Precommit Scripts

Precommit scripts

A precommit script is a configuration management tool that

- executes custom python logic automatically during configuration commit operations,
- validates proposed configuration changes against administrator-defined policies, and
- determines whether to allow or block the commit based on script evaluation results.

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. These scripts are executed automatically during a configuration commit operation. They process the configuration change and act as a 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 play a crucial role in validating configurations during commit operations. These scripts are invoked automatically when a configuration change is committed. Device administrators use them to enforce custom validation rules and simplify repetitive configuration tasks.

During a configuration commit, precommit scripts are automatically initiated to validate changes. If the changes are valid, the script permits the commit. If not, it alerts the administrator to the mismatch and blocks the commit, maintaining device parameters and reducing human error.

Functions of precommit script

Precommit scripts are automatically invoked during a commit operation and can perform several functions.

- The system automatically invokes precommit scripts during a commit operation to validate configuration changes.
- The precommit scripts validate your proposed configuration to ensure it meets the required standards.
- These scripts ensure that any changes to the target configuration stay within system or software boundaries.
- The scripts can estimate the number of Ternary Content Addressable Memory (TCAM) slots that your configuration will require.
- The scripts verify that TCAM usage does not exceed the defined threshold.
- The scripts check that your commit operation follows the required execution rules, such as allowing configuration changes that affect traffic only during specified time intervals.
- If your configuration is invalid, the scripts block the commit operation and display a detailed error message.
- The scripts generate system log messages to help you analyze and troubleshoot failed commit operations.

Restrictions of precommit script

These 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.

Run the precommit script

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

Figure 1: Workflow to run precommit scripts

Before you begin

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.

Procedure

Step 1 Download the script to the router.

Step 2 Configure checksum for precommit script.

Step 3 Activate precommit script.

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

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.

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.

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

Procedure

Step 1 Add the script to the script management repository on the router using one of the two options:

- Add the script from a server.
- Copy the script from an external repository.

a) Add the script from a configured remote server (HTTP, HTTPS, FTP or SCP) or the harddisk location in the router.

Example:

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

You can add a maximum of 10 scripts simultaneously. 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.

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.

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

Example:

```
Router#scp userx@192.0.2.0:/scripts/precommit-bgp.py /harddisk:/
```

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

Example:

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

 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 the precommit script

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. 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.

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
=============================================================================================================

 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
```

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
=============================================================================================================

 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
2. Configure Checksum for Precommit Script

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
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.