



Process Script

- [Process scripts, on page 1](#)
- [Run the process script, on page 1](#)
- [Manage actions on process script , on page 10](#)

Process scripts

A process script is a persistent automation script that

- runs continuously on Cisco IOS XR as long as it is activated,
- is managed by the AppMgr, which controls startup, monitoring, and restart, and
- registers as an application instance for operational monitoring and status retrieval.

Process scripts, also referred to as daemon scripts, are designed to execute on Cisco IOS XR platforms without manual intervention. The AppMgr is responsible for starting, stopping, monitoring, and retrieving the status of these scripts. AppMgr ensures that scripts automatically restart if they terminate depending on the configured restart policy and manages startup dependencies with other processes on the router.

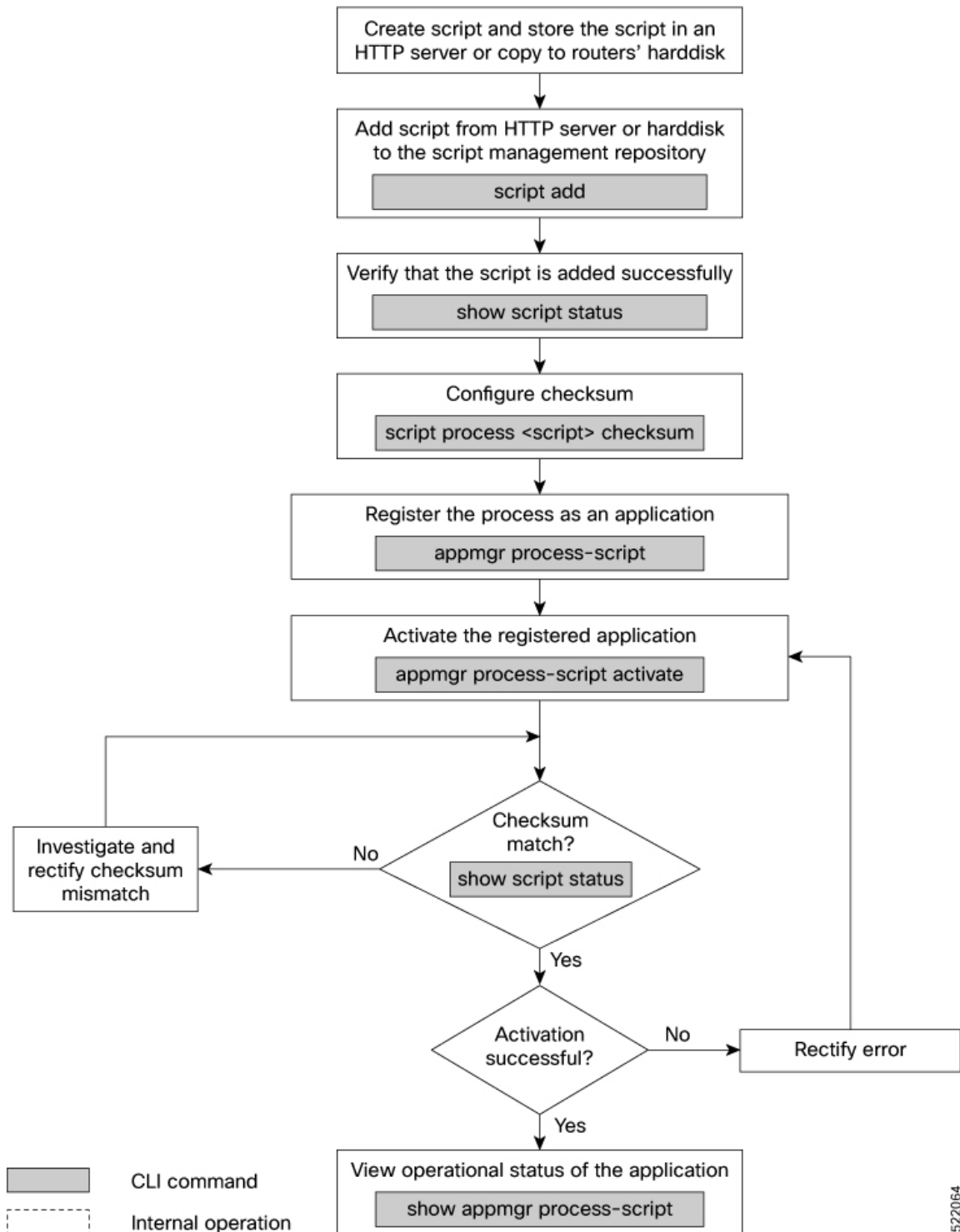
Process scripts support Python 3.5 as their programming language. For information on supported Python packages, refer to Cisco IOS XR Python Packages.

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

Run the process script

Run a process script to automate a specific set of system tasks.

Figure 1: Workflow of Process script



Before you begin

522064

Procedure

- Step 1** [Download the script.](#)
- Step 2** [Configure the checksum.](#)
- Step 3** [Register the script.](#)
- Step 4** [Activate the script.](#)
- Step 5** View the script execution details.

What to do next

Download the script to the router

Add a script to the script management repository on the router to enable script execution and management.

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.

Table 1: Location to download the script

Script Type	Download Location
config	harddisk:/mirror/script-mgmt/config
exec	harddisk:/mirror/script-mgmt/exec
process	harddisk:/mirror/script-mgmt/process
eem	harddisk:/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 harddisk using **scp** or **copy** command

Before you begin

- Confirm you have access to the script file(s) and to the router's CLI.
- If copying from an external repository, ensure required network connectivity and protocol support (HTTP, HTTPS, FTP, TFTP, SCP).

Procedure

- 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 any server or the harddisk location in the router.

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

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

```
Router#script add process http://192.0.2.0/scripts process-script.py
process-script.py has been added to the script repository
```

The `script add process` supports the HTTP, HTTPS, FTP, TFTP, and SCP protocols for copying a script. You can add a maximum of 10 scripts simultaneously.

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

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

```
Router#script add process 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.

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

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

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

```
Router#script add process /harddisk:/ process-script.py
process-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
```

```
=====
Name                | Type      | Status          | Last Action | Action Time
-----
process-script.py   | process   | Config Checksum | NEW         | Tue Aug 24 10:44:53 2021
=====
```

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

Configure checksum for the process script

Ensure script integrity by configuring a checksum that verifies the script has not been tampered with before execution.

Each process script is associated with a SHA256 checksum value, which acts as a fingerprint to ensure the script's integrity. If the configured checksum does not match the calculated value during script execution, the script will not run, and a warning is displayed. Configuring the checksum is mandatory for running the script.

It is mandatory to configure the checksum to run the script.



Note Process scripts support the SHA256 checksum hash.

Before you begin

Ensure that the script is added to the script management repository. See [Download the Script to the Router](#) for detailed instructions.

Procedure

Step 1 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/process/process-script.py
94336f3997521d6e1aec0ee6faab0233562d53d4de7b0092e80b53caed58414b
/harddisk:/mirror/script-mgmt/process/process-script.py
```

Make note of the checksum value.

Step 2 View the status of the script.

Example:

```
Router#show script status detail
=====
Name                | Type      | Status          | Last Action | Action Time
-----
process-script.py   | process   | Config Checksum | NEW         | Fri Aug 20 05:03:41 2021
-----
Script Name         : process-script.py
History:
-----
1.  Action          : NEW
    Time            : Fri Aug 20 05:03:41 2021
    Description      : User action IN_CLOSE_WRITE
=====
```

The status shows that the checksum is not configured.

Step 3 Configure the checksum.

Example:

```
Router#configure
Router(config)#script process process-script.py checksum SHA256
94336f3997521d6e1aec0ee6faab0233562d53d4de7b0092e80b53caed58414b
```

```
Router(config)#commit
Router(config)#end
```

Step 4 Verify the status of the script.

Example:

```
Router#show script status detail
```

```
=====
Name                | Type      | Status   | Last Action | Action Time
-----
process-script.py   | process   | Ready    | NEW         | Fri Aug 20 05:20:41 2021
-----

Script Name       : process-script.py
Checksum          : 94336f3997521d6e1aec0ee6faab0233562d53d4de7b0092e80b53caed58414b
History:
-----
1.  Action        : NEW
    Time          : Fri Aug 20 05:20:41 2021
    Checksum       : 94336f3997521d6e1aec0ee6faab0233562d53d4de7b0092e80b53caed58414b
    Description    : User action IN_CLOSE_WRITE
```

The status `Ready` indicates that the checksum is configured and the script is ready to be run. When the script is run, the checksum value is recalculated to check if it matches with the configured hash value. If the values differ, the script fails. It is mandatory for the checksum values to match for the script to run.

Register the process script as an application

Enable and manage a process script on your router by registering it as an application, ensuring it runs under the app manager and applies the desired execution settings.

Registering the process script with the app manager is a required step before you can use the script on the router. This process allows you to define application parameters and control restart policies for continuous operation or error handling.

Before you begin

Ensure that the following prerequisites are met before you register the script:

- [Download the Script to the Router](#)
- [Configure Checksum for Process Script](#)

Procedure

Step 1 Register the script with an application (instance) name in the app manager.

Example:

```
Router#configure
Router(config)#appmgr process-script my-process-app
Router(config-process)#executable process-script.py
```

Here, `my-process-app` is the application for the executable `process-script.py` script.

Step 2 Provide the arguments for the script.

Example:

```
Router(config-process)#run-args --host <host-name> --runtime 3 --log script
```

Step 3

Set a restart policy for the script if there is an error.

Example:

```
Router(config-process)#restart on-failure max-retries 3
Router(config-process)#commit
```

Here, the maximum attempts to restart the script is set to 3. After 3 attempts, the script stops.

Table 2: Options to restart the process

Keyword	Description
always	Always restart automatically. If the process exits, a scheduler queues the script and restarts the script. Note This is the default restart policy.
never	Never restart automatically. If the process exits, the script is not rerun unless you provide an action command to invoke the process.
on-failure	Restart on failure automatically. If the script exits successfully, the script is not scheduled again.
unless-errored	Restart script automatically unless errored.
unless-stopped	Restart script automatically unless stopped by the user using an action command.

Step 4

View the status of the registered script.

Example:

```
Router#show appmgr process-script-table
```

Name	Executable	Activated	Status	Restart Policy	Config Pending
my-process-app	process-script.py	No	Not Started	On Failure	No

The script is registered but is not active.

Activate the process script

Activate the process script that you registered with the app manager.

Before you begin

Ensure that the following prerequisites are met before you run the script:

- [Download the Script to the Router](#)
- [Configure Checksum for Process Script](#)
- [Register the Process Script as an Application](#)

Procedure

Step 1 Activate the process script.

Example:

```
Router#appmgr process-script activate name my-process-app
```

The instance `my-process-app` is activated for the process script.

Step 2 View the status of the activated script.

Example:

```
Router#show appmgr process-script-table
```

Name	Executable	Activated	Status	Restart Policy	Config Pending
my-process-app	process-script.py	Yes	Running	On Failure	No

The process script is activated and running.

Note

You can modify the script while the script is running. However, for the changes to take effect, you must deactivate and activate the script again. Until then, the configuration changes are pending. The status of the modification is indicated in the `Config Pending` option. In the example, value `No` indicates that there are no configuration changes that must be activated.

Obtain operational data and logs

Retrieve and review operational data and logs for a process script on the router.

Use this task when you need to monitor process script health, analyze performance, or troubleshoot errors on your router. This ensures accurate status information and supports problem resolution.

Before you begin

Ensure that the following prerequisites are met before you obtain the operational data:

- [Download the Script to the Router](#)
- [Configure Checksum for Process Script](#)
- [Register the Process Script as an Application](#)
- [Activate the Process Script](#)

Procedure

Step 1 View the registration information, pending configuration, execution information, and run time of the process script.

Example:


```
Router#show appmgr process-script my-process-app info
Application: my-process-app

Registration info:
  Executable           : process-script.py
  Run arguments        : --host <host-name> --runtime 3 --log script
  Restart policy       : On Failure
  Maximum restarts     : 3

Pending Configuration:
  Run arguments        : --host <host-name> --runtime 3 --log script
  Restart policy       : Always

Execution info and status:
  Activated           : Yes
  Status              : Running
  Executable Checksum : 94336f3997521d6e1aec0ee6faab0233562d53d4de7b0092e80b53caed58414b

  Last started time   : Fri Aug 20 06:20:21.947
  Restarts since last activate : 0/3
  Log location        :
/harddisk:/mirror/script-mgmt/logs/process-script.py_process_my-process-app
  Last exit code      : 1
```

Step 2 View the logs for the process scripts. App manager shows the logs for errors and output.

The following example shows the output logs:

Example:

```
Router#show appmgr process-script my-process-app logs output
[2021-08-20 06:20:55,609] INFO [sample-process]:: Beginning execution of process..
[2021-08-20 06:20:55,609] INFO [sample-process]:: Connecting to host '<host-name>'
[2021-08-20 06:20:56,610] INFO [sample-process]:: Reading database..
[2021-08-20 06:20:58,609] INFO [sample-process]:: Listening for requests..
```

The following example shows the error logs with errors:

Example:

```
Router#show appmgr process-script my-process-app logs errors
-----Run ID:1632914459  Fri Aug 20 06:30:20 2021-----
Traceback (most recent call last):
  File "/harddisk:/mirror/script-mgmt/process/process-script.py", line 121, in <module>
    main(args)
  File "/harddisk:/mirror/script-mgmt/process/process-script.py", line 97, in main
    printer()
  File "/harddisk:/mirror/script-mgmt/process/process-script.py", line 37, in wrapper
    result = func(*args, **kwargs)
  File "/harddisk:/mirror/script-mgmt/process/process-script.py", line 88, in printer
    time.sleep(1)
  File "/harddisk:/mirror/script-mgmt/process/process-script.py", line 30, in _handle_timeout
    raise TimeoutError(error_message)
__main__.TimeoutError: Timer expired
-----Run ID:1632914460  Fri Aug 20 06:31:03 2021-----
```

This example shows the log without errors:

Example:

```
Router#show appmgr process-script my-process-app logs errors
-----Run ID:1624346220  Fri Aug 20 10:46:44 2021-----
-----Run ID:1624346221  Fri Aug 20 10:47:50 2021-----
-----Run ID:1624346222  Fri Aug 20 10:52:39 2021-----
-----Run ID:1624346223  Fri Aug 20 10:53:45 2021-----
-----Run ID:1624346224  Fri Aug 20 11:07:17 2021-----
```

```

-----Run ID:1624346225 Fri Aug 20 11:08:23 2021-----
-----Run ID:1624346226 Fri Aug 20 11:09:29 2021-----
-----Run ID:1624346227 Fri Aug 20 11:10:35 2021-----
-----Run ID:1624346228 Fri Aug 20 11:11:41 2021-----

```

Manage actions on process script

The process script runs as a daemon continuously. You can, however, perform these actions on the process script and its application.

Table 3: Supported actions for process script applications

Action	Description
Deactivate	Clears all the resources that the application uses. <pre>Router#appmgr process-script deactivate name my-process-app</pre> You can modify the script while the script is running. However, for the changes to take effect, you must deactivate and activate the script again. Until then, the configuration changes do not take effect.
Kill	Terminates the script if the option to stop the script is unresponsive. <pre>Router#appmgr process-script kill name my-process-app</pre>
Restart	Restarts the process script. <pre>Router#appmgr process-script restart name my-process-app</pre>
Start	Starts an application that is already registered and activated with the app manager. <pre>Router#appmgr process-script start name my-process-app</pre>
Stop	Stops an application that is already registered, activated, and is currently running. Only the application is stopped; resources that the application uses is not cleared. <pre>Router#appmgr process-script stop name my-process-app</pre>