Exec Scripts

Cisco IOS XR exec scripts are on-box scripts that automate configurations of devices in the network. The exec scripts are written in Python using the Python libraries that Cisco provides with the base package. For the list of supported packages

A script management repository on the router manages the exec scripts. This repository is replicated on both RPs.

In IOS XR, AAA authorization controls the user access and privileges to perform operations. To run the exec script, you must have root user permissions.

Exec scripts provide the following advantages:

- Provides automation capabilities to simplify complex operations.
- Create customized operations based on the requirement.
- Provide flexibility in changing the input parameters for every script run. This fosters dynamic automation of operational information.
- Detect and display errors and warnings when executing an operation.
- Run multiple automated operations in parallel without blocking the console.

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

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.

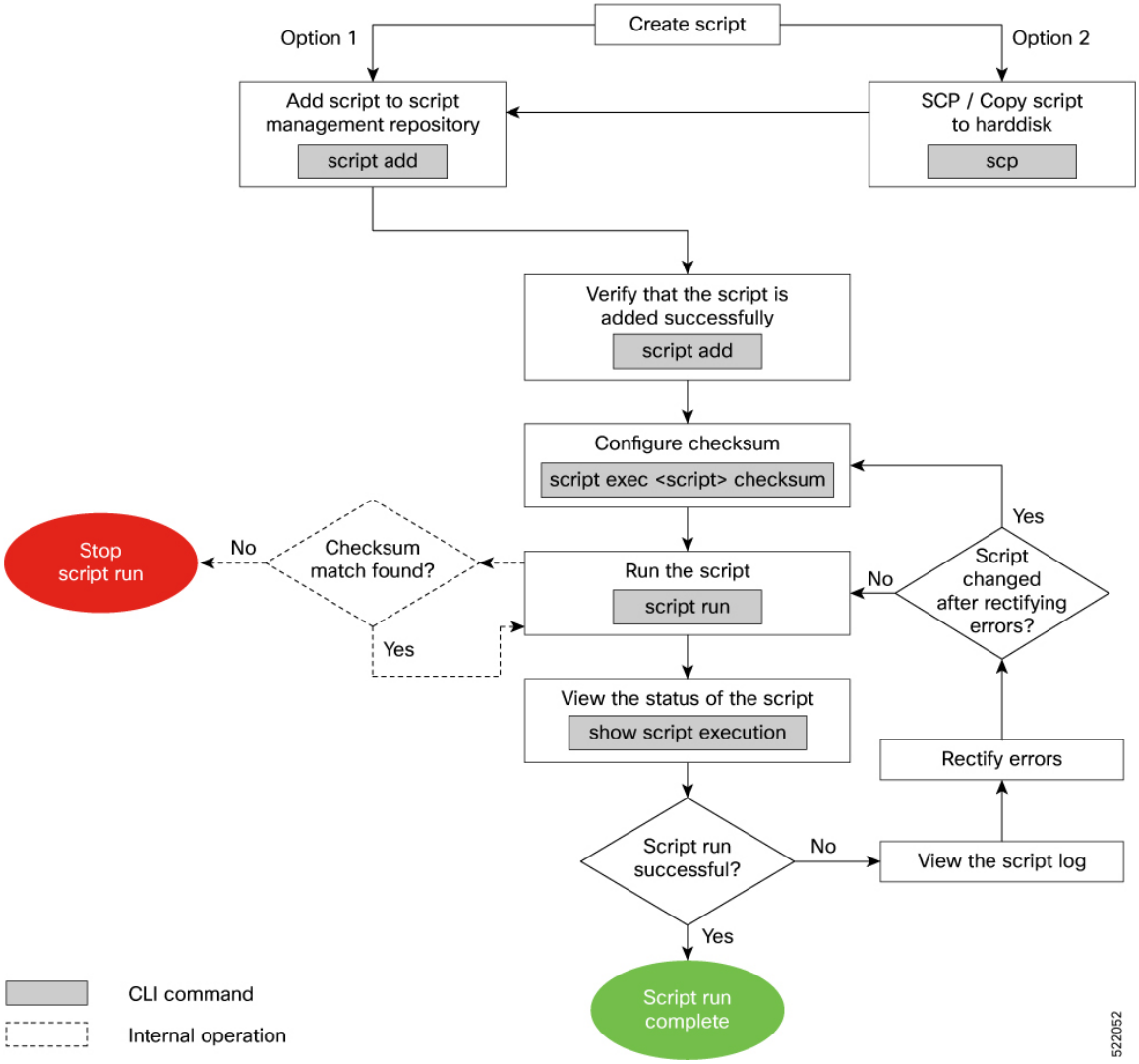
- Workflow to Run an Exec Script, on page 1
- Manage Scripts, on page 12
- Example: Exec Script to Verify Bundle Interfaces, on page 13

Workflow to Run an Exec Script

Complete the following tasks to provision exec scripts:

- Download the script—Add the script to the appropriate exec script directory on the router. using the **script add exec** command.
- Configure checksum—Check script integrity and authenticity using the **script exec <script.py> checksum** command.
- Run the script—Trigger changes to the router configuration. Include arguments, set the maximum time for the script to run, setup log levels using the **script run** command.
- View the script execution details—Validate the script and retrieve the operational data using the **show script execution** command.

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



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	`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

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

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 a configured HTTP server or the harddisk location in the router.

  ```
  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.

  Note
  Only SHA256 checksum is supported.

  ```
  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.

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

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

  ```
  Router#script add exec /harddisk:/ exec-script.py
  Fri Aug 20 05:03:40.791 UTC
  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
Wed Aug 25 23:10:50.453 UTC
=============================================================================================
 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

Table 1: Feature History Table

Feature Name	Release Information	Description
Update Automation Scripts from Remote Server	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I

Feature Name	Release Information	Description
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.

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.

Procedure

You can update the script using one of the following options.

- **Config CLI commands:**

  a. Update the script on the router with the version on the remote server.

  ```
  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.

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.

Condition	Description
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.
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.

b. Commit the configuration.

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

c. Run the script.

```
Router#script run sample3.py background
Tue Nov 16 12:50:33.512 UTC
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.

- **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.

a. Update the script on the router with the version on the remote server.

```
Router#script update manual exec sample2.py
Tue Nov 16 12:20:23.058 UTC
sample2.py has been added to the script repository
```

You can set options when updating the script:

Option	Description
WORD	Script name.
all	Update all scripts in config.

Invoke Scripts from a Remote Server

You can directly run the script using the URL to the remote server and provide the checksum value. The checksum is a mandatory parameter. The format of the URL is `[protocol]://[user:password@]server[:port]/directory/file_name`.

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
```

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

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.

- **vrf:** Specify the VRF for the network file system.

Configure Checksum for Exec Script

Every script is associated with a checksum value. The checksum ensures the integrity of the script that is downloaded from the server or external repository is intact, and that the script is not tampered. The checksum is a string of numbers and letters that act as a fingerprint for script. The checksum of the script is compared with the configured checksum. If the values do not match, the script is not run and a syslog warning message is displayed.

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

Note Exec scripts support SHA256 checksum.

Before you begin

Ensure that the script is added to the script management repository. See Download the Script to the Router, on page 3.

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:

```
Server$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
Fri Aug 20 05:04:13.539 UTC
==========================================================================================

 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
Tue Aug 24 10:23:10.546 UTC
Router(config)#end
```

Step 5 Verify the status of the script.

Example:

```
Router#show script status detail
Fri Aug 20 05:06:17.296 UTC
==========================================================================================

 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 the Exec 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.

Before you begin

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

1. Download the Script to the Router, on page 3
2. Configure Checksum for Exec Script, on page 8

Procedure

Run the exec script.

Example:

```
Router#script run sample1.py
Wed Aug 25 16:40:59.134 UTC
Script run scheduled: sample1.py. Request ID: 1629800603
Script sample1.py (exec) Execution complete: (Req. ID 1629800603) : Return Value: 0 (Executed)
```

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

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.
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.

View the Script Execution Details

View the status of the script execution.

Before you begin

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

1. Download the Script to the Router, on page 3
2. Configure Checksum for Exec Script, on page 8
3. Run the Exec Script, on page 9

Procedure

Step 1 View the status of the script execution.

Example:

```
Router#show script execution
Wed Aug 25 18:32:12.351 UTC
==========================================================================================================

 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
Wed Aug 25 18:37:12.920 UTC
==========================================================================================================

 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      :
===================================================================================================================
```

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}]`

Manage Scripts

This section shows the additional operations that you can perform on a script.

Delete Exec Script from the Router

Delete the script from the script management repository using the **script remove** command. This repository stores the downloaded scripts.

Procedure

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

Example:

```
Router#show script status
Wed Aug 25 23:10:50.453 UTC
=============================================================================================================
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
Wed Aug 25 231:46:38.170 UTC
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
Wed Aug 25 23:48:50.453 UTC
=============================================================================================================
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.

Example: Exec Script to Verify Bundle Interfaces

In this example, you create a script to verify the bandwidth usage of bundle interfaces on the router, and check if it is beyond the defined limit. If usage is above the limit, the script generates a syslog indicating that the bandwidth is above the limit, and additional interfaces must be added to the bundle.

Before you begin

Ensure you have completed the following prerequisites before you validate the script:

1. Create an exec script `verify_bundle.py`. Store the script on an HTTP server or copy the script to the harddisk of the router.

   ```
   """
   Bundle interfaces bandwidth verification script
   ```

```
Verify bundle interfaces mpls packets per sec is below threshold.
If pkts/sec is greater than threshold then print syslog message
and add list of new interfaces to bundle

Arguments:
  -h, --help            show this help message and exit
  -n NAME, --name NAME  Bundle interface name
  -t THRESHOLD, --threshold THRESHOLD
                        Bandwidth threshold
  -m MEMBERS, --members MEMBERS
                        interfaces (coma separated) to add to bundle
"""
import re
import argparse
from iosxr.xrcli.xrcli_helper import XrcliHelper
from cisco.script_mgmt import xrlog

syslog = xrlog.getSysLogger('verify_bundle')
log = xrlog.getScriptLogger('verify_bundle')


def add_bundle_members(bundle_name, members):

    helper = XrcliHelper()
    bundle_pattern = re.compile('[A-Z,a-z, ]([0-9]+)')
    match = bundle_pattern.search(bundle_name)
    if match:
        bundle_id = match.group(1)
    else:
        raise Exception('Invalid bundle name')
    cfg = ''
    for member in members:

        cfg = cfg + 'interface %s \nbundle id %s mode active\nno shutdown\n' % \
              (member.strip(), bundle_id)

    log.info("Configs to be added : \n%s" % cfg)
    result = helper.xr_apply_config_string(cfg)
    if result['status'] == 'success':
        msg = "Configuring new bundle members successful"
        syslog.info(msg)
        log.info(msg)
    else:
        msg = "Configuring new bundle members failed"
        syslog.warning(msg)
        log.warning(msg)


def verify_bundle(script_args):

    helper = XrcliHelper()
    cmd = "show interfaces %s accounting rates" % script_args.name
    cmd_out = helper.xrcli_exec(cmd)
    if not cmd_out['status'] == 'success':
        raise Exception('Invalid bundle or error getting interface accounting rates')

    log.info('Command output : \n%s' % cmd_out['output'])
    rate_pattern = re.compile("MPLS +[0-9]+ +[0-9]+ +[0-9]+ +([0-9]+)")
    match = rate_pattern.search(cmd_out['output'])
    if match:
        pktspersec = int(match.group(1))
        if pktspersec > int(script_args.threshold):
            msg = 'Bundle %s bandwidth of %d pps is above threshold of %s pps' % \
```

```
                    (script_args.name, pktspersec, script_args.threshold)
            log.info(msg)
            syslog.info(msg)
            return False
        else:
            msg = 'Bundle %s bandwidth of %d pps is below threshold of %s pps' % \
                    (script_args.name, pktspersec, script_args.threshold)
            log.info(msg)
            return True


if __name__ == '__main__':

    parser = argparse.ArgumentParser(description="Verify budle")
    parser.add_argument("-n", "--name",
                        help="Bundle interface name")
    parser.add_argument("-t", "--threshold",
                        help="Bandwidth threshold")
    parser.add_argument("-m", "--members",
                        help="interfaces (coma separated) to add to bundle")
    args = parser.parse_args()
    log.info('Script arguments :')
    log.info(args)
    if not verify_bundle(args):
        syslog.info("Adding new members (%s) to bundle interfaces %s" %
                    (args.members, args.name))
        add_bundle_members(args.name, args.members.split(','))
```

2. Add the script from HTTP server or harddisk to the script management repository. See Download the Script to the Router, on page 3.
3. Configure the checksum to verify the authenticity and integrity of the script.

Procedure

Step 1 View the script status.

Example:

```
Router#show script status
Sat Sep 25 00:10:11.222 UTC
================================================================================
Name                | Type   | Status   | Last Action | Action Time
--------------------------------------------------------------------------------
verify_bundle.py    | exec   | Ready    | MODIFY      | Sat Sep 25 00:08:55 2021
================================================================================
```

The status indicates that the script is ready to be run.

Step 2 Run the script.

Example:

```
Router#script run verify_bundle.py arguments '--name' 'Bundle-Ether6432' '-t'
'400000' '-m' 'FourHundredGigE0/0/0/2
Sat Sep 25 00:11:14.183 UTC
Script run scheduled: verify_bundle.py. Request ID: 1632528674
[2021-09-25 00:11:14,579] INFO [verify_bundle]:: Script arguments :
[2021-09-25 00:11:14,579] INFO [verify_bundle]:: Namespace(members='FourHundredGigE0/0/0/2,
FourHundredGigE0/0/0/3', name='Bundle-Ether6432', threshold='400000')
[2021-09-25 00:11:14,735] INFO [verify_bundle]:: Command output :
```

```
-------------- show interfaces Bundle-Ether6432 accounting rates --------------
Bundle-Ether6432
                              Ingress                        Egress
  Protocol             Bits/sec         Pkts/sec     Bits/sec         Pkts/sec
  IPV4_UNICAST            22000               40            0                0
  MPLS                        0                0   1979249000           430742
  ARP                         0                0            0                0
  IPV6_ND                     0                0            0                0
  CLNS                     1000                1        26000                3


[2021-09-25 00:11:14,736] INFO [verify_bundle]:: Bundle Bundle-Ether6432 bandwidth
of 430742 pps is above threshold of 400000 pps
[2021-09-25 00:11:14,737] INFO [verify_bundle]:: Configs to be added :
interface FourHundredGigE0/0/0/2
bundle id 6432 mode active
no shutdown
interface FourHundredGigE0/0/0/3
bundle id 6432 mode active
no shutdown

[2021-09-25 00:11:18,254] INFO [verify_bundle]:: Configuring new bundle members successful
Script verify_bundle.py (exec) Execution complete: (Req. ID 1632528674) : Return Value: 0 (Executed)
```

Step 3 View the detailed output based on request ID. A request ID is generated for each script run.

Example:

```
Router#show script execution request-id 1632528674 detail output
Sat Sep 25 00:11:58.141 UTC
=============================================================================================================
Req. ID   | Name (type)               | Start                    | Duration   | Return | Status
-------------------------------------------------------------------------------------------------------------
1632528674| verify_bundle.py (exec)    | Sat Sep 25 00:11:14 2021 | 4.06s      | 0      | Executed
-------------------------------------------------------------------------------------------------------------
Execution Details:
------------------
Script Name  : verify_bundle.py
Log location : /harddisk:/mirror/script-mgmt/logs/verify_bundle.py_exec_1632528674
Arguments    : '--name', 'Bundle-Ether6432', '-t', '400000', '-m', 'FourHundredGigE0/0/0/2,
FourHundredGigE0/0/0/3'
Run Options  : Logging level - INFO, Max. Runtime - 300s, Mode - Foreground
Events:
-------
1.   Event        : New
      Time         : Sat Sep 25 00:11:14 2021
      Time Elapsed : 0.00s Seconds
      Description  : None
2.   Event        : Started
      Time         : Sat Sep 25 00:11:14 2021
      Time Elapsed : 0.02s Seconds
      Description  : Script execution started. PID (29768)
3.   Event        : Executed
      Time         : Sat Sep 25 00:11:18 2021
      Time Elapsed : 4.06s Seconds
      Description  : Script execution complete
-------------------------------------------------------------------------------------------------------------
Script Output:
--------------
Output File  : /harddisk:/mirror/script-mgmt/logs/verify_bundle.py_exec_1632528674/stdout.log
Content      :
[2021-09-25 00:11:14,579] INFO [verify_bundle]:: Script arguments :
[2021-09-25 00:11:14,579] INFO [verify_bundle]:: Namespace(members='FourHundredGigE0/0/0/2,
```

```
FourHundredGigE0/0/0/3',
name='Bundle-Ether6432', threshold='400000')
[2021-09-25 00:11:14,735] INFO [verify_bundle]:: Command output :

-------------- show interfaces Bundle-Ether6432 accounting rates --------------
Bundle-Ether6432
                              Ingress                        Egress
  Protocol             Bits/sec         Pkts/sec     Bits/sec         Pkts/sec
  IPV4_UNICAST            22000               40            0                0
  MPLS                        0                0   1979249000           430742
  ARP                         0                0            0                0
  IPV6_ND                     0                0            0                0
  CLNS                     1000                1        26000                3

[2021-09-25 00:11:14,736] INFO [verify_bundle]:: Bundle Bundle-Ether6432 bandwidth of 430742 pps is
 above threshold
of 400000 pps
[2021-09-25 00:11:14,737] INFO [verify_bundle]:: Configs to be added :
interface FourHundredGigE0/0/0/2
bundle id 6432 mode active
no shutdown
interface FourHundredGigE0/0/0/3
bundle id 6432 mode active
no shutdown

[2021-09-25 00:11:18,254] INFO [verify_bundle]:: Configuring new bundle members successful
================================================================================================================================
```

Step 4 View the running configuration for the bundle interfaces.

Example:

```
Router#show running-config interface FourHundredGigE0/0/0/2
Sat Sep 25 00:12:30.765 UTC
interface FourHundredGigE0/0/0/2
bundle id 6432 mode active
!

Router#show running-config interface FourHundredGigE0/0/0/3
Sat Sep 25 00:12:38.659 UTC
interface FourHundredGigE0/0/0/3
bundle id 6432 mode active
!
```

Step 5 View the latest logs for more details about the script run. Here, the last 10 logs are displayed. The logs show that configuring new bundle members is successful.

Example:

```
Router#show logging last 10
Sat Sep 25 00:13:34.383 UTC
Syslog logging: enabled (0 messages dropped, 0 flushes, 0 overruns)
    Console logging: level warnings, 178 messages logged
    Monitor logging: level debugging, 0 messages logged
    Trap logging: level informational, 0 messages logged
    Buffer logging: level debugging, 801 messages logged

Log Buffer (2097152 bytes):

RP/0/RP0/CPU0:Sep 25 00:10:05.763 UTC: config[66385]: %MGBL-CONFIG-6-DB_COMMIT : Configuration
committed by user 'cisco'.
Use 'show configuration commit changes 1000000045' to view the changes.
RP/0/RP0/CPU0:Sep 25 00:10:07.971 UTC: config[66385]: %MGBL-SYS-5-CONFIG_I : Configured from console
 by cisco on vty0 (6.3.65.175)
```

```
RP/0/RP0/CPU0:Sep 25 00:11:14.447 UTC: script_control_cli[66627]: %OS-SCRIPT_MGMT-6-INFO :
Script-control: Script run scheduled:
verify_bundle.py. Request ID: 1632528674
RP/0/RP0/CPU0:Sep 25 00:11:14.453 UTC: script_agent_main[347]: %OS-SCRIPT_MGMT-6-INFO :
Script-script_agent: Script execution
verify_bundle.py (exec) Started : Request ID : 1632528674 :: PID: 29768
RP/0/RP0/CPU0:Sep 25 00:11:14.453 UTC: script_agent_main[347]: %OS-SCRIPT_MGMT-6-INFO :
Script-script_agent: Starting execution
verify_bundle.py (exec) (Req. ID: 1632528674) : Logs directory:
/harddisk:/mirror/script-mgmt/logs/verify_bundle.py_exec_1632528674
RP/0/RP0/CPU0:Sep 25 00:11:14.736 UTC: python3_xr[66632]: %OS-SCRIPT_MGMT-6-INFO : Script-verify_bundle:
 Bundle Bundle-Ether6432
bandwidth of 430742 pps is above threshold of 400000 pps
RP/0/RP0/CPU0:Sep 25 00:11:14.736 UTC: python3_xr[66632]: %OS-SCRIPT_MGMT-6-INFO : Script-verify_bundle:
 Adding new members
(FourHundredGigE0/0/0/2, FourHundredGigE0/0/0/3) to bundle interfaces Bundle-Ether6432
RP/0/RP0/CPU0:Sep 25 00:11:16.916 UTC: config[66655]: %MGBL-CONFIG-6-DB_COMMIT : Configuration
committed by user 'cisco'. Use 'show
configuration commit changes 1000000046' to view the changes.
RP/0/RP0/CPU0:Sep 25 00:11:18.254 UTC: python3_xr[66632]: %OS-SCRIPT_MGMT-6-INFO : Script-verify_bundle:
 Configuring new bundle members
successful
RP/0/RP0/CPU0:Sep 25 00:11:18.497 UTC: script_agent_main[347]: %OS-SCRIPT_MGMT-6-INFO :
Script-script_agent: Script verify_bundle.py
(exec) Execution complete: (Req. ID 1632528674) : Return Value: 0 (Executed)
```