be mapped to a policy map.

**Example:**

```
Router(config-policy-map)#action action1
```

**Step 3** Configure the name of the event or multiple events to trigger the policy-map.

**Example:**

```
Router(config-policy-map)#trigger event event10
```

The following example shows the policy-map for multiple events:

```
event manager policy-map policy001
trigger multi-event "event1 OR (event4 AND event2)"
period 60
action action2
occurrence 2
!
```

---

## View Operational Status of Event Scripts

Retrieve the operational status of events, actions and policy maps.

### Before you begin

Ensure that the following prerequisites are met before you trigger the event:

- [Define Trigger Conditions for an Event, on page 4](#)
- [Create Actions for Events, on page 7](#)
- [Create a Policy Map of Events and Actions, on page 8](#)

---

**Step 1** Run the **show event manager event-trigger all** command to view the summary of basic data of all events that are configured.

**Example:**

```
Router#show event manager event-trigger all
Tue Aug 24 14:47:35.803 IST
Thu May 20 20:41:03.690 UTC
No. Name      esid   Type   Occurs  Period  Trigger-Count  Policy-Count  Status
1  event1    1008  syslog  2       1800    4              1             active
2  event2    1009  syslog  2       1800    4              1             active
3  event3    1010  syslog  2       1800    4              1             active
4  event4    1011  syslog  2       1800    4              1             active
5  event5    1012  syslog  2       1800    4              1             active
6  event6    1013  syslog  2       1800    4              1             active
7  event7    1014  syslog  2       1800    4              1             active
8  event8    1015  syslog  2       1800    4              1             active
9  event9    1016  syslog  2       1800    4              1             active
```

Use the **show event manager event-trigger all detailed** command to view the details about the match criteria that you configured, severity level, policies mapped to the events and so on.

Use the **show event manager event-trigger <event-name> detailed** command to view the details about the individual events.

```
Router#show event manager event-trigger event1 detailed
Fri Nov 19 04:21:45.558 UTC
```

```
Event trigger name: event1
Event esid: 107
Event type: timer
Event occurrence: NA
Event period: NA
Event rate-limit: NA
Event triggered count: 12861
Event policy reg count: 1
Event status: active
Timer type: watchdog
Timer value: 10
```

```
Policy mapping info
1  event1                policy1
```

**Step 2** Run the **show event manager policy-map all** command to view the summary of all the configured policy maps.

#### Example:

```
Router#show event manager policy-map all
Tue Aug 24 14:48:52.153 IST
No. Name      Occurs  period  Trigger-Count  Status
1  policy1    NA      NA      1              active
2  policy2    NA      NA      1              active
3  policy3    NA      NA      1              active
4  policy4    NA      NA      1              active
```

Use the **show event manager policy-map all detailed** command to view the details about mapping of associated events and actions in the policy maps.

```
Router#show event manager policy-map policy1 all detailed
Fri Nov 19 11:35:40.282 UTC
```

```
Policy name: policy1
Policy occurrence: 3
Policy period: 120
Policy triggered count: 0
Policy status: active
Multi event policy: FALSE
```

```
Events mapped to the policy
No.      Name                Status
```

```

1          event2          active

Actions mapped to the policy
No.      Name              Checksum
1        action1          SHA256

```

Use the **show event manager policy-map <policy-map-name> detailed** command to view the details about the individual policy maps.

```

Router#show event manager policy-map policy1 detailed
Fri Nov 19 11:05:38.828 UTC

```

```

Policy name: policy1
Policy occurrence: 2
Policy period: 60
Policy triggered count: 0
Policy status: active
Multi event policy: TRUE
Multi event string : "event1 OR (event4 AND event2)"
Current Correlation State : FALSE

```

```

Events mapped to the policy
No.      Name              Status      Corr Status  Reset time(sec)
1        event1             active      0             0
2        event2             active      0             0
3        event4             active      0             0

```

```

Actions mapped to the policy
No.      Name              Checksum
1        action2          SHA256

```

**Step 3** Run the **show event manager action <action-name> detailed** command to view the details of an action.

**Example:**

```

Router#show event manager action action1 detailed
Tue Aug 24 16:05:44.298 UTC

```

```

Action name: action1
Action type: script
EEM Script name: event_script_1.py
Action triggered count: 1
Action policy count: 1
Username: eem_user
Checksum: 407ce32678a5fc4b0ad49e83acad6453ad1d47e8dad9501cf139daa75d53e3dd
Last execution status: Success

```

```

Policy mapping info
1  action1          policy1

```

Use the **show event manager action all** and **show event manager action all detailed** command to view the summary and details about all the configured actions.

## Example: Shut Inactive Bundle Interfaces Using EEM Script

In this example, you use an EEM event to look for a syslog message and trigger a Python script. The script does two things:

- Triggers an event on the interface inactive log as part of Bundle-Ether1, and shuts down the interface.

- Runs the **show tech-support bundles** command to collect debug data.

**Step 1** Create an eem script `event_script_action_bundle_shut.py`. Store the script on an HTTP server or copy the script to the harddisk of the router.

**Example:**

```
from iosxr.xrcli.xrcli_helper import *
from cisco.script_mgmt import xrlog

logger = xrlog.getScriptLogger('sample_script')
syslog = xrlog.getSysLogger('sample_script')
helper = XrcliHelper(debug = True)

syslog.info('Execution of event manager action script event_script_action_bundle_shut.py started')

config = """interface Bundle-Ether1
shutdown"""

cmd = "show tech-support bundles"

if __name__ == '__main__':
    res = helper.xr_apply_config_string(config)
    if res['status'] == 'success':
        syslog.info('OPS_EVENT_SCRIPT_ACTION : Configuration succeeded')
    else:
        syslog.error('OPS_EVENT_SCRIPT_ACTION : Configuration failed')

    res = helper.xrcli_exec(cmd)
    if res['status'] == 'success':
        syslog.info('OPS_EVENT_SCRIPT_ACTION : show tech started')
    else:
        syslog.error('OPS_EVENT_SCRIPT_ACTION : show tech failed')

    syslog.info('Execution of event manager action script event_script_action_bundle_shut.py ended')
```

**Step 2** Add the script from HTTP server or harddisk to the script management repository..

**Step 3** After the configured type matches the syslog pattern, the script is triggered in response to the detected event. You can view the running configuration for the event manager.

**Example:**

```
Router#show running-config event manager
Mon Aug 30 06:23:32.974 UTC
event manager action action1
  username eem_user
  type script script-name eem_script_bundle_shut.py maxrun seconds 600 checksum sha256
2386d8f71b2d6f6f6e77a7a39d3b4d38cca07f9eaf2a4de7cd40c1b027a4e248
!
event manager policy-map policy1
  trigger event event1
  action action1
!
event manager event-trigger event1
  type syslog pattern "%L2-BM-6-ACTIVE : FortyGigE0/0/0/13 is no longer Active as part of Bundle-Ether1"
!
```