Configuration and File System Management

This module describes methods for configuration management and file transfer enhancements.

- Secure file transfer from the Router, on page 1
- Auto-Save Configuration, on page 4
- Auto-Save and Copy Router Configuration Using Public Key Authentication, on page 6
- Increasing Commit Limit, on page 8

Secure file transfer from the Router

Table 1: Feature History Table

Feature Name	Release Information	Feature Description
Secure file transfer from the Router	Release 7.9.1	Your routers are now enabled to transfer files securely to an archive server. It's made possible because the copy command now supports SFTP (Secure File Transfer Protocol) and SCP (Secure Copy Protocol using the underlying SSH protocol implementation. Secure transfer of files from the router maintains the integrity, confidentiality, and availability of network configurations. This feature modifies the **copy** command.

You can duplicate files or data in the router from one location to another using the **copy** command. This functionality helps to create a copy of a file, folder, or data set and place it in a specific destination. You can use the copy functionality to back up files, move data between directories, create duplicates of the files for editing or distribution without modifying the original content. It also allows you to retain the original data while making a duplicate that you can further manipulate independently.

Starting with Cisco IOS XR Release 7.9.1, we've enhanced the functionality of the copy command to support secure file transfer from the router. Secure file transfer protects data during transit using the SFTP (Secure

File Transfer Protocol) and SCP (Secure Copy Protocol) when sharing files within or across networks. The SFTP and SCP functionalities in the copy feature use the SSH protocol implementation in the router to secure transfer the files to a remote server.

You can use the following options in the **copy** command for secure file transfer:

- **sftp:** You can transfer the files to a remote location using the **SFTP** file transfer protocol. SFTP is a secure file transfer protocol for transferring large files.
- **scp:** You can transfer the files to a remote location using the **SCP** file transfer protocol. SCP is a secure copy protocol to transfer files between servers.

Starting Cisco IOS XR Software Release 7.10.1, you can use public-key authentication while copying the running configuration.

Prerequisites:

Enable the SSH Server in the router as follows:

```
Router# config
Router(config)# ssh server v2
Router(config)# ssh server vrf default
Router(config)# ssh server netconf vrf default
Router(config)# commit
```

Configuration Example for Secure File Transfer Protocol

You can copy the running configuration file from the router to a remote server using SFTP as follows:

Configuration in the Router

```
Router# copy running-config sftp://root:testpassword@192.0.2.1//var/opt/run_conf_sftp.txt

Destination file name (control-c to cancel): [/var/opt/run_conf_sftp.txt]?

.
215 lines built in 1 second
[OK]Connecting to 192.0.2.1...22
Password:
sftp> put /tmp/tmpsymlink/nvgen-34606-_proc_34606_fd_75 /var/opt/run_conf_sftp.txt


/tmp/tmpsymlink/nvgen-34606-_proc_34606_fd_75

  Transferred 3271 Bytes
  3271 bytes copied in 0 sec (3271000)bytes/sec
sftp> exit
```

Verification in the SFTP Server

```
[root@sftp_server ~]# ls -ltr /var/opt/run_conf_sftp.txt
-rw-r--r-- 1 root root 3271 Mar 21 18:07 /var/opt/run_conf_sftp.txt
```

Configuration Example for Secure Copy Protocol

You can copy the running configuration file from the router to a remote server using SCP as follows:

Configuration in the Router

```
Router# copy running-config scp://root:testpassword@192.0.4.2//var/opt/run_conf_scp.txt

Destination file name (control-c to cancel): [/var/opt/run_conf_scp.txt]?
```

```
.
215 lines built in 1 second
[OK]Connecting to 192.0.4.2...22
Password:

  Transferred 3271 Bytes
  3271 bytes copied in 0 sec (0)bytes/sec
```

Verification in the SCP Server

```
[root@scp_server ~]# ls -ltr /var/opt/run_conf_scp.txt
-rw-r--r-- 1 root root 3271 Mar 21 18:07 /var/opt/run_conf_scp.txt
```

Configuration Example for SCP and SFTP Using Public-Key Authentication

While you're using public-key authentication for copying running configuration from the router to a remote server, you don't need to mention **password** in the command. The following example shows how you can configure public-key authentication while copying configuration using the SCP protocol:

```
Router#copy running-config scp://root@192.0.4.2//var/opt/run_conf_scp.txt
```

Auto-Save Configuration

Table 2: Feature History Table

Feature Name	Release Information	Feature Description
Auto-Save with Secure File-Transfer and Additional Configurable Parameters	Release 7.9.1	Apart from automatically backing up the running configuration after every commit, you can also do the following with Auto-Save: • Save running configurations to remote systems using Secure Copy Protocol (SCP) and Secure File Transfer Protocol (SFTP). • Configure wait-time between two subsequent auto-saves. • Append time-stamp to the file name of the saved configuration. • Save the encrypted password. • Specify the maximum number of files that you can auto-save. The feature introduces these changes: CLI: Modified the **configuration commit auto-save**command by adding the following keywords: • **filename scp** • **filename sftp** • **wait-time** • **timestamp** • **password** • **maximum** Yang Data Model: • New XPaths for Cisco-IOS-XR-config-autosave-cfg • New XPaths for Cisco-IOS-XR-um-config-commit-cfg

You can configure the router to automatically take the backup of the running configuration by using **configuration commit auto-save** command. This auto-save feature saves the configuration to the specified location on the router after every **commit** is made. These auto-save files are stored in the form of Linux files.

Starting Cisco IOS XR Software Release 7.9.1, the auto-save feature is enhanced to provide a set of functionalities. Use the following keywords to achieve the same:

- **scp and sftp** - You can save the running configuration backup files to remote location using **scp** and **sftp** file transfer protocols. SCP is a secure copy protocol to transfer files between servers. Whereas SFTP is a secure file transfer protocol for transfering large files.
- **password** - You can can save encrypted passwords for the remote and non-remote URLs.
- **maximum** - You can mention maximum number of files that can be saved automatically. Once the maximum number of auto-saved file is reached, the newer auto-save files starts replacing the older auto-save files. The default value of **maximum** is 1. You can save upto 4294967295 files.
- **timestamp** - Using this keyword, the time-stamp can be appended to the auto-saved configuration file name. The **timestamp** uses the time and timezone configured on the router. The saved file displays timestamp in <day> <month> <date> <hours> <minutes> <seconds> <milliseconds> format. Here is an example of auto-saved file with time-stamp - *: test_123.autosave.1.ts.Tue_Jan_31_15-15-51_805_IST*
- **wait-time** - You can specify how long to wait before the next auto-save happens in terms of days, months, or hours after the commit is made. The default value of **wait-time** is zero.

Restriction for Auto-Save Configuration

The auto-save configuration is only available on the local paths, scp, and sftp paths.

Configure Auto-Save

Use the **configuration commit auto-save** command to auto save the configuration.

```
Router#configure
Router(config)#configuration commit auto-save
Router(config-cfg-autosave)#commit
```

You can also configure options such as **password**, **timestamp**, **maximum**, and **wait-time** with the **configuration commit auto-save** command. The location to save the file-name must be specified in <protocol>://<user>@<host>:<port>/<url-path>/<file-name> format.

When filename is accessed through VRF, you can specify filename in **filename** <protocol>://<user>@<host>:<port>;<vrf name>/<url-path>/<file-name> format.

When you are using public key authentication, you don't need to mention **password**.

```
Router(config-cfg-autosave)#configuration commit auto-save filename
sftp://user1@server1://test-folder/test_123
Router(config-cfg-autosave)#password clear encryption-default cisco
Router(config-cfg-autosave)#timestamp
Router(config-cfg-autosave)#maximum 10
Router(config-cfg-autosave)#wait-time days 0 hours 0 minutes 0 seconds 5
Router(config-cfg-autosave)#commit
```

Running Configuration

```
Router#show running-config configuration commit auto-save
configuration commit auto-save
 filename sftp://user1@server1://test-folder/test_123
 password encrypted encryption-default <password for above user>
 timestamp
 maximum 10
 wait-time days 0 hours 0 minutes 0 seconds 5
!
```

Auto-Save and Copy Router Configuration Using Public Key Authentication

Table 3: Feature History Table

Feature Name	Release Information	Feature Description
Auto-Save and Copy Router Configuration Using Public Key Authentication	Release 7.10.1	You can now experience passwordless authentication while automatically saving running configurations and securely copying them on the router. The feature uses public key-based authentication, a secure logging method using a secure shell (SSH), which provides increased data security. This feature offers automatic authentication and single sign-on benefits, which also aids in a secure automation process. This feature modifies **configuration commit auto-save** and **copy** command to support password-less authentication.

From Cisco IOS XR Software Release 7.10.1, you don't need to remember and enter the **password** as you can use public key-based authentication while doing the following:

- Automatically saving your running configuration
- Copying the configuration from a source (such as a network server) to a destination (such as a flash disk in a router)

Password is automatically verified when you have enabled SSH connection using public key-based authentication. Using public key-based authentication avoids several problems such as password disclosure and password leakage.

Public key is mathematically related to private key. The private key is secret, whereas the public key is available on the servers. You can copy the public key to the SSH server from the SSH client. Then, when you try to secure the running configuration, the SSH server tries to authenticate by generating a challenge using the public key. Only the private key can answer this challenge. As the keys are related, log-in is successful.

Prerequisites for Auto-Save and Copy Router Configuration Using Public Key Authentication

Ensure you have enabled public key-based authentication of SSH clients, using the following steps:

- Generate RSA key pair on the router configured as the SSH client. Use the **cyrpto key generate authentication-ssh rsa** command to generate the RSA key pair.
- Use the **show crypto key mypubkey authentication-ssh rsa** command to view the details of the RSA key. The key value starts with *ssh-rsa* in this output.
- Copy the RSA public key from the SSH client to the SSH server:

  You can do this either by logging in to the remote SSH server with your established user credentials, or have a system administrator on the remote system add the key on the SSH server.

  If the SSH server is a Cisco IOS XR router, then you can use the **crypto key import authentication rsa** command on the router prompt of the server to import the key from the SSH client. You will then be prompted to enter the public key.

  If the SSH server is a Linux server, then you must add the public key to the `~/.ssh/authorized_keys` file of the respective user account in that server. This file contains a list of all authorized public keys on that server.

For more detailed information on how to enable SSH connection using public-key based authentication, see *Public Key Based Authentication of SSH Clients* in System Security Configuration Guide for Cisco ASR 9000 Series Routers.

Configuration Example for Auto-Save Using Public Key Authentication

When you are using public key authentication, you don't need to mention **password**.

```
Router(config-cfg-autosave)#configuration commit auto-save filename
sftp://user1@server1://test-folder/test_123
Router(config-cfg-autosave)#timestamp
Router(config-cfg-autosave)#maximum 10
Router(config-cfg-autosave)#wait-time days 0 hours 0 minutes 0 seconds 5
Router(config-cfg-autosave)#commit
```

Running Configuration

```
Router#show running-config configuration commit auto-save
configuration commit auto-save
 filename sftp://user1@server1://test-folder/test_123
 timestamp
 maximum 10
 wait-time days 0 hours 0 minutes 0 seconds 5
!
```

Increasing Commit Limit

Table 4: Feature History Table

Feature Name	Release Information	Feature Description
Increasing Commit Limit	Release 24.2.1	The maximum number of commits is increased in the router that allows you to configure complex topology changes without interruptions caused by the default blocking of commit changes during rebase or ASCII backup operations. You can prevent the commit operation from getting blocked by using the **cfs check** command, which increases the commit (pacount) count from 20 to 40, and the commit file diff size (configuration data) from 2 MB to 4 MB, and by using the **clear configuration ascii inconsistency** command, which performs an ASCII backup after 55 minutes. The feature modifies the following commands: • **cfs check** • **clear configuration ascii inconsistency**

The Cisco IOS XR Routers use a two-stage configuration model. The first stage is target configuration, where you build the configurations using the necessary commands in the command line interface. The second stage is the commit, where the configuration made in the target stage is added to the router configuration using the **commit** command. After each commit, the router generates a file for the newly configured changes and adds it to its running configuration, making it an integral part of the running configuration.

Note This target configuration doesn’t impact the router's running configuration.

The Cisco IOS XR routers perform rebase and ASCII backup operations to maintain the real time configuration in the backup copy. The rebase and ASCII backup operations block you from committing configurations to the router.

In rebase, the router automatically saves your changes to the backup binary configuration file after 20 commits, or 2 MB of configuration data. The router blocks the commit while saving the configuration to the backup file. The router takes a few seconds to complete the rebase operation, during which, if you terminate the CLI session, the router loses the target configurations in the blocked commit.

In ASCII backup, the router automatically saves a copy of its running configuration in the ASCII format. This backup process takes place if there has been a commit to the router configuration and when the ASCII backup timer completes a 55-minute window after the previous backup event. However, if there was no commit when the ASCII backup timer completes 55 minutes, the counter is reset without any backup. During the ASCII backup, the router blocks the configuration commits.

Starting with Release 24.2.1, we have made the following enhancements:

- You can use the **cfs check** command to increase the rebase limits in the router from 20 to 40 commits and the configuration data from 2 MB to 4 MB. When configuring the router, you can check the current commit count and configuration data size using the **show cfgmgr commitdb** command. If the commit count is 20 or higher, or the configuration data size is 2 MB or above, the router initiates a rebase within 10 seconds. By using the **cfs check** command to increase the commit count to 40 and the configuration data to 4 MB, you can commit without delay.
- You can use the **clear configuration ascii inconsistency** command to perform an ASCII backup and reset the ASCII backup timer to zero. Once the backup is complete, the router will automatically initiate the next periodic ASCII backup operation only after 55 minutes from the time the **clear configuration ascii inconsistency** command is executed.

Guidelines and Restrictions for Increasing the Commit Limit

- The **cfs check** command increases the rebase limits only for one instance. After executing the **cfs check** command, the router will perform a rebase operation after 40 commits or when the configuration data reaches 4 MB. Once the router performs a rebase operation, the limits will reset to the default values of 20 commits and 2 MB configuration data. To enable 40 commits and 4 MB configuration data, you must perform the **cfs check** command again.
- After executing the **cfs check** command, if a router switches over to standby RP, the rebase limits are retained as 40 commits and configuration data of 4 MB. However, if the router reloads, the rebase limits are reset to 20 commits and 2 MB of configuration data. For example, after executing the **cfs check** command, if the router switches over to standby RP after 30 commits, it will still have ten more commits before a rebase. However, if the router reloads, the rebase limits are reset to default 20 commits and 2 MB of configuration data.
- The **clear configuration ascii inconsistency** command initiates an ASCII backup and resets the ASCII backup timer count to zero. Following this, the router will automatically initiate the next periodic ASCII backup operation only after 55 minutes from the time **clear configuration ascii inconsistency** command is executed. For example, if you execute a commit operation after executing a **clear configuration ascii inconsistency** command, the router will perform an ASCII backup operation 55 minutes after the**clear configuration ascii inconsistency** command was executed, and merge the new commit into ASCII backup. Hence, before the next 55 minutes, you must execute the **clear configuration ascii inconsistency** command again to reset the ASCII backup timer to zero.
- When the router enters standby mode or reloads, the ASCII timer does not reset to zero, and the router performs an ASCII backup operation 55 minutes after the first commit operation before the standby mode or reload.
- Cisco does not recommend executing **clear configuration inconsistency** and **clear configuration ascii inconsistency** commands regularly after each commit, as it causes hard disk wear and tear. You should execute these commands only before a commit or sequence of commits that must be done within a specific timeframe and without being delayed by rebase and ASCII backup operations. As these commands

perform disk input and output operations in the background, frequent execution of these commands causes frequent access to the hard disk, which increases the wear and tear on the hard disk.

Increasing the Rebase Limits

You can increase the rebase limits as follows:

1. Use the **cfs check** command to increase the commit count to 40 and the configuration data to 4 MB.

```
Router# cfs check
Creating any missing directories in Configuration File system...OK
Initializing Configuration Version Manager...OK
Syncing commit database with running configuration...OK
```

2. Verify if the **cfs check** command is executed using the **show configuration history** command.

```
Router# show configuration history last 5
Sno.  Event      Info                          Time Stamp
~~~~  ~~~~~      ~~~~                          ~~~~~~~~~~
1     cfs check  completed                     Wed Jan 10 11:42:21 2024
2     commit     id 1000000001                 Wed Jan 10 11:39:26 2024
3     startup    configuration applied         Wed Jan 10 11:39:02 2024
```

Perform ASCII Backup and Rest ASCII Backup Timer

You can perform ASCII backup and rest ASCII backup timer as follows:

1. Use the **clear configuration ascii inconsistency** command to perform ASCII backup at that instance and reset the ASCII backup timer count to zero.

```
Router# clear configuration ascii inconsistency
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! Warning: !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!! It is recommended to run this command only when all nodes in router      !!!!
!!!! are in IOS-XR RUN state. To determine node state, run following command: !!!!
!!!! 'show platform'.                                                        !!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! Warning: !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Proceed with the command ?[confirm] y
 Ascii configuration backup is in progress...
Configuration ascii backup complete
```

2. Verify if the **clear configuration ascii inconsistency** command is executed using the **show configuration history** command.

```
Router# show configuration history last 5
Sno.  Event      Info                          Time Stam
~~~~  ~~~~~      ~~~~                          ~~~~~~~~~~
1     backup     Periodic ASCII backup         Wed Jan 10 11:48:20 2024
2     cfs check  completed                     Wed Jan 10 11:42:21 2024
3     commit     id 1000000001                 Wed Jan 10 11:39:26 2024
4     startup    configuration applied         Wed Jan 10 11:39:02 2024
```