Exec Scripts

Exec scripts

Exec scripts are on-box scripts that

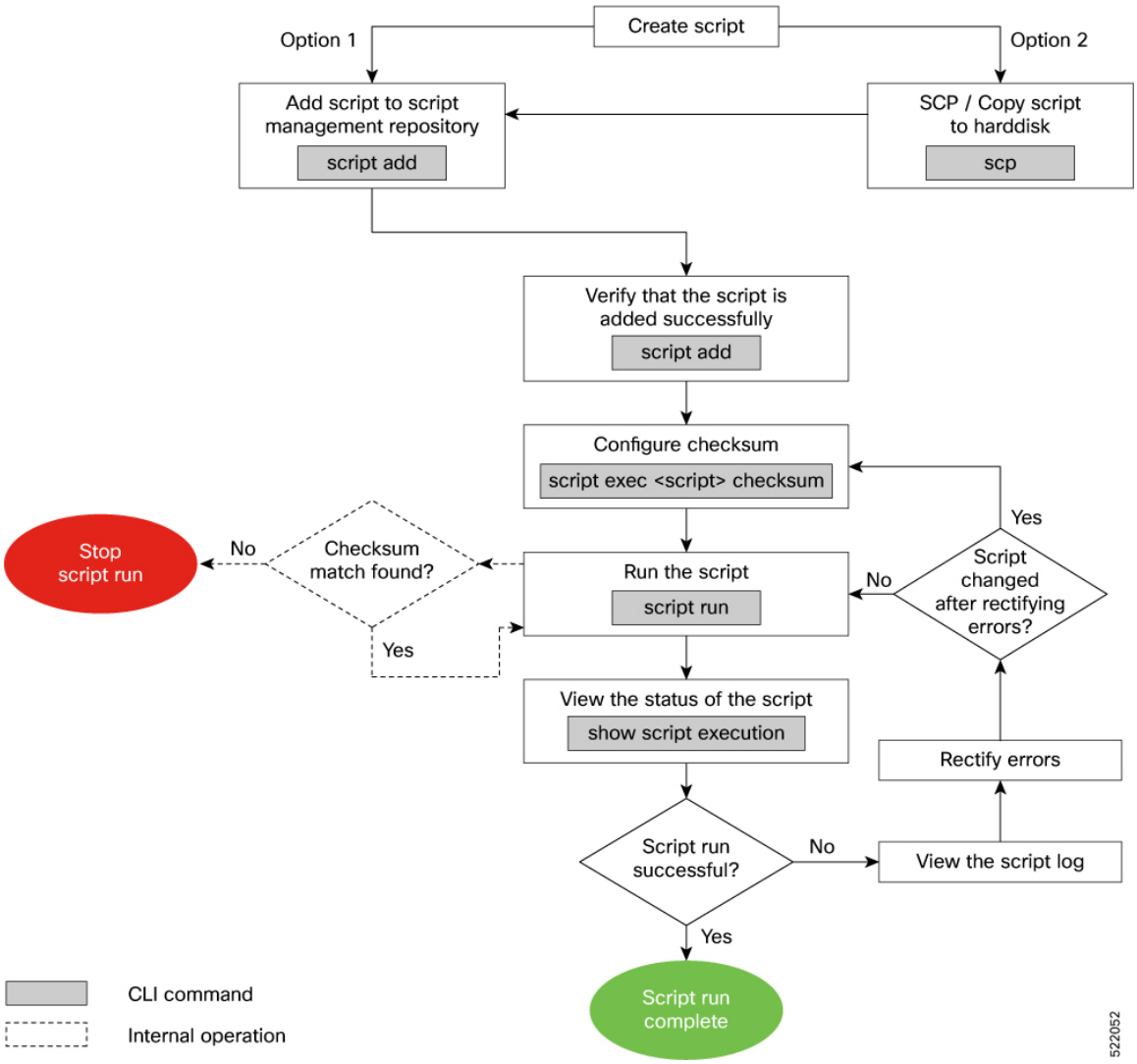
- automate device configurations throughout the network,
- are written in Python using libraries provided by Cisco with the base package, and
- execute within the Cisco IOS XR operating system.
- A script management repository on the router manages exec scripts and is replicated on both route processors (RPs).
- In IOS XR, AAA authorization determines user permissions required to run exec scripts; root privileges are necessary.

Note This chapter 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.

Provision an exec script

Set up and execute an exec script on a router to automate administrative or configuration tasks.

Figure 1: Steps involved in provisioning an exec script.



Procedure

Step 1 Download the script.

Step 2 Configure checksum.

Step 3 Run the script.

Step 4 View the script execution details.

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.

Table 1: Script download locations

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

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.

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

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

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

Note
The repository can be local to the router, or accessed remotely through TFTP, SCP, FTP, HTTP, or HTTPS protocols. In addition to the default Virtual Routing and Forwarding (VRF), support is also extended for non-default VRF.

You can add a maximum of 10 scripts simultaneously.

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

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

```
Router#script add exec 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.

- 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.

Example:

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

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

Example:

```
Router#script add exec /harddisk:/ exec-script.py
exec-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
---------------------------------------------------------------------------------------------
 exec-script.py     | exec     | Config Checksum  | NEW         | Tue Aug 24 10:18:23 2021
 =============================================================================================
```

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

Update scripts from a remote server

A remote script update capability is a script management feature that

- enables routers to synchronize automation scripts from a master script stored at a remote repository,
- automates the update process by applying changes based on predefined triggers such as manual requests, scheduled intervals, or execution events, and
- supports multiple transfer protocols and configurable authentication to ensure reliable and secure distribution of script updates.

You can maintain the latest copy of the scripts in a remote location, and configure the routers to automatically update the local copy with the latest copy on the server as required.

Table 2: Feature History Table

Feature Name	Release information	Description
Update Automation Scripts from Remote Server	Release 25.1.1	Introduced in this release on: 8700 [ASIC: K100](select variants only*) *This feature is now supported on Cisco 8712-MOD-M routers.

Feature Name	Release information	Description
Update Automation Scripts from Remote Server	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100(select variants only*)Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is now supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-12TH24FH-E • 88-LC1-36EH+A8:B12 • 88-LC1-52Y8H-EM
Update Automation Scripts from Remote Server	Release 7.5.1	This feature lets you update automation scripts across routers by accessing the master script from a remote site. This eases script management, where you make changes to the master script and then copy it to routers where it is deployed. This feature introduces the **auto-update** keyword in the **script exec** command.

Update scripts from a remote server

You can update script from a remote server using one of the following options:

- Config CLI commands
- Exec CLI commands

Exec CLI commands:

When you run the script, the script is downloaded and the checksum is automatically configured on the router.

- If **on-run** option is configured, running the **script run** command downloads the script.
- If **manual** option is configured, then you must run **script update** Exec command.
- If **schedule** option is selected, then the script is automatically updated after the specified interval.

Procedure

Step 1 Update the script on the router with the version on the remote server.

Example:

```
Router(config)#script exec auto-update sample3.py http://10.23.255.205
condition [manual | on-run | schedule]
```

In this example, `sample3.py` script is automatically updated from the remote server at `http://10.23.255.205`. You can set conditions when updating the script.

The repository can be accessed remotely through FTP, HTTP, HTTPS, TFTP or SCP protocols.

Table 3: Script update methods and options

Condition	Description
manual	Update manually with an Exec CLI (default). The following option is supported: • vrf—Specify the non-default Virtual Routing and Forwarding (VRF) name. • username—Enter the username. • password—Enter the password.
on-run	Update the exec script during run time. The following options are supported: • on-fail—Specify one of the actions on failure. • do-not-run—Do not run the script on failure. • run-local—Run the local copy of the script. • vrf—Specify the non-default VRF name. • username—Enter the username. • password—Enter the password. **Note** Only the exec scripts support the **on-run** option.

Condition	Description
schedule	Update automatically at specified time intervals. The following option is supported: • <60-262800>—Update interval in minutes • username—Enter the username. • password—Enter the password. **Note** The **schedule** option does not support SCP protocol.

Note
Do not specify the username and password inside the URL of the remote server.

Step 2 Commit the configuration.

Example:

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

Step 3 Run the script.

Example:

```
Router#script run sample3.py background
sample3.py has been added to the script repository
Script run scheduled: sample3.py. Request ID: 1624990452
```

You can specify additional options to the command:

- arguments: Script command-line arguments. The format is strings in single quotes. Escape double quotes inside string arguments.
- description: Description of script run.
- log-level: Script logging level. Default is INFO.
- log-path: Location to store script logs.
- max-runtime: Maximum run time of script.

Step 4 Update the script on the router with the version on the remote server.

Example:

```
Router#script update manual exec sample2.py
sample2.py has been added to the script repository
```

You can set options when updating the script:

Table 4: Description of various keywords

Option	Description
WORD	Script name.

Option	Description
all	Update all scripts in config.

Invoke scripts from a remote server

Execute a script remotely using a specified protocol URL and checksum for verification.

Use a remote server URL to run Python scripts. The checksum is required to ensure script integrity. Supported protocols include FTP, HTTP, HTTPS, TFTP, and SCP. Additional command options are available:

- arguments: Script command-line arguments, as strings in single quotes.
- description: Description of the script run.
- log-level: Script logging level (default: INFO).
- log-path: Location to store script logs.
- max-runtime: Maximum runtime for the script.
- vrf: Specify the VRF for the network file system.

Procedure

Run the script from the remote server.

Example:

```
Router#script run http://10.23.255.205/sample1.py checksum
5103a843032505decc37ff21089336e4bcc6a1061341056ca8add3ac5d6620ef background
Tue Nov 16 12:12:08.614 UTC
Script run scheduled: sample1.py. Request ID: 1624990451
```

Configure the checksum for an exec script

Ensure the integrity of exec scripts by configuring and verifying a SHA256 checksum, preventing tampering.

Every exec script requires a configured checksum. SHA256 is supported. The router verifies the checksum before executing the script. If the values do not match, the script will not run and a syslog warning is displayed.

Before you begin

- Ensure that the script is added to the script management repository. See Download the Script to the Router.
- Retrieve the SHA256 checksum value for your script on a trusted device.

Procedure

Step 1 Retrieve the SHA256 checksum hash value for the script. Ideally this action would be performed on a trusted device, such as the system on which the script was created. This minimizes the possibility that the script is tampered with.

Example:

```
Router#sha256sum sample1.py
94336f3997521d6e1aec0ee6faab0233562d53d4de7b0092e80b53caed58414b sample1.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
-------------------------------------------------------------------------------------------------------------

 sample1.py                       | exec   | Config Checksum  | NEW         | Fri Aug 20 05:03:41 2021
-------------------------------------------------------------------------------------------------------------

 Script Name       : sample1.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 Enter global configuration mode.

Example:

```
Router#configure
```

Step 4 Configure the checksum.

Example:

```
Router(config)#script exec sample1.py checksum SHA256
94336f3997521d6e1aec0ee6faab0233562d53d4de7b0092e80b53caed58414b

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

Step 5 Verify the status of the script.

Example:

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

 Name                              | Type   | Status           | Last Action | Action Time
-------------------------------------------------------------------------------------------------------------

 sample1.py                       | exec   | Ready            | NEW         | Fri Aug 20 05:03:41 2021
-------------------------------------------------------------------------------------------------------------
```

```
Script Name       : cpu_load.py
Checksum          : 94336f3997521d6e1aec0ee6faab0233562d53d4de7b0092e80b53caed58414b
History:
--------
1.   Action       : NEW
     Time         : Fri Aug 20 05:03:41 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.

Run an exec script

Execute an exec script on the router using the script run command and receive a unique request ID for tracking the execution.

Scripts allow automation of functions on your router. You can specify various runtime options such as arguments, descriptions, logging level, and maximum runtime when running a script.

To run an exec script, use the **script run** command. After the script is run, a request ID is generated. Each script run is associated with a unique request ID.

Scripts can be run with more options. This table lists the various options that you can provide at run time:

Table 5: Runtime configuration options

Keyword	Description
arguments	Script command-line arguments. Syntax: Strings in single quotes. Escape double quotes inside string arguments (if any). For example: `Router#script run sample1.py arguments 'hello world' '-r' '-t' 'exec' '--sleep' '5' description "Sample exec script"`
background	Run script in background. By default, the script runs in the foreground. When a script is run in the background, the console is accessible only after the script run is complete.
description	Description about the script run. `Router#script run sample1.py arguments '-arg1' 'reload' '-arg2' 'all' 'description' "Script reloads the router"` When you provide both the argument and description ensure that the arguments are in single quote and description is in double quotes.

Keyword	Description
log-level	Script logging level. The default value is `INFO`. You can specifiy what information is to be logged. The log level can be set to one of these options—Critical, Debug, Error, Info, or Warning.
log-path	Location to store the script logs. The default log file location is in the script management repository harddisk:/mirror/script-mgmt/logs.
max-runtime	Maximum run-time of script can be set between 1–3600 seconds. The default value is 300.

The script run is complete.

Before you begin

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

1. Download the Script to the Router
2. Configure Checksum for Exec Script

Procedure

Run the exec script.

Example:

```
Router#script run sample1.py
Script run scheduled: sample1.py. Request ID: 1629800603
Script sample1.py (exec) Execution complete: (Req. ID 1629800603) : Return Value: 0 (Executed)
```

What to do next

Review the script execution results and logs as needed. Use the request ID to track or troubleshoot any issues with the script run.

View the script execution details

View the status of the script execution.

Before you begin

Ensure you have completed these tasks.

- Download the script to the router.
- Configure checksum for exec script.
- Run the exec script.

Procedure

Step 1 View the status of the script execution.

Example:

```
Router#show script execution
==========================================================================================================================

 Req. ID   | Name (type)              | Start                     | Duration   | Return | Status
--------------------------------------------------------------------------------------------------------------------------

 1629800603| sample1.py (exec)        | Wed Aug 25 16:40:59 2021 | 60.62s     | 0      | Executed
==========================================================================================================================
```

You can view detailed or filtered data for every script run.

Step 2 Filter the script execution status to view the detailed output of a specific script run via request ID.

Example:

```
Router#show script execution request-id 1629800603 detail output
==========================================================================================================================

 Req. ID   | Name (type)                              | Start                     | Duration   |
Return | Status
--------------------------------------------------------------------------------------------------------------------------

 1629800603| sample1.py (exec)                        | Wed Aug 25 16:40:59 2021 | 60.62s     | 0
     | Executed
--------------------------------------------------------------------------------------------------------------------------

 Execution Details:
 ------------------
 Script Name  : sample1.py
 Log location : /harddisk:/mirror/script-mgmt/logs/sample1.py_exec_1629800603
 Arguments    :
 Run Options  : Logging level - INFO, Max. Runtime - 300s, Mode - Foreground
 Events:
 -------
 1.   Event        : New
      Time         : Wed Aug 25 16:40:59 2021
      Time Elapsed : 0.00s Seconds
      Description  : None
 2.   Event        : Started
      Time         : Wed Aug 25 16:40:59 2021
      Time Elapsed : 0.03s Seconds
      Description  : Script execution started. PID (20736)
 3.   Event        : Executed
      Time         : Wed Aug 25 16:42:00 2021
      Time Elapsed : 60.62s Seconds
      Description  : Script execution complete
--------------------------------------------------------------------------------------------------------------------------

 Script Output:
 --------------
 Output File  : /harddisk:/mirror/script-mgmt/logs/sample1.py_exec_1629800603/stdout.log
 Content      :
==========================================================================================
```

Table 6: Description of various keywords

Keyword	Description
detail	Display detailed script execution history, errors, output and deleted scripts. `Router#show script execution detail [errors | output | show-del]`
last <number>	Show last N (1-100) execution requests. `Router#show script execution last 10` This example will display the list of last 10 script runs with their request IDs, type of script, timestamp, duration that the script was run, number of errrors, and the status of the script run.
name <filename>	Filter operational data based on script name. If not specified, all scripts are displayed. `Router#show script execution name sample1.py`
request-id <value>	Display summary of the script using request-ID that is generated with each script run. `Router#show script execution request-ID 1629800603`
reverse	Display the request IDs from the script execution in reverse chronological order. For example, the request-ID from the latest run is displayed first, followed by the descending order of request-IDs. `Router#script script execution reverse`
status	Filter data based on script status. `Router#[status {Exception,Executed, Killed, Started, Stopped, Timed-out}]`

Delete exec scripts from the router

Remove exec scripts from the router's script management repository.

Use this task to delete unwanted exec scripts stored in the script management repository using the `script remove` command. The repository stores downloaded scripts available for use on the router.

Procedure

Step 1 View the list of scripts present in the script management repository.

Example:

```
Router#show script status
=============================================================================================================
Name       | Type   | Status          | Last Action | Action Time
-------------------------------------------------------------------------------------------------------------
sample1.py | exec   | Config Checksum | NEW         | Tue Aug 24 10:18:23 2021
sample2.py | exec   | Config Checksum | NEW         | Wed Aug 25 23:44:53 2021
sample3.py | config | Config Checksum | NEW         | Wed Aug 25 23:44:57 2021
```

Ensure the script you want to delete is present in the repository.

Step 2 Delete the script.

Example:

```
Router#script remove exec sample2.py
sample2.py has been deleted from the script repository
```

You can also delete multiple scripts simulataneoulsy.

Step 3 Verify the script is deleted from the subdirectory.

Example:

```
Router#show script status
=============================================================================================================
Name       | Type   | Status           | Last Action | Action Time
-------------------------------------------------------------------------------------------------------------
sample1.py | exec   | Config Checksum | NEW         | Tue Aug 24 10:18:23 2021
sample3.py | config | Config Checksum | NEW         | Wed Aug 25 10:44:57 2021
```

The script is deleted from the script management repository.