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 Exec Scripts Using RPCs, on page 1
- Manage EEM Script Using RPCs, on page 5

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* 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* 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* 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* **occurance** *value* **period seconds** *value* **period seconds** *value***type syslog pattern** *"syslog-pattern"* **severity** *syslog-severity* See **event manager action** *action-name* **username** *username* **type script script-name** *python-script-name.py* **maxrun seconds** *value* **checksum sha256** *checksum-value* 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* **action** *action-name* **trigger event** *event-name* 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>
```