Precommit Scripts

Table 1: Feature History Table

Feature Name	Release Information	Description
Precommit Script to Validate Configuration Change	Release 7.5.4	With this feature, you can deploy custom python scripts to be executed automatically during a configuration commit operation. These scripts process the configuration change and act as deciding factor to either proceed with applying the configuration or stop the commit operation in the event of an error.

Cisco IOS XR precommit scripts can validate the configuration during the commit operation. They allow device administrators to enforce custom configuration validation rules. These scripts are invoked automatically when you change a configuration and commit the changes. When a configuration commit is in progress, a precommit script is automatically initaited to validate the changes. If the change is valid, the script allows committing the new configuration. If the configuration is invalid, or does not adhere to the enforced validation rules, the script notifies you about the mismatch and blocks the commit operation. Overall, precommit scripts help to maintain crucial device parameters, and reduce human error in managing the network.

When you commit a configuration, the system automatically invokes the precommit scripts to validate that change. Precommit scripts can perform the following actions during a commit operation:

- Validate the proposed new configuration, ensure that the changes to the target configuration does not exceed the boundaries defined for the system or software functionality. For example, you can program the script to estimate the Ternary Content Addressable Memory (TCAM) slots needed for the target configuration, and verify that the TCAM usage does not exceed a defined threshold.
- Verify that the commit operation adheres to the predefined execution rules. For example, you can use the script to ensure that certain configuration changes that impact traffic are allowed only at specified time intervals.
- Block the commit operation if the configuration is invalid and notify the details in an error message.
- Generate system log messages for in-depth analysis of the configuration change. This log also helps in troubleshooting a failed commit operation.

Precommit Script Limitations

The following restrictions apply when using precommit scripts:

- Precommit scripts cannot modify a configuration.
- Configuration validation before a commit operation is supported only using CLI commands. Operations using NETCONF, gNMI and XML are not supported even if the precommit script is enabled.

Get Started with Precommit Scripts

Precommit scripts can be written in Python 3.9 (and earlier) programming language using the packages that Cisco supports. For more information about the supported packages, see Script Infrastructure and Sample Templates.

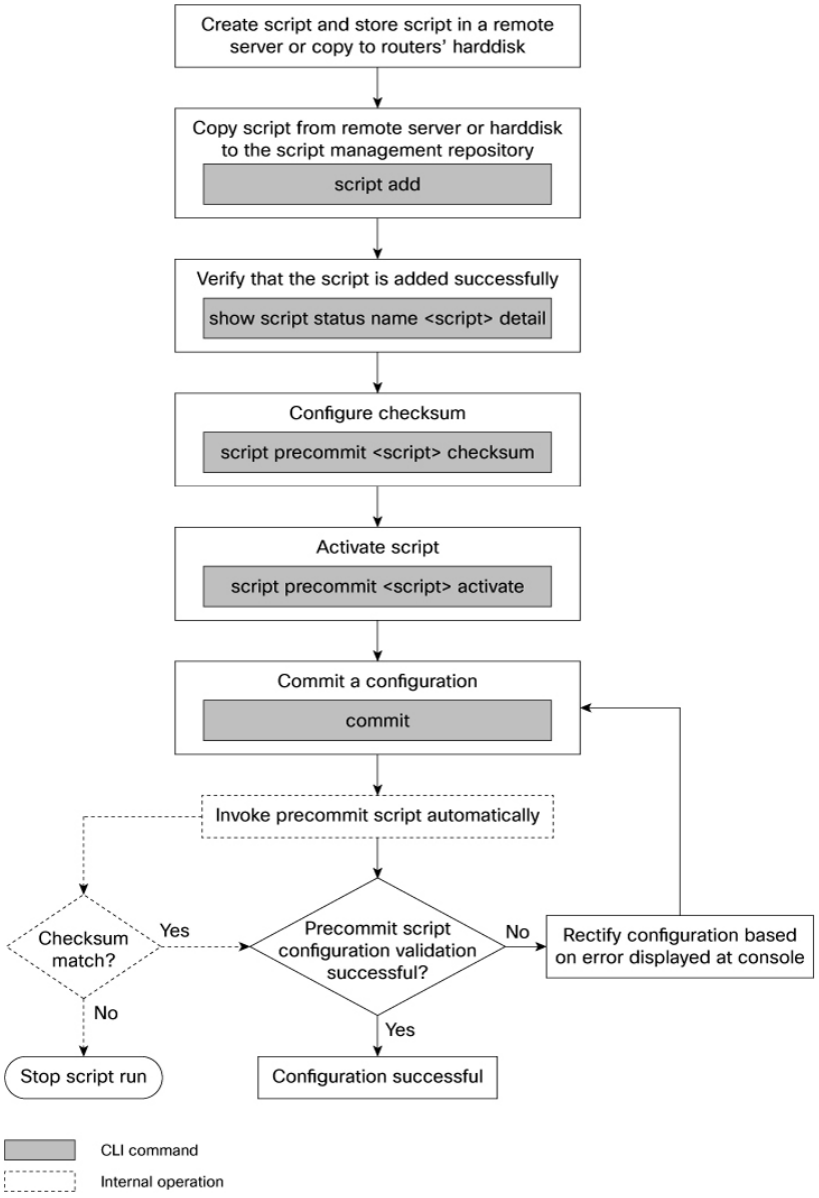
This chapter gets you started with provisioning your precommit automation scripts on the router.

Note This chapter does not delve into creating Python scripts, but assumes that you have basic understanding of Python programming language. This section walks you through the process involved in deploying and using the precommit scripts on the router.

- Workflow to Run Precommit Scripts, on page 2
- Example: Verify BGP Configuration Using Precommit Script, on page 8

Workflow to Run Precommit Scripts

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



Complete the following tasks to provision precommit scripts:

- Download the Script to the Router—Store the precomit script on a remote server or copy to the harddisk of the router. Add the precommit script from the server to the script management repository (`hardisk:/mirror/script-mgmt`) on the router using the **script add precommit** command.
- Configure Checksum for Precommit Script—Configure the script integrity and authenticity using the **script precommit** *script* **checksum** command. A script cannot be used unless the checksum is configured.
- Activate Precommit Scripts—Activate the precommit script using **script precommit** *script* **activate** command to validate the configuration from a commit operation. The script ensures that the configuration changes comply with the predefined conditions in the script, and uncover potential errors, if any.

Note A precommit script is invoked automatically when you commit a configuration change to modify the router configuration. You can view the result from the script execution on the console.

Download the Script to the Router

Script Type	Download Location
precommit	`harddisk:/mirror/script-mgmt/precommit`
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 `precommit-bgp.py` script to the script management repository.

Before you begin

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.

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 remote server (HTTP, HTTPS, FTP or SCP) or the harddisk location in the router.

  ```
  Router#script add precommit script-location script.py
  ```

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

  ```
  Router#script add precommit http://192.0.2.0/scripts precommit-bgp.py
  Tue Jan 24 05:03:40.791 UTC
  Copying script from http://192.0.2.0/scripts/precommit-bgp.py
  precommit-bgp.py has been added to the script repository
  ```

  You can add a maximum of 10 scripts simultaneously.

```
Router#script add precommit 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 precommit http://192.0.2.0/scripts precommit-bgp.py checksum SHA256 checksum-value
```

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

```
Router#script add precommit 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/precommit-bgp.py /harddisk:/
  ```

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

  ```
  Router#script add precommit /harddisk:/ precommit-bgp.py
  Tue Jan 24 05:03:40.791 UTC
  Copying script from /harddisk:/precommit-bgp.py
  precommit-bgp.py has been added to the script repository
  ```

Step 2 Verify that the script is downloaded to the script management repository on the router.

Example:

```
Router#show script status
Tue Jan 24 05:10:40.791 UTCC
=============================================================================================================

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

 precommit-bgp.py          | precommit   | Config Checksum  | NEW         | Tue Jan 24 05:10:18 2023
```

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

Configure Checksum for Precommit Script

Every script is associated with a checksum hash value. This value ensures the integrity of the script, and that the script is not tampered with. 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 Precommit scripts support SHA256 checksum.

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. However, if the router is secure, you can retrieve the checksum hash value from the IOS XR Linux bash shell.

Example:

```
Router#run
[node0_RP0_CPU0:~]$sha256sum /harddisk:/mirror/script-mgmt/precommit/precommit-bgp.py
6bb460920a694a0f91a27892f457203090e7a6391ab7d2f8656f477af17f9ed1
/harddisk:/mirror/script-mgmt/precommit/precommit-bgp.py
```

Make note of the checksum value.

Step 2 View the status of the script.

Example:

```
Router#show script status detail
Tue Jan 24 05:20:13.539 UTC
=====================================================================================================

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

 precommit-bgp.py          | precommit    | Config Checksum  | NEW            | Tue Jan 24 05:19:41
2023
-----------------------------------------------------------------------------------------------------

 Script Name       : precommit-bgp-script.py
 History:
 --------
 1.   Action       : NEW
      Time         : Tue Jan 24 05:19:41 2021
      Description  : User action IN_CLOSE_WRITE
=========================================================================================
```

The `Status` shows that the checksum is not configured.

You can view the details of the specific script using the **show script status name** *script* **detail** command.

Step 3 Configure the checksum and set the priority.

Example:

```
Router#configure
Router(config)#script precommit precommit-bgp.py checksum SHA256
6bb460920a694a0f91a27892f457203090e7a6391ab7d2f8656f477af17f9ed1 priority 20
Router(config)#commit
Tue Jan 24 10:23:10.546 UTC
Router(config)#end
```

If you are configuring multiple scripts, the system decides an appropriate order to run the scripts. However, you can control the order in which scripts execute using a priority value. For more information on configuring the priority value, see Control Priority When Running Multiple Scripts.

Step 4 Verify the status of the script.

Example:

```
Router#show script status detail
Tue Jan 24 05:06:17.296 UTC
=============================================================================================================

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

 precommit-bgp.py              | precommit | Ready            | NEW         | Tue Jan 24 06:17:41 2023
-------------------------------------------------------------------------------------------------------------

 Script Name       : precommit-bgp.py
 Checksum          : 6bb460920a694a0f91a27892f457203090e7a6391ab7d2f8656f477af17f9ed1
 History:
 --------
 1.   Action       : NEW
      Time         : Tue Jan 24 06:17:41 2023
      Checksum     : 6bb460920a694a0f91a27892f457203090e7a6391ab7d2f8656f477af17f9ed1
      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 is not run, and the commit operation that triggered the script is rejected. It is mandatory for the checksum values to match for the script to run.

Activate Precommit Scripts

Activate the precommit script to validate a configuration change on the set of active configuration (including any scripts newly activated as part of the configuration change) before committing the changes.

Note If the precommit script rejects one or more items in the configuration change, the entire configuration is rejected before committing the change.

Before you begin

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

1. Download the Script to the Router, on page 4
2. Configure Checksum for Precommit Script, on page 5

Procedure

Step 1 Activate the precommit script for the configuration validation to take effect.

Example:

```
Router(config)#script precommit precommit-bgp.py activate
```

Step 2 Commit the changes and verify that the precommit script is automatically initiated. You can choose to perform one of the following options based on the requirement:

- Commit the changes to automatically initiate the precommit verification script.

```
Router(config-bgp-nbr)#commit
Tue Jan 24 00:13:37.050 UTC
Precommit Script Report Start
------------------------------
Pre-commit Verification Result: Pass
Pre-commit Verification Script precommit-bgp.py (req id 1656378102): Pass
------------------------------
Precommit Script Report Done
```

- Ignore the result of the precommit script execution and proceed to the next step in the commit process using **ignore-results** keyword. Use this keyword if you want to bypass the commit verification. The precommit script is still executed, but the result is ignored.

```
Router(config-bgp-nbr)#commit script-verification ignore-results
```

- View all the logs generated by the commit script on the console using **verbose** keyword. If this keyword is not specified, only the result of the script verification is displayed on the console.

```
Router(config-bgp-nbr)#commit script-verification verbose
```

An execution report from the script is displayed on the console. If the script displays an error message, rectify the error and rerun the commit operation. If there are no validation errors, the commit operation is successful indicating that the configuration change is valid.

Example: Verify BGP Configuration Using Precommit Script

In this example, you create a precommit script to validate the following Border Gateway Protocol (BGP) configuration:

- Check that the autonomous system (AS) value is in the range from `123` to `234`
- Check that the remote AS of neighbours is not set to `25`

Procedure

Step 1 Create a precommit script named `verify-bgp.py`. Store the script on a remote server or copy the script to the harddisk: location of the router.

Example:

```
"""

import re
from iosxr.xrcli.xrcli_helper import XrcliHelper
from cisco.script_mgmt import xrlog
from cisco.script_mgmt import precommit

syslog = xrlog.getSysLogger('precommit_verify_bgp')
log = xrlog.getScriptLogger('precommit_verify_bgp')
```

```
helper = XrcliHelper(debug=True)


def verify_bgp():
    """
    Query for target configs and check if the target configs has bgp configs
    Check if the bgp AS is in the range 123-234
    Check if remote AS is not 25.
    :return: None on pass / Raise exception on failure.
    """

    # CLI verification
    cfg = precommit.get_target_configs()
    #cfg = "Thu Feb 23 18:54:28.605 UTC\nrouter bgp 100\n neighbor 10.0.0.1\n remote-as 25\n !\n!\n"

    #cfg = cfg.split("\n")
    print(cfg)

    for cfg_line in cfg:

        bgp_cfg_start_pattern = re.match("^router bgp (.*)", cfg_line)
        if bgp_cfg_start_pattern:
            log.info("BGP config found")

            bgp_as = int(bgp_cfg_start_pattern.group(1))
            if not bgp_as in range(123, 234):
                precommit.config_warning("BGP AS number (%d) " % bgp_as +
                                         "not in recommended range (123-234)")

    # sysdb verification
    cfg = precommit.get_target_configs(format="sysdb")
    # cfg = [Item(name='gl/ip-bgp/default/0/100/aya', value=1, datatype=1),
    # Item(name='gl/ip-bgp/default/0/100/gbl/edm/ord_a/running', value=1, datatype=1),
    #
Item(name='gl/ip-bgp/default/0/100/ord_a/default/nbr/________/edm/ord_u/0x3/10.0.0.1/________/________/aya',
 value=1, datatype=1),
    #
Item(name='gl/ip-bgp/default/0/100/ord_a/default/nbr/________/edm/ord_u/0x3/10.0.0.1/________/________/ord_a/exists',
 value=1, datatype=1),
    #
Item(name='gl/ip-bgp/default/0/100/ord_a/default/nbr/________/edm/ord_u/0x3/10.0.0.1/________/________/ord_b/remote-as',
 value=(0, 26), datatype=5)]
    print(cfg)

    for item in cfg:

        remote_as_pattern = re.match("^gl/ip-bgp/default/0/.*/remote\-as", item.name)
        if remote_as_pattern:
            log.info("BGP remote AS config found")
            remote_as = int(item.value[1])
            if remote_as == 25:
                syslog.info("Attempt to configure BGP remote AS %d" % remote_as)
                precommit.config_error("Remote AS (%d) is not permitted" % remote_as)

    log.info("BGP verification is good")


if __name__ == '__main__':

    result = helper.xrcli_exec("show version")
    match = re.search(r'Version +: (.*)\n*', result['output'])
    print("Image version: %s" % match.group(1))
    verify_bgp()
```

Step 2 Add the script from the remote server or the harddisk: location to the script management repository. See Download the Script to the Router, on page 4.

Step 3 Configure the checksum value to check the script integrity. See Configure Checksum for Precommit Script, on page 5.

Step 4 Activate the script. See Activate Precommit Scripts, on page 7.

Step 5 Configure BGP and commit the configuration.

Example:

```
Router(config)#router bgp 100
Router(config-bgp)#neighbor 10.0.0.1
Router(config-bgp-nbr)#remote-as 25
Router(config-bgp-nbr)#commit
Wed Jan 25 22:53:21.910 UTC
Precommit Script Report Start
----------------------------
Pre-commit Verification Result: Fail
Pre-commit Verification Script verify-bgp.py (req id 1674671641):  Fail
% Script exception return value 1
Errors:
  Remote AS (25) is not permitted
 Warnings:
  BGP AS number (100) not in recommended range (123-234)
----------------------------
Precommit Script Report Done

% Failed to commit .. As an error (Unknown) encountered during commit operation. Changes may not have
 been committed:
'SCRIPT_MGMT' detected the 'fatal' condition 'One or more Pre-Commit script verifications failed'
```

The precommit script is automatically initiated when you commit the configuration. The result from the script run is displayed.

In this example, the precommit script validates the BGP configuration. The AS value limit that is configured in the script is not within the permissible range of 123 to 234. The script rejects the configuration, and displays the details of the validation failure on the console.

Step 6 Verify the script execution details. You can either choose to ignore the script results or view the detailed report of the script execution.

- Ignore the script results using **ignore-results** keyword, and proceed to commit the configuration.

```
Router(config-bgp-nbr)#commit script-verification ignore-results
Wed Jan 25 23:00:02.057 UTC
Precommit Script Report Start
----------------------------
Pre-commit Verification Result: Pass (Failures Ignored)
Pre-commit Verification Script verify-bgp.py (req id 1674671645):  Fail (Ignored)
% Script exception return value 1
Errors:
  Remote AS (25) is not permitted
 Warnings:
  BGP AS number (100) not in recommended range (123-234)
----------------------------
Precommit Script Report Done
```

- View the detailed report using **verbose** keyword.

```
Router(config-bgp-nbr)#commit script-verification verbose
Wed Jan 25 22:53:30.881 UTC
Precommit Script Report Start
----------------------------
Pre-commit Verification Result: Fail
```

```
Pre-commit Verification Script verify-bgp.py (req id 1674671642):  Fail
% Script exception return value 1
Errors:
  Remote AS (25) is not permitted
 Warnings:
  BGP AS number (100) not in recommended range (123-234)
 Script output logs:
/harddisk:/mirror/script-mgmt/logs/verify-bgp.py_precommit_1674671642/stdout.log
Image version: 7.5.4.29I
['!! IOS XR Configuration 7.5.4.29I', 'router bgp 100', ' neighbor 10.0.0.1', '  remote-as 25',
 ' !', '!', 'end', '', '']
[2023-01-25 22:53:31,545] INFO [precommit_verify_bgp]:: BGP config found
!!!!!$$$$$CONFIG WARNING: BGP AS number (100) not in recommended range (123-234) $$$$$!!!!!
[Item(name='gl/ip-bgp/default/0/100/aya', value=1, datatype=1),
Item(name='gl/ip-bgp/default/0/100/gbl/edm/ord_a/running',
value=1, datatype=1),
Item(name='gl/ip-bgp/default/0/100/ord_a/default/nbr/________/edm/ord_u/0x3/10.0.0.1/________/________/aya',
 value=1,
datatype=1),
Item(name='gl/ip-bgp/default/0/100/ord_a/default/nbr/________/edm/ord_u/0x3/10.0.0.1/________/________/
ord_a/exists', value=1, datatype=1),
Item(name='gl/ip-bgp/default/0/100/ord_a/default/nbr/________/edm/ord_u/0x3/10.0.0.1/
________/________/ord_b/remote-as', value=(0, 25), datatype=5)]
[2023-01-25 22:53:31,571] INFO [precommit_verify_bgp]:: BGP remote AS config found
!!!!!$$$$$CONFIG ERROR: Remote AS (25) is not permitted $$$$$!!!!!

Script error logs: /harddisk:/mirror/script-mgmt/logs/verify-bgp.py_precommit_1674671642/stderr.log
Traceback (most recent call last):
  File "/harddisk:/mirror/script-mgmt/precommit/verify-bgp.py", line 107, in <module>
    verify_bgp()
  File "/harddisk:/mirror/script-mgmt/precommit/verify_bgp.py", line 97, in verify_bgp
    precommit.config_error("Remote AS (%d) is not permitted" % remote_as)
  File "infra/script-mgmt/src/Packages/precommit.py", line 87, in config_error
cisco.script_mgmt.precommit.PrecommitConfigError: !!!!!$$$$$CONFIG ERROR: Remote AS (25) is not
 permitted $$$$$!!!!!

------------------------------
Precommit Script Report Done

% Failed to commit .. As an error (Unknown) encountered during commit operation. Changes may not
 have been committed:
'SCRIPT_MGMT' detected the 'fatal' condition 'One or more Pre-Commit script verifications failed'
```

Step 7 Rectify the errors and commit the configuration.

Example:

```
Router(config)#router bgp 200
Router(config-bgp)#neighbor 10.0.0.1
Router(config-bgp-nbr)#remote-as 26
Router(config-bgp-nbr)#commit
Wed Jan 25 22:59:06.704 UTC
Precommit Script Report Start
------------------------------
Pre-commit Verification Result: Pass
Pre-commit Verification Script verify-bgp.py (req id 1674671644):  Pass
------------------------------
Precommit Script Report Done
```

The precommit script validates the BGP configuration to ensure that the conditions configured in the script are met.