# Manage Automation Scripts Using YANG RPCs

***Table 1: Feature History Table***

| Feature Name | Release Information | Description |
|---|---|---|
| Manage Automation Scripts Using YANG RPCs | Release 7.3.2 | This feature enables you to use remote procedure calls (RPCs) on YANG data models to perform the same automated operations as CLIs, such as edit configurations or retrieve router information. |

You can use automation scripts to interact with the router using NETCONF, helper modules or gNMI python modules.

An SSH session must be established between the client and the server to run RPCs on a device. The client can be a script or application that runs as part of a network manager. The server is a network device such as a router. To enable the NETCONF SSH agent, use the following commands:

```
ssh server v2
netconf agent tty
```

After a NETCONF session is established, the client sends one or more RPC requests to the server. The server processes the requests and sends an RPC response back to the client. For example, the get-config operation retrieves the configuration of the device and the edit-config operation edits the configuration on the device.

For more information about data models and how to use the models


- Manage Common Script Actions Using YANG RPCs, on page 2
- Manage Exec Scripts Using RPCs, on page 4
- Manage EEM Script Using RPCs, on page 8

# Manage Common Script Actions Using YANG RPCs

*Table 2: Feature History Table*

| Feature Name | Release Information | Description |
|---|---|---|
| Manage Common Script Actions Using YANG RPCs | Release 7.5.1 | This feature enables you to use YANG remote procedure calls (RPCs) on `Cisco-IOS-XR-infra-script-mgmt-act.yang` data model to perform actions on the automation scripts such as add or remove script from the script repository, run, or stop script from running. |

This section provides information about YANG RPC messages for common actions on automation scripts. The `Cisco-IOS-XR-infra-script-mgmt-act.yang` action YANG model is enhanced to perform the actions such as adding or removing a script from the repository, and also include output responses. The output response provides a description about the action and displays the status as `True` for a successful action, and `False` for a failed action.

The YANG RPC supports these scripts:

- Config
- Exec
- Process
- EEM

The following section shows the various script actions, sample RPC request, and RPC response.

### Add Script

You can add up to a maximum of 10 scripts to the script repository. You can set the script type to config, exec, process, or eem. The following example shows the RPC to add the exec script to the repository:

```
<add xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-infra-script-mgmt-act">
<script-type>exec</script-type>
<vrf></vrf>
<source>/harddisk:/</source>
<script-name>sample.py</script-name>
</add>
```

You can add more than one script to the repository simultaneously:

```
<add xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-infra-script-mgmt-act">
<script-type>exec</script-type>
<source>/harddisk:/</source>
<script-name>sample2.py</script-name>
<script-name>sample3.py</script-name>
</add>
```

To add a checksum value to the script, use the following RPC request:

```
<add-checksum xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-infra-script-mgmt-act">
<script-type>exec</script-type>
<source>/harddisk:/</source>
<script-checksums>
    <script-name>sample.py</script-name>
<checksum>e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855</checksum>
</script-checksums>
</add-checksum>
```

You can add more than one script with their checksum values:

```
<add-checksum xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-infra-script-mgmt-act">
<script-type>exec</script-type>
<source>/harddisk:/</source>
<script-checksums>
    <script-name>sample.py</script-name>
<checksum>e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855</checksum>
</script-checksums>
<script-checksums>
    <script-name>sample2.py</script-name>
<checksum>e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855</checksum>
</script-checksums>
</add-checksum>
```

### Remove Script

To remove script from the repository, provide the script type and the script name. You can send an RPC request to remove up to 10 scripts.

```
<remove xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-infra-script-mgmt-act">
<script-type>exec</script-type>
<script-name>sample.py</script-name>
</remove>
```

You can remove more than one script simultaneously:

```
<remove xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-infra-script-mgmt-act">
<script-type>exec</script-type>
<script-name>sample2.py</script-name>
<script-name>sample3.py</script-name>
</remove>
```

The following example shows a sample RPC response indicating that the script `sample1.py` is removed from the repository:

```
<responses>
<script-name>sample.py<script-name>
 <response>sample.py has been removed from the script repository</response>
<status>True</status>
<responses>
```

### Stop Script

You must provide the request ID for the script instance to be stopped.

```
<stop xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-infra-script-mgmt-act">
<request-id>1622058854</request-id>
<description></description>
</stop>
```

The following example shows that the script has stopped:

```
<script-stop-response>
           <response></response>
```

```
           <status>True</status>
</script-stop-response>
```

### Run Script

You must provide the script name to run the script. You can also configure the log levels to one of these values—Critical, Debug, Error, Info, or Warning.

```
<run xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-infra-script-mgmt-act">
    <script-name>sample.py</script-name>
    <argument-list></ argument-list>
    <description></description>
    <log-level></log-level>
    <log-path></log-path>
    <max-runtime></max-runtime>
</run>
```

The following example shows a sample RPC response where the script with the request ID `1622058854` is run:

```
<script-run-response>
 <response>Script run scheduled</response>
 <request-id>1622058854</request-id>
 <status>True</status>
</script-run-response>
```

# Manage Exec Scripts Using RPCs

The following data models support exec scripts:

- Edit or get configuration—Cisco-IOS-XR-infra-script-mgmt-cfg.yang
- Perform action—Cisco-IOS-XR-infra-script-mgmt-act.yang
- Retrieve operational data—Cisco-IOS-XR-infra-script-mgmt-oper.yang

This section provides examples of using RPC messages on exec scripts, and also the YANG data model and equivalent CLI command to perform the tasks:

### Add Script

You use data model to add an exec script from an external repository to the `harddisk:/mirror/script-mgmt/exec` script management repository on the router.

| YANG Data Model | Equivalent CLI |
|---|---|
| Cisco-IOS-XR-infra-script-mgmt-act.yang | **script add exec** *script-location script.py*<br>See. |

RPC Request:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <script-add-type-source xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-infra-script-mgmt-act">
    <type>exec</type>
    <source>/harddisk:</source>
    <file-name-1>sample1.py</file-name-1>
  </script-add-type-source>
</rpc>
```

Syslog:

```
Router: script_manager[66762]: %OS-SCRIPT_MGMT-6-INFO :
Script-script_manager: sample1.py has been added to the script repository
```

### Configure Checksum

Every script is associated with a checksum value for integrity. You can configure the checksum using data models.

| YANG Data Model | Equivalent CLI |
|---|---|
| Cisco-IOS-XR-infra-script-mgmt-act.yang | **script exec** *sample1.py* **checksum SHA256** *checksum-value*<br>See, . |

RPC Request:

```
<rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
message-id="urn:uuid:16fa22ed-3f46-4369-806a-3bccd1aefcaf">
  <nc:edit-config>
    <nc:target>
      <nc:candidate/>
    </nc:target>
    <nc:config>
      <scripts xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-infra-script-mgmt-cfg">
        <exec-script>
          <scripts>
            <script>
              <script-name>sample1.py</script-name>
              <checksum>
                <checksum-type>sha256</checksum-type>

<checksum>5103a843032505decc37ff21089336e4bcc6a1061341056ca8add3ac5d6620ef</checksum>
              </checksum>
            </script>
          </scripts>
        </exec-script>
      </scripts>
    </nc:config>
  </nc:edit-config>
</nc:rpc>
```

RPC Response:

```
<?xml version="1.0" ?>
<rpc-reply message-id="urn:uuid:16fa22ed-3f46-4369-806a-3bccd1aefcaf"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

### Run Script

| YANG Data Model | Equivalent CLI |
|---|---|
| Cisco-IOS-XR-infra-script-mgmt-act.yang | **script run** *sample1.py* |

RPC Request:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <script-run xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-infra-script-mgmt-act">
    <name>sample1.py</name>
  </script-run>
</rpc>
```

RPC Response:

```
<?xml version="1.0" ?>
<rpc-reply message-id="urn:uuid:d54247c7-cf29-42f2-bfb8-517d6458f77c" xmlns="urn:ietf:
params:xml:ns:netconf:base:1.0" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

Syslog:

```
Router: UTC: script_control_cli[67858]: %OS-SCRIPT_MGMT-6-INFO : Script-control:
Script run scheduled: sample1.py. Request ID: 1631795207
Router: script_agent_main[248]: %OS-SCRIPT_MGMT-6-INFO : Script-script_agent: Script
execution sample1.py (exec) Started : Request ID : 1631795207 :: PID: 18710
```

### Stop Script

| YANG Data Model | Equivalent CLI |
|---|---|
| Cisco-IOS-XR-infra-script-mgmt-act.yang | **script stop** *value [short-decription]* |

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <script-stop-request xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-infra-script-mgmt-act">
    <request>1614930988</request>
  </script-stop-request>
</rpc>
```

### Remove Script

You can remove scripts from the script management repository. The data about script management and execution history is not deleted when the script is removed.

| YANG Data Model | Equivalent CLI |
|---|---|
| Cisco-IOS-XR-infra-script-mgmt-act.yang | **script remove exec** *script.py*<br>See,. |

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <script-remove-type xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-infra-script-mgmt-act">
    <type>exec</type>
    <file-name-1>load_modules_ut.py</file-name-1>
  </script-remove-type>
</rpc>
```

### Show Script Execution

View the status of the script execution.

| YANG Data Model | Equivalent CLI |
|---|---|
| Cisco-IOS-XR-infra-script-mgmt-oper.yang | **show script execution** *[request-id <value>] [name <filename>] [status {Exception \| Executed \| Killed \| Started \| Stopped \| Timed-out}] [reverse] [last <number>]* |

RPC Request:

```
-------------- Sent to NETCONF Agent ---------------
<rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
message-id="urn:uuid:7fd0d184-0004-4a51-9765-d29bc94c793b">
  <get>
    <filter>
      <script xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-infra-script-mgmt-oper">
        <execution>
          <requests>
            <request>
              <request-id>1631795207</request-id>
              <detail>
                <execution-detail/>
              </detail>
            </request>
          </requests>
        </execution>
      </script>
    </filter>
  </get>
</rpc>
```

RPC Response:

```
----------------- Received from NETCONF agent --------------------
<?xml version="1.0" ?>
<rpc-reply message-id="urn:uuid:7fd0d184-0004-4a51-9765-d29bc94c793b"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <script xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-infra-script-mgmt-oper">
      <execution>
        <requests>
          <request>
            <request-id>1631795207</request-id>
            <detail>
              <execution-detail>
                <execution-summary>
                  <request-id>1631795207</request-id>
                  <return-val>0</return-val>
                  <script-type>exec</script-type>
                  <script-name>sample1.py</script-name>
                  <duration>60.65s</duration>
                  <event-time>Thu Sep 16 12:26:46 2021</event-time>
                  <status>Executed</status>
                </execution-summary>
                <execution-detail>

<log-path>/harddisk:/mirror/script-mgmt/logs/sample1.py_exec_1631795207</log-path>
                  <run-options>Logging level - INFO, Max. Runtime - 300s, Mode -
Background</run-options>
                </execution-detail>
                <execution-event>
                  <description>None</description>
                  <duration>0.00s</duration>
```

```
                <event>New</event>
                <time>Thu Sep 16 12:26:46 2021</time>
              </execution-event>
              <execution-event>
                <description>Script execution started. PID (18710)</description>
                <duration>0.03s</duration>
                <event>Started</event>
                <time>Thu Sep 16 12:26:46 2021</time>
              </execution-event>
              <execution-event>
                <description>Script execution complete</description>
                <duration>60.65s</duration>
                <event>Executed</event>
                <time>Thu Sep 16 12:27:47 2021</time>
              </execution-event>
            </execution-detail>
          </detail>
        </request>
      </requests>
    </execution>
  </script>
 </data>
</rpc-reply>
```

# Manage EEM Script Using RPCs

The following data model supports eem scripts:

- Edit configuration—Cisco-IOS-XR-um-event-manager-policy-map-cfg.yang

The model is augmented to `Cisco-IOS-XR-um-event-manager-cfg.yang` data model.

This section provides examples of using RPC messages on eem scripts, and also the YANG data model and equivalent CLI command to perform the tasks:

### Define Actions for Events Using Data Model

You use data model to create actions for events.

| YANG Data Model | Equivalent CLI |
|---|---|
| Cisco-IOS-XR-um-event-manager-policy-map-cfg | **event manager event-trigger** *event-name*<br>**occurance** *value*<br>**period seconds** *value*<br>**period seconds** *value***type syslog pattern** *"syslog-pattern"* **severity** *syslog-severity*<br>See<br>**event manager action** *action-name*<br>**username** *username*<br>**type script script-name** *python-script-name.py* **maxrun seconds** *value* **checksum sha256** *checksum-value*<br>See. |

RPC Request:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <edit-config>
   <target>
    <candidate/>
   </target>
   <config>
    <event xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-um-event-manager-cfg">
     <manager>
      <event-trigger
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-um-event-manager-policy-map-cfg">
       <event>
        <event-name>event_1</event-name>
        <occurrence>2</occurrence>
        <period>
         <seconds>60</seconds>
        </period>
        <type>
         <syslog>
          <pattern>"Syslog for EEM script"</pattern>
          <severity>
           <warning/>
          </severity>
         </syslog>
        </type>
       </event>
      </event-trigger>
   <actions xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-um-event-manager-policy-map-cfg">

      <action>
       <action-name>action_1</action-name>
        <type>
         <script>
          <script-name>event_script_1.py</script-name>
          <maxrun>
           <seconds>30</seconds>
          </maxrun>
          <checksum>
         <sha256>bb19a7a286db72aa7c7bd75ad5f224eea1062b7cdaaeee06f11f0f86f976831d</sha256>
```

```
          </checksum>
         </script>
        </type>
       <username>eem_user_1</username>
      </action>
     </actions>
    </manager>
   </event>
  </config>
 </edit-config>
</rpc>
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="102">
<commit>
</rpc>
```

RPC Response:

```
<?xml version="1.0" ?>
<rpc-reply message-id="urn:uuid:16fa22ed-3f46-4369-806a-3bccd1aefcaf"
xmlns="urn:ietf:params:xml:ns:
netconf:base:1.0" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

### Create Policy Map for Events and Actions Using Data Model

You use data model to create actions for events.

| YANG Data Model | Equivalent CLI |
|---|---|
| Cisco-IOS-XR-um-event-manager-policy-map-cfg | **event manager policy-map** *policy-name*<br>**action** *action-name*<br>**trigger event** *event-name*<br>See, . |

RPC Request:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
 <edit-config>
 <target>
  <candidate/>
  </target>
 <config>
 <event xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-um-event-manager-cfg">
 <manager>
  <policy-maps xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-um-event-manager-policy-map-cfg">

     <policy-map>
      <policy-map-name>policy_1</policy-map-name>
      <trigger>
       <event>event_1</event>
      </trigger>
      <actions>
      <action>
        <action-name>action_1</action-name>
      </action>
     </actions>
    </policy-map>
   </policy-maps>
   </manager>
```

```
  </config>
  </edit-config>
</rpc>

<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="102">
  <commit/>
</rpc>
```

RPC Response:

```
<?xml version="1.0" ?>
<rpc-reply message-id="urn:uuid:16fa22ed-3f46-4369-806a-3bccd1aefcaf"
xmlns="urn:ietf:params:xml:ns:
netconf:base:1.0" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

# Operational Model for EEM Script

***Table 3: Feature History Table***

| Feature Name | Release Information | Description |
|---|---|---|
| Operational Data Model for EEM Script | Release 7.5.2 | You can programmatically retrieve the operational status of events, actions, and policy maps using the YANG data model.<br>In earlier releases, you used the `show event manager` command to view the operational status of event scripts.<br>This release introduces `Cisco-IOS-XR-ha-eem-policy-oper.yang` and `Cisco-IOS-XR-event-manager-policy-map-oper.yang` data models. |

### Operational Data Model to Retrieve Actions

You use data model to view the details of an action. IOS XR actions are RPC statements that trigger an operation or execute a command on the router. This action is executed when the router receives the corresponding NETCONF RPC request. Once the router executes an action, it replies with a NETCONF RPC response.

| YANG Data Model | Equivalent Command |
|---|---|
| Cisco-IOS-XR-ha-eem-policy-oper | **show event manager action** *action-name* **detailed**<br>See, View Operational Status of Event Scripts. |

RPC Request:

```
<rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
message-id="urn:uuid:62b9e81b-5d9e-44f6-8a5d-d193a0f8b3d3">
  <get>
    <filter>
      <eem xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ha-eem-policy-oper">
        <action-names
```

```
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-event-manager-policy-map-oper">
          <action-name>
            <action-name>action2</action-name>
          </action-name>
        </action-names>
      </eem>
    </filter>
  </get>
</rpc>
```

RPC Response:

```
<?xml version="1.0" ?>
<rpc-reply message-id="urn:uuid:62b9e81b-5d9e-44f6-8a5d-d193a0f8b3d3"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <eem xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ha-eem-policy-oper">
      <action-names
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-event-manager-policy-map-oper">
        <action-name>
          <action-name>action2</action-name>
          <action-name-xr>action2</action-name-xr>
          <script-name>event_script_2.py</script-name>
          <action-type>script</action-type>
          <triggered-count>7</triggered-count>
          <policy-count>1</policy-count>
          <max-run>20</max-run>
          <checksum-enabled>SHA256</checksum-enabled>
          <last-run-status>Success</last-run-status>
          <user-name>eem_user</user-name>

<checksum-string>270b9730e77c9bd6f5784084ed21e29d8d7b8edaf8f98a4513879a1631c493ad</checksum-string>

          <action-policy-map>
            <policy-name>policy3</policy-name>
          </action-policy-map>
        </action-name>
      </action-names>
    </eem>
  </data>
</rpc-reply>
```

### Operational Data Model to Retrieve Policy Map

You use data model to view the details of a policy map.

| YANG Data Model | Equivalent Command |
|---|---|
| Cisco-IOS-XR-ha-eem-policy-oper | **show event manager policy-map** *policy-name* **detailed**<br><br>See, View Operational Status of Event Scripts. |

RPC Request:

```
<rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
message-id="urn:uuid:3cec3f3a-395b-4763-b1a1-1053149da60c">
  <get>
    <filter>
      <eem xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ha-eem-policy-oper">
        <policy-map-names
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-event-manager-policy-map-oper">
```

```
          <policy-map-name>
            <policy-name>policy4</policy-name>
          </policy-map-name>
        </policy-map-names>
      </eem>
    </filter>
  </get>
</rpc>
```

RPC Response:

```
<?xml version="1.0" ?>
<rpc-reply message-id="urn:uuid:3cec3f3a-395b-4763-b1a1-1053149da60c"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <eem xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ha-eem-policy-oper">
      <policy-map-names
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-event-manager-policy-map-oper">
       <policy-map-name>
         <policy-name>policy4</policy-name>
          <policy-name-xr>policy4</policy-name-xr>
          <policy-status>active</policy-status>
          <policy-occurrence>2</policy-occurrence>
          <policy-period>30</policy-period>
          <policy-triggered-count>0</policy-triggered-count>
          <event-count>2</event-count>
          <action-count>1</action-count>
          <policy-event-map>
            <event-name>event5</event-name>
            <event-status>active</event-status>
            <corr-status>false</corr-status>
            <reset-time>0</reset-time>
          </policy-event-map>
          <policy-event-map>
            <event-name>event4</event-name>
            <event-status>active</event-status>
            <corr-status>false</corr-status>
            <reset-time>0</reset-time>
          </policy-event-map>
          <policy-action-map>
            <action-name>action4</action-name>
            <checksum-enabled>SHA256</checksum-enabled>
          </policy-action-map>
          <multi-event-policy>true</multi-event-policy>
          <current-correlation-state>false</current-correlation-state>
          <multi-event-string>&quot;event4 AND event5&quot;</multi-event-string>
        </policy-map-name>
      </policy-map-names>
    </eem>
  </data>
 </rpc-reply>
```

### Operational Data Model to Retrieve Events With Trigger Conditions

You use data model to view the details of a event-trigger conditions.

| YANG Data Model | Equivalent CLI |
|---|---|
| Cisco-IOS-XR-ha-eem-policy-oper | show event manager event-trigger *event-trigger-name* **detailed**<br><br>See, View Operational Status of Event Scripts. |

RPC Request:

```
<rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
message-id="urn:uuid:77229832-1a44-47e4-b0cf-2c2066ac579a"><nc:get>
    <filter>
      <eem xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ha-eem-policy-oper">
        <event-trigger-names
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-event-manager-policy-map-oper">
          <event-trigger-name>
            <event-name>event4</event-name>
          </event-trigger-name>
        </event-trigger-names>
      </eem>
    </filter>
  </get>
</rpc>
```

RPC Response:

```
<?xml version="1.0" ?>
<rpc-reply message-id="urn:uuid:77229832-1a44-47e4-b0cf-2c2066ac579a"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <eem xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ha-eem-policy-oper">
      <event-trigger-names
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-event-manager-policy-map-oper">
        <event-trigger-name>
          <event-name>event4</event-name>
          <event-name-xr>event4</event-name-xr>
          <event-status>active</event-status>
          <event-type>syslog</event-type>
          <eventesid>16</eventesid>
          <event-occurrence>NA</event-occurrence>
          <event-period>NA</event-period>
          <rate-limit>0</rate-limit>
          <event-triggered-count>2</event-triggered-count>
          <event-policy-reg-count>1</event-policy-reg-count>
          <event-policy-map>
            <policy-name>policy4</policy-name>
          </event-policy-map>
          <event-syslog-info>
            <pattern>%PKT_INFRA-LINK-3-UPDOWN : Interface GigabitEthernet0/0/0/4, changed
state to Down</pattern>
            <severity>ALL</severity>
          </event-syslog-info>
          <event-timer-info>
            <wd-info>
              <timer-value>0</timer-value>
            </wd-info>
          </event-timer-info>
          <event-telemetry-info>
            <sample-interval>0</sample-interval>
          </event-telemetry-info>
        </event-trigger-name>
      </event-trigger-names>
    </eem>
  </data>
</rpc-reply>
```