Redundancy Configuration Manager – Configuration and Administration Guide, Release 2022.01

First Published: 2022-02-14

Last Updated: 2022-07-28

Contents

Overview

> **NOTE**: The documentation set for this product strives to use bias-free language. For purposes of this documentation set, bias-free is defined as language that does not imply discrimination based on age, disability, gender, racial identity, ethnic identity, sexual orientation, socioeconomic status, and intersectionality. Exceptions may be present in the documentation due to language that is hardcoded in the user interfaces of the product software, language used based on RFP documentation, or language that is used by a referenced third-party product.

The Redundancy Configuration Manager (RCM) is a Cisco proprietary node/network function (NF) that provides redundancy of StarOS-based UP/UPFs. The RCM provides N:M redundancy of UP/UPFs wherein "N" is number of Active UP/UPFs and is less than 10, and "M" is the number of Standby UP/UPF in the redundancy group.

The RCM is developed using the Subscriber Microservices Infrastructure (SMI), a Cloud Native (CN) application. The RCM is delivered both as VM-based and CN-based image. For 4G, the RCM VM image is collocated with UPs and deployed as a VM. For 5G, the RCM is deployed as CN application leveraging the SMI Cluster Manager.

For details about RCM integration with SMI Cluster Manager, refer to the RCM and SMI Cluster Manager Integration chapter.

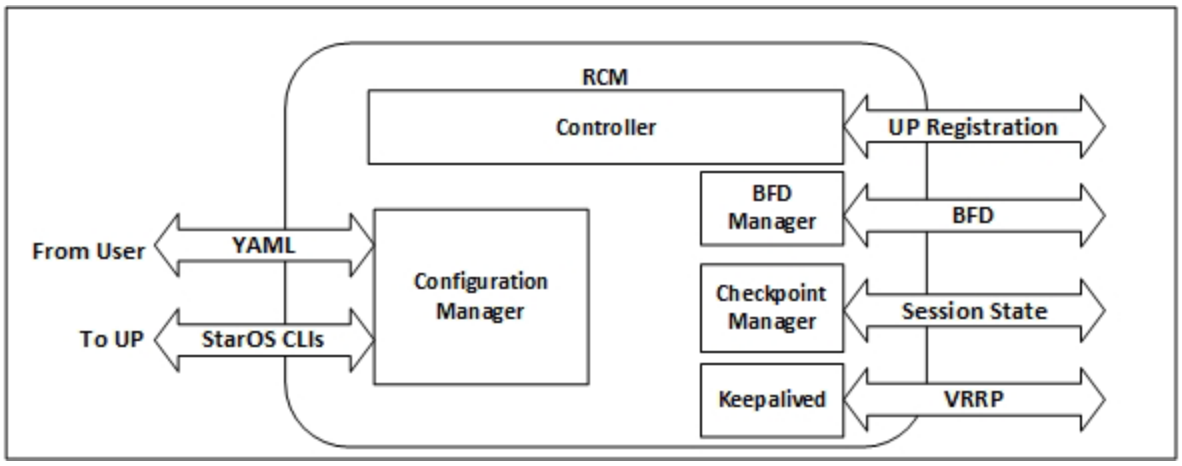
The RCM supports the following functions:

- Redundancy Management
  - Monitors the health of each active UP/UPF
  - Selects a standby UP/UPF to become active when an active UP/UPF becomes unreachable
  - Stores session checkpoint information from active UPs/UPFs
  - Sends session checkpoints to new active UP/UPF during switchover
- Configuration Management
  - Decides the role of a UP/UPF; Active or Standby
  - Stores the configuration data that is sent to a UP/UPF
  - Provides the configuration data to a new UP/UPF (includes Day-0 to Day-N):
    - **Day-0**: Configuration required on UP/UPF with local context, management IP, and other static configurations other than ECS, APN and host-specific configurations (for example, bulkstats, thresholds, and so on.)
    - **Day-0.5** (RCM Context configurations): Configuration required to connect to UP/UPF.

    Note: Day-0 and Day-0.5 configuration is outside the scope of RCM and must be configured in UP/UPF using ESC/CM/NSO/CFP. Contact your Cisco Account representative for more details.
  - Determines the Control Planes (CPs)/Session Management Functions (SMFs) that register with the UPs/UPFs.

Components

The following diagram illustrates the components of RCM.



The RCM comprises of the following components which runs as pods in the RCM VM:

- **Controller**: It communicates event-specific decisions with all the other pods in RCM.
- **BFD Manager** (**BFDMgr**): It uses the BFD protocol to identify the state of data plane.
- **Configuration Manager** (**ConfigMgr**): It loads the requested configuration to the UPs.
- **Redundancy Manager** (**RedMgr**): It is also called the Checkpoint Manager; it stores and sends the checkpoint data to a standby UP.
- **Keepalived**: It communicates between Active and Standby RCM using VRRP.

Minimum Requirements

Its recommended that the RCM, which runs as a Kubernetes cluster on a VM, meets the minimum hardware and software requirements provided in the following tables.

For sizing parameters and recommendations for typical deployment setup, see Appendix A: Deployment Parameters.

Hardware Requirements

Node Type	CPU Cores	RAM	Storage
RCM	(2 + No. of SessMgrs in one UP) x 2.60 GHz	8GB + 4GB for each active UP	40GB HDD

Software Requirements

VIM	RCM Component Operating System
VMware – ESXI 5.5 or later OpenStack 13 or later	Ubuntu 18.04

NOTE: The above VM recommendation is validated for a single User Plane group of 12 (10:2) UPs.

Deploying the RCM

For 5G deployments, see the RCM and SMI Cluster Manager Integration chapter.

For 4G CUPS deployments, the RCM is deployed as VNF in a single VM. The following VM image types are available for deployment.

Image	Description
rcm-vm-airgap.<release>.ova.SPA.tgz	The RCM software image that is used to on-board the software directly into VMware.
rcm-vm-airgap.<release>.qcow2.SPA.tgz	The RCM software image in a format that can be loaded directly with KVM using an XML definition file, or with OpenStack.
rcm-vm-airgap.<release>.vmdk.SPA.tgz	The RCM virtual machine disk image software for use with VMware deployments.

To deploy the RCM:

1. Download the RCM image tar file that corresponds to your VIM type.
2. Unpack the RCM image from the tarfile related to your VIM.
3. Onboard the RCM image into your VIM per the instructions for your VIM.
4. Create a VM flavor based on the information in Hardware Requirements section.
5. Create cloud-init configuration per your VIM requirements. See RCM VM Deployment Procedure section.
6. Deploy the RCM image to the compute per the instructions for your VIM.
7. Proceed to “Configuring the RCM”.

RCM VM Deployment Procedure

The following steps describes the procedure to deploy the RCM VM:

1. Initialize the RCM VM instance using the cloud-init file.

   Following is a sample configuration from cloud-init file (user_data.yaml):

```
#cloud-config
users:
  - default
  # password generated via: mkpasswd --method=SHA-512 --rounds=4096
  - name: luser
    gecos: luser
    primary_group: luser
```

```
    groups: [admin, adm, audio, cdrom, dialout, docker, floppy, luser, video,
plugdev, dip, netdev]
    lock_passwd: false
    passwd:
$6$rounds=4096$Zl7FnVp7dYlT0bw$CFqAVNA8M1wxaEXwAVYlKfekSrTXbGUne1ihJ11xdNk29b
fCk2AURQG4XV.LnFNX6MmsW9OPvFxDZpeapXkYG.
    shell: /bin/bash
    home: /home/luser
ntp:
    servers:
     - 10.84.96.130
     - 10.84.98.2
     - 10.81.254.131
manage_etc_hosts: localhost


ssh_pwauth: yes
```

2. Create the cloud-locald rcm-seed image from the user_data file using the following command. The coomand can be executed from any local server or a jump server from where the deployment is carried out:

cloud-localds -H rcm rcm-seed.img user_data.yaml

Where:

- **cloud-localds**: Creates a disk for cloud-init to utilize nocloud.
- **-H**: Specifies the host name.

Usage/Syntax:

cloud-localds [options] output user-data [meta-data]

3. Attach the rcm-seed.img as CDROM to the VM.
4. After the VM is up:
   a. Ensure the hostname of RCM VM is added to */etc/hosts*

   For example: **127.0.1.1 rcm** *rcm*

   Where *rcm* is the hostname.

   b. Ensure that the kubeadm_init, kubelet, calico_init, helm, and init_cluster services are running. You can verify by executing the following command:

   c. systemctl status kubelet/helm/init_cluster/calico_init

NOTE: Allow additional time (approximately 15 minutes) for Ops Center to come up on first boot. The extra time is required for the system startup to run certain hardening scripts that makes the system more secure.

You can check the following system service status. Excluding kubelet and helm, for which the status will display as Active (Running), status for all other services will display as Active (Exited):

- **systemctl status kubelet**

- **systemctl status kubeadm_init**
- **systemctl status calico_init**
- **systemctl status hardening**
- **systemctl status helm**
- **systemctl status init_cluster**

The Ops Center pods comes up after the status of all the services are Active.

d. For kubectl commands to work for a non-root user, please execute the following (this example assumes the user is called "luser"):

```
mkdir ~luser/.kube
sudo cp /etc/kubernetes/admin.conf ~luser/.kube/config
sudo chown luser:luser ~luser/.kube/config
```

e. Setup the helm using:

```
helm init --client-only --kubeconfig /etc/kubernetes/admin.conf --stable-repo-url http://10.42.1.1:9000
```

NOTE: The "http://10.42.1.1" is the location of docker images for RCM which is used by Helm. It is a local URL inside the VM.

f. RCM ops-center pods can be seen in the RCM namespace:

kubectl get pods -n rcm

g. Configure the UP username and password, in Ops Center, that are passed to ConfigMgr. For details, see Configuring UPF Credentials section.

NOTE: Without configuring the UPF credentials, the ConfigMgr pod will not be in running state.

5. Configure the Ops Center to bring up the RCM component pods:

a. Get ops-center pod name by executing below command

kubectl get pods -n *rcm_namespace* **| grep ops-center**

b. Reset the password of the Ops Center using the following command:

kubectl exec -ti *ops_center_pod* **reset-admin -n rcm**

For example: *kubectl exec -ti ops-center-rcm-ops-center-d6d9cf976-jmght reset-admin -n rcm*

c. Access the Ops Center using the password created in Step 5b.

d. Access the Ops Center CLI using ops-center service IP:

kubectl get svc -n rcm

ssh -p 2024 admin@*svc_ip*

For example: `ops-center-rcm-ops-center ClusterIP 10.43.213.124<none> 8008/TCP,8080/TCP,2024/TCP,2022/TCP,7681/TCP 39h`

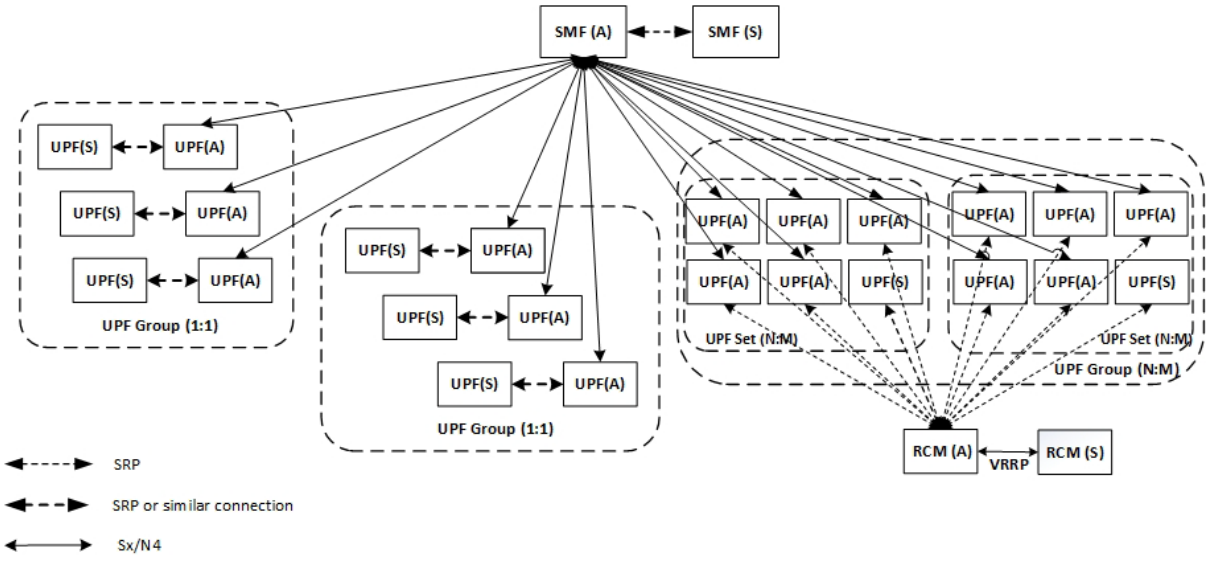
6. After Ops Center CLI is configured, run RCM components using the following command in Global Configuration mode:

system mode running

To remove RCM components: **system mode shutdown**

RCM Deployment Architecture

The following diagram illustrates a typical RCM deployment model.



- It is co-located with all the UPs it manages; RCM can manage multiple independent UP groups.

  Note that the UP groups in 1:1 redundancy mode is through ICSR and are not controlled by RCM.
- It has a high-performance network between itself and its UPs (SR-IOV or equivalent). Same is the case with Keepalive (VRRP) network between RCM HA instances.
- It monitors its UPs for failures and initiates failover to a standby UP.
- It quickly detects a UP failure using the multi-hop BFD protocol.
- It pre-configures (excluding CP and UPs Day-0 and Day-0.5 configuration as they are part of RCM) standby UPs with all the active host-specific configuration.
- Its user interface (UI) displays the state or health of a UP, stats, historical events (logs), and triggers a manual failover. In addition to monitoring, the UI supports the addition and removal of UPs and/or UP groups.
- If an RCM is configured without a redundant RCM, then the UPs under its control continue to run and provide services (not redundantly).
- If RCM restarts and connects to UPs, then the RCM re-learns or audit state from the existing UPs without disrupting services.

How it Works

This section describes how the RCM functions work:

- Redundancy Management
- Configuration Management

Redundancy Management

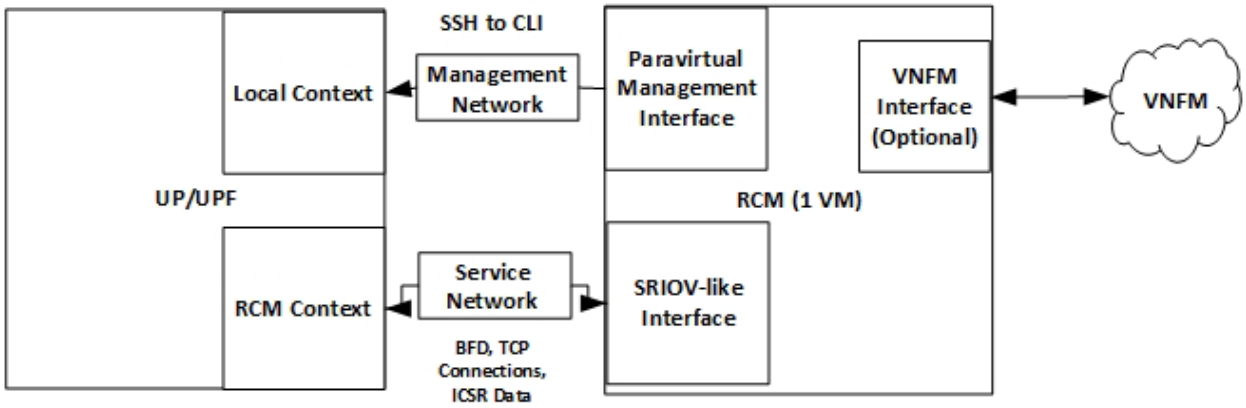
N:M redundancy using RCM, where N > 1 and M is at least 1, is a mechanism that is required to store all the required information at a common location. This common information is properly segregated based on “N” User Planes (UPs)/User Plane Functions (UPFs), and each UP/UPF contains “x” Session Managers (SessMgrs). On a switchover trigger, one of the “M” UPs/UPFs available as standby is selected to receive the appropriate data from the common location. This functionality is achieved through RCM.

To support N:M UP redundancy, the RCM provides the following functionality:

- Procedure to acquire the dynamic configuration based on the configuration changes. The Configuration Manager performs this procedure and pushes the configuration according to the state of the UP (active or standby).
- Method to store and retrieve the checkpoint data. It stores the checkpoint data because the exact standby for each active UP is not pre-allocated. When an active UP fail, the Redundancy Manager (RedMgr) pods send the stored checkpoint data to the newly selected UP from the available pool of standby UPs.
- Mechanism to detect the failure of a UP. A Bidirectional Forwarding Detection (BFD) Manager (BFDMgr) pod monitors the UP/UPFs and detects failures.
- Enables the RCM Controller pod to take decisions and instruct the other pods depending on the UP discovery or failure. Based on number of events, the Controller also controls the actions of the three managers: Configuration Manager, Redundancy Manager, and BFD Manager.
- Maintain information about the IP pool.

Architecture

The following diagram illustrates the logical network layout for a single VM RCM.



The RCM node is built on Subscriber Management Infrastructure (SMI). SMI provides the Kubernetes (K8s) infrastructure, which is used to bring all the required pods to support the redundancy for UPs.

The Ubuntu cloud image supports cloud-init based configuration (https://cloud-init.io). You must use cloud-init to configure local user accounts (ssh key or password-based) and network configuration.

RCM services (rcm-controller, ConfigMgr, RedMgr) are prepopulated in the disk image. An RCM-specific Ops Center provides the user interface to RCM. The RCM services are started on bootup and monitored via K8s liveliness.

In RCM, the VM requires multiple network interfaces. One interface for "management" connectivity (Ubuntu SSH daemon and Ops Center access), and another interface is for "service" connectivity (session state over this interface). The "management" interface has low throughput requirement and so, paravirtual NICs can be used. The "service" interface needs to be high speed and so, a SRIOV type interface is required. One optional "VNFM" interface can be used as pingable interface for ESC. Cloud-init is used to configure these NICs. Cloud-init configuration and NIC naming can vary depending on the environment

For deployment information, refer RCM Deployment section.

Redundancy with Cloud Native

The N:M redundancy with Cloud Native support involves:

- A procedure to acquire the dynamic configuration based on the configuration changes.

  This is performed by the ConfigMgr. It pushes the configuration according to the state of the UP (Active or Standby).
- A way to store and retrieve the checkpoint data. It stores the data as the exact Standby for each Active is not pre-allocated. On Active failure, the checkpoint data is pushed to the newly-selected Active from the available pool of Standby.

  This is performed by the RedMgr (CheckPointMgr) pods.
- A mechanism to detect the failure of UP. A BFDMgr pod monitors all the UPs and detects failure.
- A Controller pod determines and instructs action mechanism to other pods based on UP discovery and failure. Controller also controls the actions of all the three managers on several events.

Upgrade Process

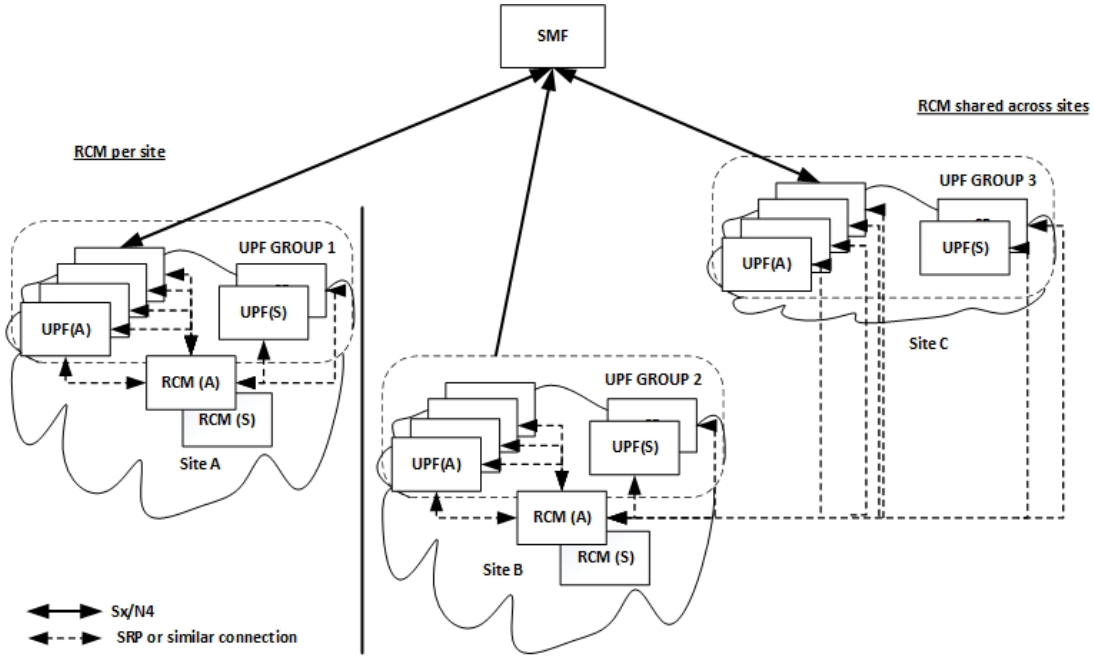
All the required RCM images are bundled as offline tar images and can be downloaded from the software repository. You can perform the rolling upgrades through Common Execution Environment (CEE) Ops Center.

For details, refer the *Upgrading CEE Products* section in the *UCC SMI Common Execution Environment Configuration and Administration Guide*.

For VM-based application, rolling upgrade of components is currently not supported. Any updates to Ubuntu packages or RCM components are delivered through a new disk image. You must delete the current RCM instance, spawn a new RCM instance based on the new disk image, and then configure the new instance.

For upgrade/downgrade of RCM in HA, see RCM Upgrade and Downgrade Procedure in HA.

N:M Redundancy Architecture



In 4G networks, UPs and CPs communicate over the Sx interface. In 5G, the UPFs and SMFs communicate over N4 interface. The UP is user plane which could be a PGW-U, SGW-U, SAEGW-U, or UP(F). A group of UPs that are deployed to support the same configuration are said to be a part of same UP group (UPG). Therefore, for an N:M redundancy configuration, the UP group is comprised of (N+M) UPs that are of the same capacity and capability.

The UP transfers the checkpoint data to Redundancy Manager pods on the RCM. These pods receive the checkpoint data, store them, and segregate them properly so that each checkpoint is retrieved or modified properly. The checkpoint data reuses the same mechanism of Service Redundancy Protocol (SRP) on the UP to transfer the data to achieve the redundancy.

During switchover, the RCM Controller indicates to all the Redundancy Manager pods to push the data of the failed UP to a new UP. First, the Controller pushes the pool-chunk information to the newly selected UP. Secondly, the call records for all the subscribers are pushed to the corresponding Session Managers. Lastly, the Controller itself pushes the route modifier and IP pool information to the new UP. On receiving all the required data, the new UP becomes active and services new calls along with the existing calls.

To ensure faster data retrieval, it's stored in memory. The checkpoint data that the Redundancy Manager receives is stored in the RAM. After careful segregation based on UP, session, and call, the checkpoint data is pushed to the new active UP after a switchover from the RAM.

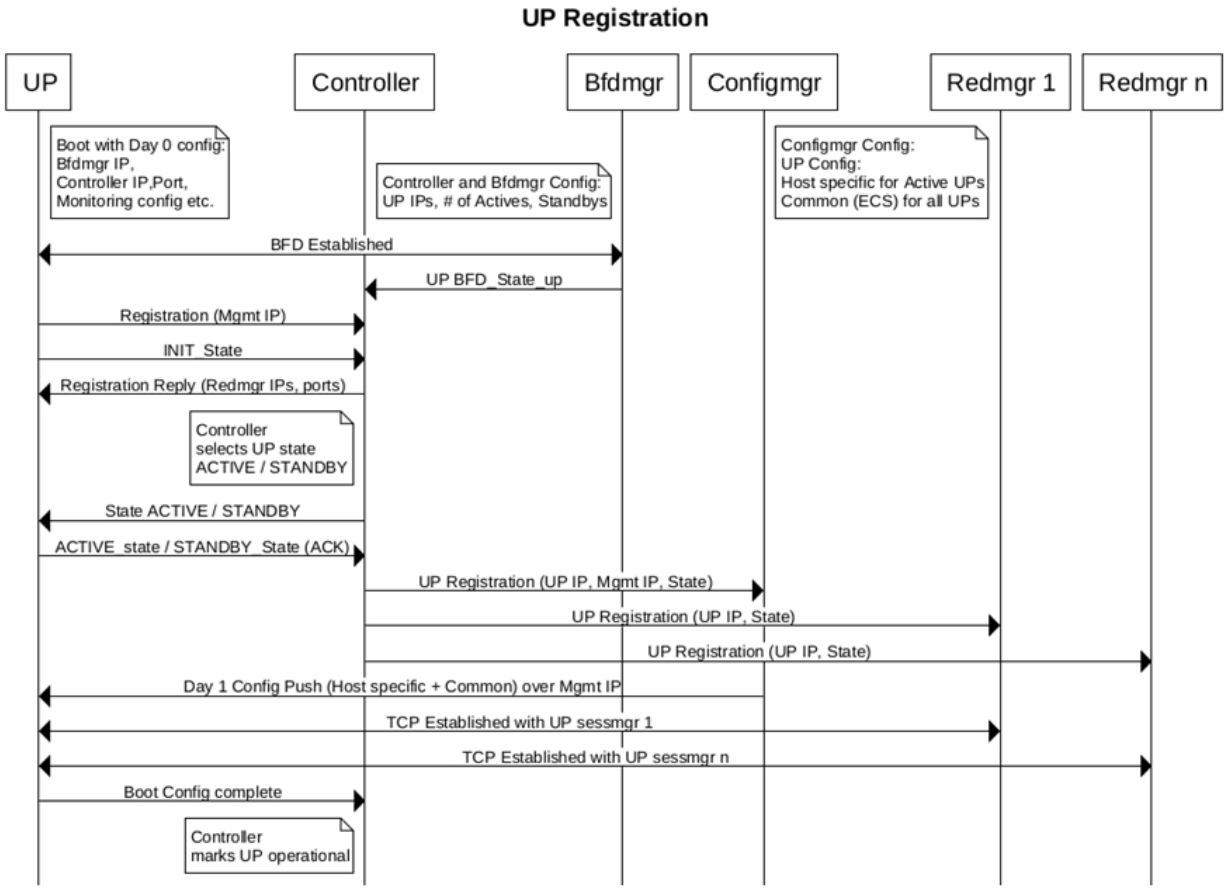
The BFD Manager detects the failure of a UP and informs the controller about the failure. Then, the Controller selects a new UP from the available pool of 'm' standby. The Controller also instructs all the Redundancy Managers to transfer the checkpoint data to the new active UP. It also handles the synchronization of data from various Redundancy Managers to the new UP.

The RCM supports multiple UP groups and segregates based on the UP group. If there is any failure notification, the Controller also identifies the UP. The Session Manager of the UP connects and stores the

checkpoint data in the same Redundancy Manager pod. To achieve this result, the Redundancy Manager pod maintains the same IP and port even after a pod reboot.

UP Registration Call Flow

The following call flow illustrates the registration of UP.



Operational Flow

Boot-up Sequence of Pods

On configuring redundancy, the Subscriber Microservices Infrastructure (SMI) brings up the following pods (not in any particular order):

- Configuration Manager Pod
- Redundancy Manager Pods: These are a set of Redundancy Managers based on the UP group and number of Sessions Managers tasks that need support. Ideally this indicates the number of Session Managers tasks in each UP.
- BFD Manager Pod: This is a UP/UPF monitoring pod that monitors all the available UP/UPFs in a redundancy group.
- Controller Pod: This pod also comes up about the same time as all other pods.

Meanwhile, each UP comes up independently. Only the endpoint address of the RCM is configured for each UP.

When each pod comes up, the Controller is notified about the existence of each pod. The Configuration Manager indicates to the Controller about its presence. The Configuration Manager also indicates the readiness to push the configuration, on UP registration. When the Redundancy Manager pods come up, their presence is notified to the Controller. The Controller establishes a communication mechanism with all the Redundancy Manager pods. After the UP starts sending the checkpoint information to each Redundancy Manager, it updates the Controller about the hosting of the checkpoint data and the Session Manager instance. The BFD Manager also establishes the connection with the Controller and exchanges a handshake with the Controller.

UP Bootup Sequence

The following is the UP bootup sequence when the RCM is used:

1. The UP sends Init state to the RCM Controller.
2. The RCM Controller determines to make it Active or Standby and sets the state as PendActive or PendStandby in its data structures (The PendStandby is not available for switchover selection).
3. The RCM Controller sends State (Active or Standby), Host ID (Host string), and Route Modifier to the UP.
4. The UP responds to the RCM Controller acknowledging the State and Host ID.
5. The UP stores or updates its State and Host ID in Shared/System Configuration Task (SCT).

   SCT is a StarOS task for configuring system parameters, retrieving information, and notifying system components of configuration changes.
6. The RCM Controller sends the UP's Endpoint, State, and Host ID notification to its Configuration Manager service.
7. The RCM Controller also sends the UP's Endpoint and State to its Redundancy Manager service.
8. The RCM Configuration Manager pushes the Active or Standby configuration to the UP.
9. The UP stores the "Config Pushed" flag in its SCT (vpnmgr) subsystem.
10. The UP sends a "Config Push Complete" notification to the RCM Controller through vpnmgr.
11. The RCM Controller marks PendActive or PendStandby as full Active or Standby. (The Standby state is now available for switchover).
12. The RCM Controller sends the "Config Push Complete" message its Configuration Manager service for reconfirmation.

UP Switchover Sequence

The following is the UP switchover sequence when the RCM is used:

1. The RCM Controller detects the failure of the UP.
2. The RCM Controller changes the mapping of UP's Endpoint to Host ID, and assigns the failed active UP's Host ID to a new active UP that was selected from the Standby list.
3. The RCM Controller notifies its Configuration Manager service about switchover notification.
4. The RCM Configuration Manager activates the host-config of old active (by negating others), and changes the mapping of Host ID to new active UP.
5. The RCM Controller pushes the IP pool checkpoints to the new Active UP.

6. The RCM Controller notifies the Redundancy Manager service to push all the checkpoint data.
7. The RCM Controller sends State (Active) and Host ID to new Active UP.
8. The new Active UP receives the negate of config and control group association.
9. The new Active UP notifies the RCM Controller about the “Config Push Complete”.
10. The RCM Controller receives the “Config Pushed” flag from new Active and updates its data with State, Host ID, and pushed flag.
11. The RCM Controller notifies the Configuration Manager service about the “Config Push Complete” for validation.

RCM Controller Restart

Upon RCM Controller restart, the UP will detect connection failure and they will reattempt. Once RCM Controller is back, the UP will reconnect. The UPs that are in new state may have to wait for approximately five minutes before successful registration.

VPNMgr Restart

Upon restart of the UP’s VPNMgr, the VPNMgr reconnects with RCM Controller and ensures that both are in sync.

Switchover and Recovery

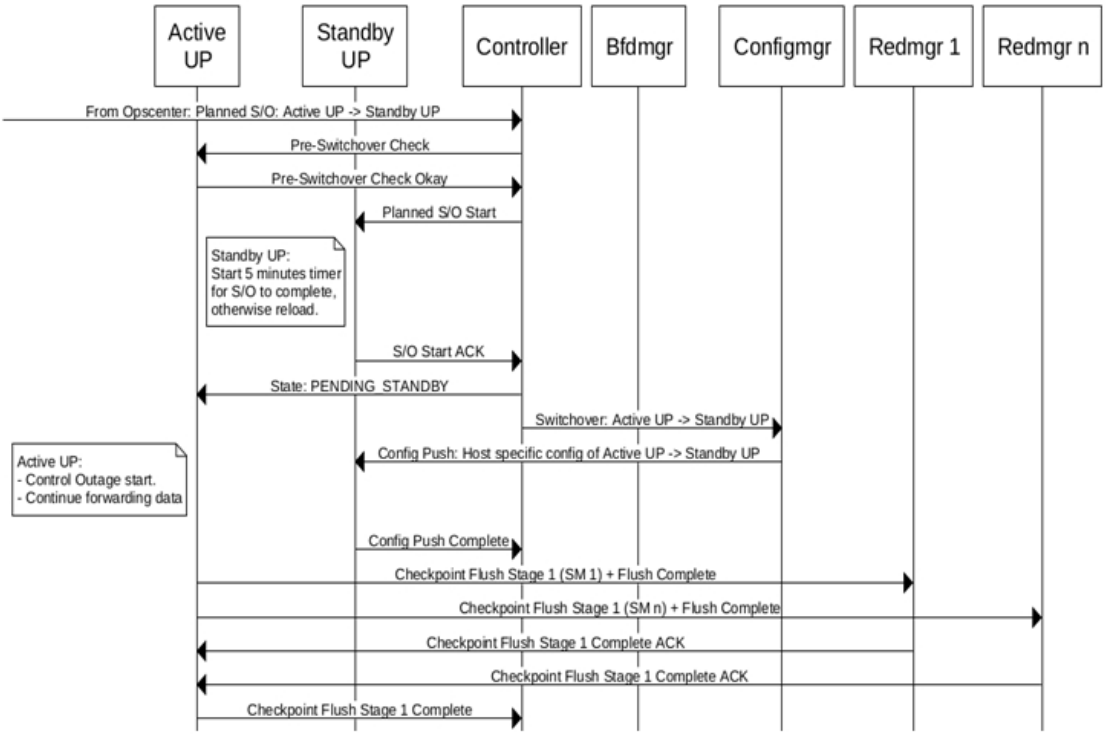
Planned Switchover

Planned switchovers can be done in the following ways:

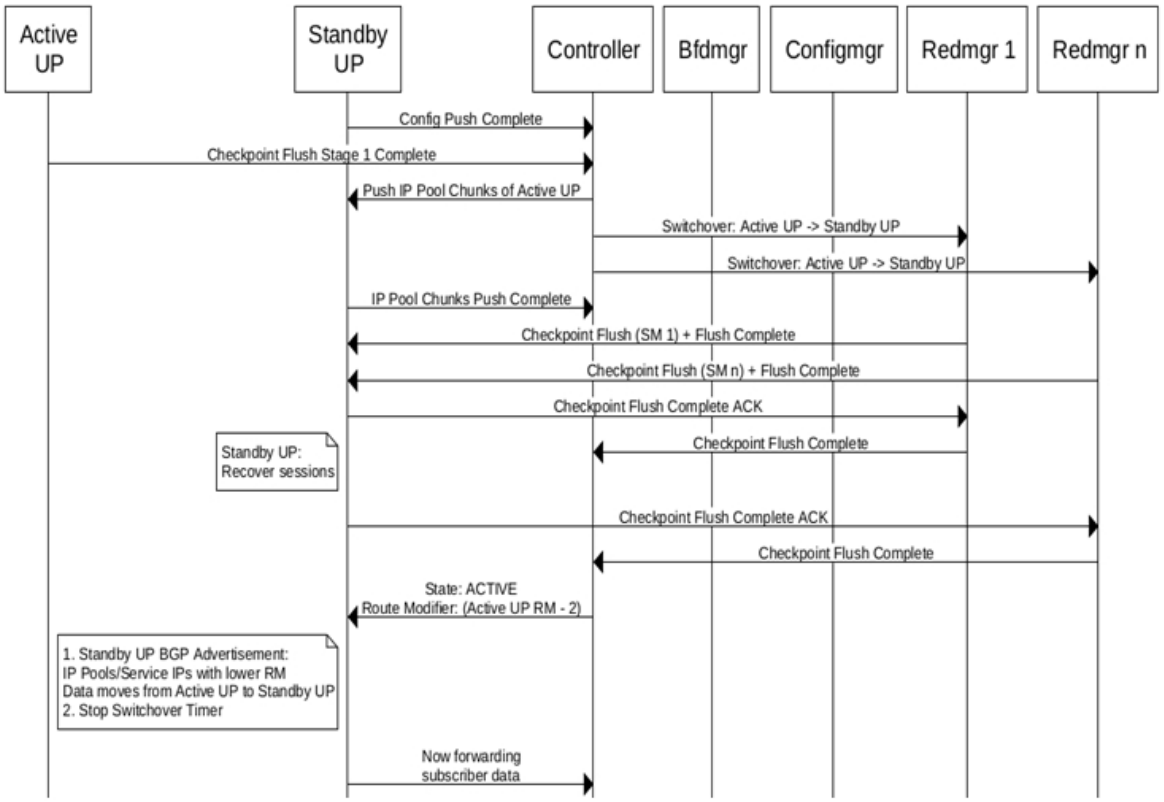
- When a UP is manually rebooted/reloaded during a maintenance window wherein the UP indicates to the RCM about the event of going down.
- When switchover is triggered from the RCM by either the Controller, Configmgr, or BFDMgr using the CLI. For information about the CLI command, refer the Performing Planned Switchover section.

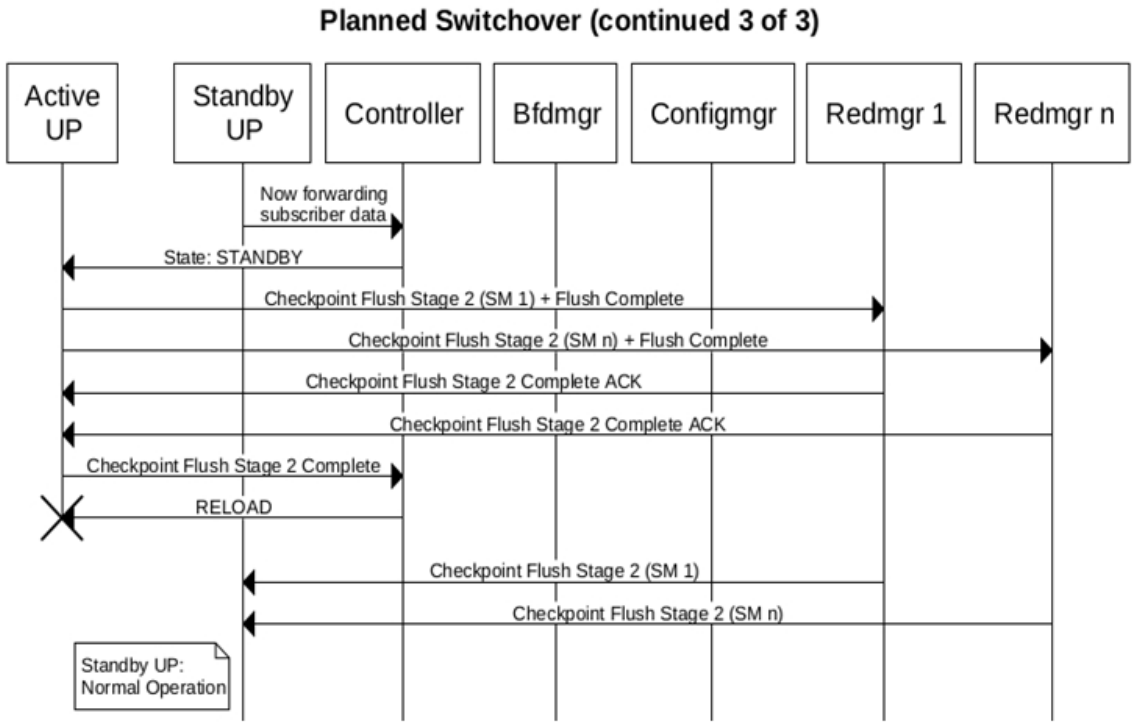
The following images illustrate the call flows for planned switchover.

Planned Switchover



Planned Switchover (continued 2 of 3)

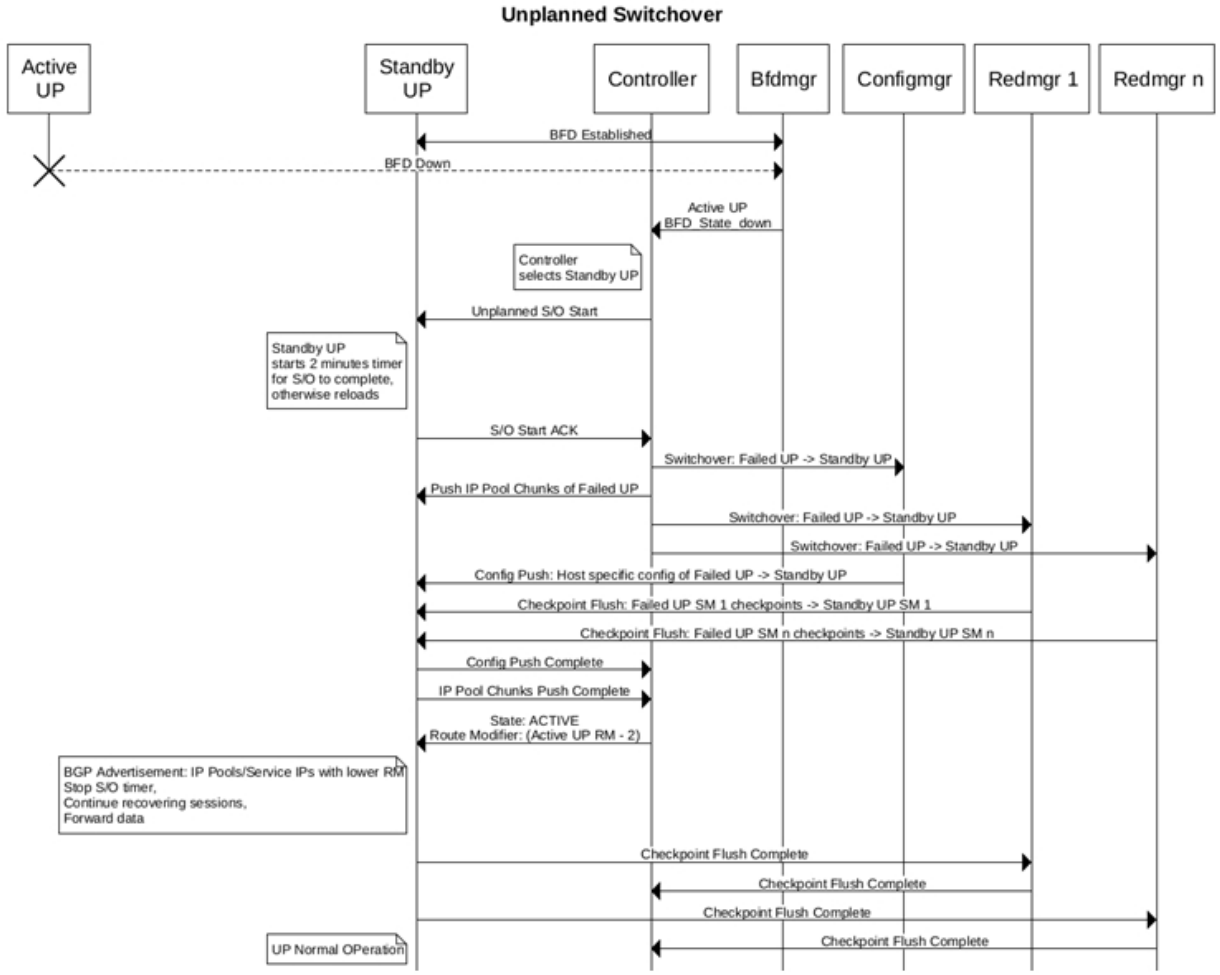




Unplanned Switchover

Unplanned switchovers occur due to issues in the UP and it gets rebooted without manual intervention. When unplanned switchover happens, the BFD monitor Pod detects that the UP has gone down and triggers the RCM Controller to start switchover mechanism. The RCM controller chooses a Standby UP and the Redundancy Manager Pods to start pushing configuration and checkpoints to the new UP.

The following image illustrates the call flow for an unplanned switchover.



Failover Time Optimization for Unplanned Switchover

The minimum time taken by the new UP to become Active and process traffic is determined by the following factors:

- Failure detection of the Active UP
- Configuration/IP Pool Flush from RCM
- Checkpoints Flush from RCM
- Active State transition and Route Convergence

As part of this functionality, a new CLI command (**`switchover allow-checkpoint-processing-active`**) is introduced that allows Active state to be pushed immediately after the “Config Push Complete”, and checkpoints are processed even after the chassis moves into Active state. This results in reducing the overall Failover time.

NOTE:

- If the “Config Pushed” time is more than the “Checkpoint Flush” time, this optimization will not yield the expected Failover time reduction.
- Failure Detection time optimization is currently not supported.

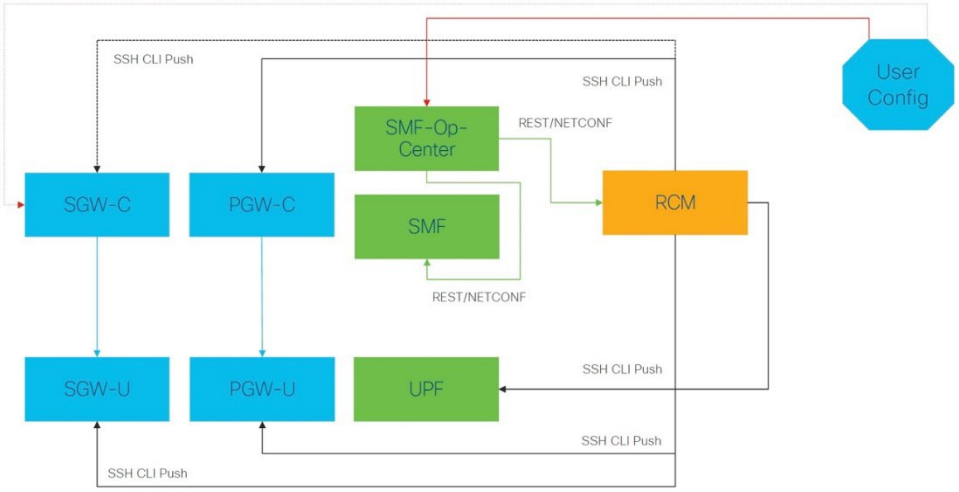
For information about the CLI command, refer the *Configuring Failover Time Optimization for Unplanned Switchover* section.

Configuration Management

Configuration Manager (ConfigMgr) module is required irrespective of whether redundancy is enabled or not. It is responsible to load the requested configuration for UPF. The functionality involves:

- On successful registration of UP with the RCM, the Controller provides the Management IP of the UP. For details, see UP Registration Call Flow.
- ConfigMgr receives the information of all Management IPs from the Controller.
- ConfigMgr pushes the UP Day-1 configuration to UP through SSH connection.
- Configmgr communicates with Controller over the gRPC link and exchanges the information.

The following figure depicts how the configuration works when RCM is used.



The ConfigMgr Pod is used to take the user input (configuration) from the Ops Center CLI and convert it into the corresponding StarOS CLI. The converted CLI is then pushed to the registered UP/UPF over SSH.

NOTE: The ConfigMgr Pod comes up only after the required CLI commands are configured in the Ops Center. Similarly, the UP/UPF node must have the appropriate CLI commands configured for it to connect to the ConfigMgr Pod. In the absence of the CLI commands, the UP/UPF node will not attempt to connect to the ConfigMgr for registration.

The following is the sequence of distributing UP/UPF-specific configuration to all the registered UP/UPFs.

1. The required configuration is applied in the Ops Center to bring up the ConfigMgr Pod. For details, see the Configuring the ConfigMgr Pod section.
2. The policy-charging related configuration is applied to generate the respective configuration maps. This is required for the ConfigMgr Pod to come up.
3. The UP/UPF node is configured with the required CLI commands to start the registration process with the ConfigMgr. For details, see Configuring UP with Day-0 Configuration section.

4. On successful registration, the UP/UPF receives the configured CLIs (configured as part of Step 2) over SSH.

Performing Planned Switchover

Use the following CLI command in Exec mode to perform planned switchover from Active to Standby User Plane.

- **rcm switchover source** *source_ip_address* **destination** *dest_ip_address*

- **NOTES:**
- The *source_ip_address* and the *dest_ip_address* are the RCM interfaces that you bind under redundancy-configuration-module mode.

Deferring SSH IP Installation

After UPF is up, the Day-0.5 configuration is executed on UPF. When UPFs register with RCM, the Controller pushes the hostID and SSH IP to UPF along with the state. The SSH IP received may get configured and saved as a part of Day-0.5 configuration. To avoid that, we must defer the SSH IP installation until the Day-0.5 configuration is saved.

The Deferring SSH IP Installation functionality is CLI-controlled. Before applying Day-0.5 configuration, you must execute the following Exec CLI command before entering the configuration mode. After the Day-0.5 configuration is done, you must first save the boot configuration (combined Day-0 and Day-0.5 configuration), and then set the CLI to default value.

[no] rcm ssh ip install

NOTES:

- When **rcm ssh ip install** CLI command is executed, the VPNMgr in UPF activates the SSH IP received from the Controller.
- When **no rcm ssh ip install** CLI command is executed, the VPNMgr defers the SSH IP installation until it's enabled again.
- By default, the CLI command is set to enabled state (**rcm ssh ip install**). After the VPNMgr receives the SSH IP from RCM Controller and finds that the SSH IP installation is disabled, it triggers the timer to defer the SSH IP configuration and rechecks the flag after every one second. Whenever the timer times out, it checks for the disable flag and if the flag is set in false state, only then it proceeds with SSH IP activation. If the disable flag is still true, the timer is restarted.
- You can verify the status of SSH IP installation by using **show rcm info** CLI command.

Important NOTE: This CLI is intended only for deployment scenario and must be kept to its default value of **rcm ssh ip install** during normal operation. Outside of deployment window, the behavior of **no rcm ssh ip** install is undefined.

Preventing Dual Active Error Scenarios

Use the following CLI configuration in CP to prevent dual Active error scenarios for N:M redundancy.

```
configure
   user-plane-group group_name
      sx-reassociation disabled
      end
```

NOTE:

- **sx-reassociation disabled**: Disables UP Sx reassociation when the association already exists with the CP.

Preventing Successive Switchover Due to Sx Monitor Failure

Use the following CLI configuration in RCM to prevent successive switchover due to Sx Monitor Failure.

```
configure
   switchover disallow-swo-timeout timeout_seconds
   end
```

NOTES:

- **switchover disallow-swo-timeout**: Specifies the timeout value to wait after receiving an Sx Monitor Failure. *timeout_seconds* is the interval in seconds.

Configuring the RCM

This section describes how to configure the RCM.

Configuring the Controller Pod

Use the following CLI configuration to configure the endpoint to bring up the RCM Controller pod.

```
configure
   endpoint rcm-controller
     vip-ip ip_address
     exit
```

NOTES:

- **vip-ip** *ip_address*: Specifies the host IP address.
- This command is disabled by default.

Configuring the BFD Manager Pod

Use the following CLI configuration to configure the endpoint to bring up the RCM BFD Manager pod.

```
configure
   endpoint rcm-bfdmgr
   vip-ip ip_address
   exit
k8 smf profile rcm-bfd-ep bfd-monitor group group_id
k8 smf profile rcm-bfd-ep setvar bfd-src-ip ip_address
   endpoint ipv4 ip_address
   endpoint ipv4 ip_address
   endpoint ipv4 ip_address
   min-tx-int tx_microseconds
   min-rx-int rx_microseconds
   multiplier count
   standby count
   exit
```

NOTES:

- **k8 smf profile**: Specifies the Kubernetes (k8) SMF configuration with the external IP or port configuration.
- **rcm-bfd-ep**: Specifies the BFD endpoint parameters.

- **bfd-monitor**: Specifies the BFD application configuration.
- **group** *group_id*: Specifies the group ID.
- **bfd-src-ip:** Specifies the source IP of BFD packets.
- **standby** *count*: Specifies the number of required Standby UP/UPFs.
- **vip-ip** *ip_address:* Specifies the host IP address.
- This command is disabled by default.

Configuring the ConfigMgr Pod

Use the following CLI configuration to configure the endpoint to bring up the RCM Configuration Manager pod.

```
configure
   endpoint rcm-configmgr
   exit
```

NOTES:

- The UP credentials are passed to Configuration Manager as K8s secret. You must create these credentials before you start the RCM cluster. For details, see Configuring UPF Credentials section.

Configuring UPF Credentials

Use the following CLI commands to configure the credentials for UPF.

```
configure
   k8 smf profile rcm-config-ep upfloginmode { upfcredentialsbased |
upfsshkeysbased }
   exit
```

NOTES:

- **upfloginmode**: Specifies the mode of UP/UPF login.
- **upfcredentialsbased**: Specifies the default or unique credentials per UP/UPF for username/password. This is the default option.
- **upfsshkeysbased**: Specifies the default or unique credentials per UP/UPF for username/SSH-Key.

Sample Configuration for Credential-based UPF Login

```
k8 smf profile rcm-config-ep upfloginmode upfcredentialsbased
k8 smf profile rcm-config-ep rcm-upfinfo group group_name
upf-default-cred username username
upf-default-cred password password
host-id active active_name1 management-ip ip_address
host-id active active_name2 management-ip ip_address
```

```
...
...
...
host-id standby standby_name1 management-ip ip_address
...
...
...
upf mgmt-ip up1_mgmt_ip upfcredentials username username password password
upf mgmt-ip up2_mgmt_ip upfcredentials username username password password
...
...
...
upf mgmt-ip up_n_and_m_mgmt_ip upfcredentials username username password
password
```

NOTE:

- **upf mgmt-ip** *up_n_and_m_mgmt_ip* **upfcredentials username** *username* **password** *password*: These credentials are required only for per-UP unique-credential scenario.
- *active_name1*, *active_name2*, and so on, must be same as host names under host configuration.
- **management-ip** *ip_address* and **mgmt-ip** *up_n_and_m_mgmt_ip* are the IP address of UP/UPF over which configuration push happens.
- *standby_name1* can be any name.

Sample Configuration for SSH Key-based UPF Login

```
k8 smf profile rcm-config-ep upfloginmode upfsshkeysbased
k8 smf profile rcm-config-ep rcm-upfinfo group group_name
upf-default-sshkeys username username
upf-default-sshkeys private-key private_key
upf mgmt-ip up1_mgmt_ip upfsshinfo username username private-key private_key
upf mgmt-ip up2_mgmt_ip upfsshinfo username username private-key private_key
...
...
...
```

upf mgmt-ip *up_n_and_m_mgmt_ip* **upfsshinfo username** *username* **private-key** *private_key*

NOTE:

- **upf mgmt-ip** *up_n_and_m_mgmt_ip* **upfsshinfo username** *username* **private-key** *private_key*: These credentials are required only for per-UPF unique-credential scenario.
- After **private-key** keyword, press "Enter" for Multiline mode and then enter the private key.
- You must pass the Management IP of the UPFs for configuration push.

Restricting Configuration Maps Pushed to Configuration Manager

Use the following command to restrict the configuration maps that are pushed to the Configuration Manager by default.

```
configure

   k8 smf profile rcm-config-ep disable-cm { apn | apnprofile |
chargingAction | common | creditCtrl | global | gtpp | gtpuService |
lawfulIntercept | misacs | packetFilter | rulebase | ruledef | sxService |
upSvcs | upfCpg | upfIfc | urrList }

   exit
```

- **NOTES**:
- **{ chargingAction | creditCtrl | global | gtpp | gtpuService | misacs | packetFilter | rulebase | ruledef | sxService | upSvcs | upfCpg | urrList }**: Specifies the Kubernetes (k8) Session Manager function (SMF) configuration with the external IP or port configuration.
- **apn**: Restricts the Access Point Name (APN) map.
- **apnprofile**: Restricts the APN Profile map.
- **chargingAction**: Restricts the charging action map.
- **common**: Restricts the common map.
- **creditCtrl**: Restricts the credit control map.
- **global**: Restricts the content filtering and URI blacklisting map.
- **gtpp**: Restricts the GTPP map.
- **lawfulIntercept**: Restricts the context LI map.
- **gtpuService**: Restricts the GTPU service map.
- **misacs**: Restricts the miscellaneous active charging services (ACS) map.
- **packetFilter**: Restricts the packet filter map.
- **rulebase**: Restricts the rulebase map.
- **ruledef**: Restricts the ruledef map.
- **sxService**: Restricts the Sx service map.
- **upSvcs**: Restricts the User Plane (UP) services map.

- **upfCpg**: Restricts the UP function (UP/UPF) Control Plane (CP) group map.
- **upfIfc**: Restricts the UP function (UP/UPF) Interface map.
- **urrList**: Restricts the usage reporting rule (URR) map.

Configuring the Redundancy Manager Pod

Use the following CLI configuration to configure the endpoint to bring up the RCM Redundancy (Checkpoint) Manager pod.

```
configure
    endpoint rcm-chkptmgr
    replicas count
    vip-ip ip_address
    exit
```

NOTES:

- **replicas** *count*: Specifies the number of replicas per node.
- **vip-ip** *ip_address*: Specifies the host IP address.
- This command is disabled by default.

Configuring UP with Day-0 Configuration

This section describes the CLI commands that are required to configure UP with Day-0 configuration. It involves:

- Registering the UP to RCM
- Multi-hop BFD
- Configuring BGP Monitoring
- Configuring BFD Monitoring
- Configuring UP Auto-reboot
- Configuring Failover Time Reduction in RCM

NOTES:

- All UPs have Day-0 configuration with RCM context.
- The physical interfaces and port configurations are part of the Day-0 configuration.

A new context **rcm** is created in UP. The VPNMgr in UP is used to communicate with the RCM.

If RCM Redundancy is configured, use the following command to configure UP.

```
configure
```

```
   context context_name

      bfd-protocol

         bfd multihop-peer bfd_manager_ip_address interval tx_interval min_rx
rx_interval multiplier number_of_rx_timeouts

         exit

      redundancy-configuration-module rcm_name

         bind address local_bind_ip_address

         rcm controller-endpoint dest-ip-addr control_ep port port_number

         up-mgmt-ip-addr up_mgmt_addr node-name node_name

         monitor bfd peer bfd_peer_ip_address

         monitor bgp context context_name peer-ip [ group group_number ]

         [ default | no ] rcm-unreachable triggers-reload delay
reload_interval_seconds

         [ default | no ] switchover allow-early-active-transition store-
checkpoint { enable | disable }

         exit

   ip route static multihop bfd bfd_name local_bind_ip_address
bfd_peer_ip_address

   switchover allow-checkpoint-processing-active { true | false }

   end
```

NOTES:

- **redundancy-configuration-module**: Configures the Redundancy Configuration Module (RCM) and enters the RCM mode.
- **rcm**: Specifies the CUPS-related options in this context.
- **controller-endpoint**: Enables the registration of the UP at the Controller Manager endpoint.
- **dest-ip-addr** *control_ep*: Specifies the destination IP address. *control_ep* is the IP address of the Controller endpoint.
- **port** *port_number*: Specifies the TCP port value. *port_number* must be 9200.
- **up-mgmt-ip-addr** *up_mgmt_addr*: Specifies the management IP of the UP, which the Configuration Manager uses to securely connect. *up_mgmt_addr* is the IP address of the UP.
- **node-name** *node_name*: Specifies the node name of the UP to send the Configuration Manager endpoint. *node_name* is the UP node name.

- **peer** *bfd_peer_ip_address*: The *bfd_peer_ip_address* must already be configured as a multihop BFD peer session with RCM BFD Manager pod in the context *context_name*. *bfd_peer_ip_address* IP address can be different from the RCM Controller Pod IP address.
- **group** *group_number*: [Optional] Configure to track multiple BGP peers under same monitor group. *group_number* must be in the range of 1 through 10.

  If group is not specified, monitor will act as an individual monitor.

  If previously configured, use **no monitor bgp context** *context_name peer-ip* CLI command to remove the BGP monitor in RCM.
- UP/UPF informs RCM to trigger UP/UPF switchover under following scenarios:
  - When a monitor, configured without group number, fails.
  - When all the monitors, configured under single group number, fails.
- **peer** *bfd_ip_address*: The *bfd_ip_address* should already be configured as a multi-hop BFD peer session with RCM Bfdmgr pod in **context** *context_name*. This IP address can be different than the RCM Controller Pod IP address.

  If previously configured, use **no monitor bfd peer** *bfd_ip_address* CLI command to remove the BFD monitor in RCM.
- Configuring BFD monitor in RCM provide the following functionality:
  - Switchover Scenario 1: The UP/UPF tears down TCP connection with RCM controller if BFD monitor goes down. It waits for both BFD monitor with RCM BFDMgr and TCP connection with RCM Controller to come up per configured CLI.
  - Switchover Scenario 2: If UP/UPF is not in Init state and receives Init state from Controller, the UP/UPF reboots itself.

  At initial boot time, UP/UPF does not initiate TCP connection with RCM Controller if BFD monitor is down.
- **switchover allow-early-active-transition**: Allows early transition to active during switchover. Traffic is processed along with the pre-allocation of other calls.
- **store-checkpoint { enable | disable }**: Enable or disable storing of checkpoints until the end of checkpoint flushing from RCM.
- **enable**: Enable storing of checkpoints and defer call pre-allocation until end of flushing.
- **disable**: Disable storing of checkpoints and allow call pre-allocation during flushing.
- **bfd multihop-peer**: Specifies the time interval within which RCM must detect failure by using a protocol such as multi-hop BFD.
- *reload_interval_seconds*: Specifies the time interval, zero (0) seconds to 30 days (in seconds), for chassis reload after BFD with RCM goes down. Default is no reload.
- **switchover allow-checkpoint-processing-active { true | false }**:
  - When set to **true**, RCM does not wait for the completion of checkpoint flush before sending Active state and lower Route Modifier to new Active UP.
  - When set to **false**, RCM waits for the completion of checkpoint flush before sending Active state and lower Route Modifier to new Active UP.
  - By default, the CLI command is set to **false**.

Configuring Failover Time Optimization for Unplanned Switchover

Use the following configuration to allow checkpoint processing in Active state, for a brief period, during unplanned switchover.

```
configure
   context context_name
      redundancy-configuration-module rcm_name
         [ no ] switchover allow-checkpoint-processing-active
         end
```

NOTES:

- By default, the configuration is disabled.
- When this CLI command is configured under RCM context in UP, it accepts and processes checkpoints in Active state. This is done only for the Unplanned switchover initiated by the RCM. The checkpoints are accepted only till the checkpoint flush is completed from RCM. Any checkpoint that comes after flush is complete, is dropped.
- Use the **show config context** *context_name* CLI command to verify if **switchover allow-checkpoint-processing-active** CLI command is enabled.

For sample configurations, refer Appendix B: Sample Common and Host-specific Configurations

Configuring the RCM through Network Services Orchestrator

Feature Description

Redundancy of User-planes is achieved through RCM. Currently, the User-planes are configured from RCM and the redundancy module is also managed by RCM. RCM currently uses a String-based Config Approach to configure the UPs. However, this approach results in several issues such as configuration validation, failure handling such as rollback, and most importantly configuration automation by orchestrators such as Network Services Orchestrator (NSO). This feature addresses all these issues by moving the configuration out of RCM and let the NSO (or any third part Orchestrator) deal with configuration. With this feature, NSO pushes all day-1 and day-N configurations to UP and RCM. RCM manages Redundancy functionality by pushing only the config required during UP switchover.

Architecture

With this feature support, we let the NSO configure each of the UPs with common and all host specific configurations. RCM gets the config from NSO. Once UPs are brought up by the NSO CFP, UPs first register with RCM and get the SSH-IP. Parallelly RCM also notifies NSO on UP Registration. NSO then pushes the entire config to UPs via the SSH-IP provided by RCM. During switchover, RCM is responsible for Host configuration negation. Standby UP then applies the received config and becomes Active UP.

This new method of configuring RCM and UP is CLI controlled. The CLI option is provided at the RCM OPS-CENTER.

Components

NSO

Network Services Orchestrator (NSO) is an added component to the RCM Architecture that functions as the configurator for both UP and RCM.

With this feature getting introduced, there are changes in the RCM component's functionality.

RCM changes its role to only managing redundancy modules rather than both redundancy and configuration modules.

RCM Configmger: RCM Configmgr varies in its functionality. Configmgr is responsible for pushing switchover time host specific configuration only.

How it Works

On Ops-center, a CLI is provided to configure the mode of operation and to determine if the config push is done by NSO or RCM.

If it is configured by RCM, the existing string-based mechanism is supported, and this is used mostly in the case of RCM as a VM where config scripts are used to configure the RCM for UP config.

If it is configured by NSO mode, all UPs are configured by NSO. Additionally, NSO pushes the host specific config to RCM as well. For this RCM also provides the Netconf method of configuring host specific UPF config. RCM has yang models for accepting the host-specific config of UPFs. This Yang is slightly different when

compared to StarOS yang for configuring the same CLI. RCM needs to know the redundancy group information for each of the hosts. So, the yang model is enveloped inside the redundancy group container.

RCM Ops-center also has Yang models developed to configure the SSH Ips for all the UPs in a redundancy group along with the hostIDs. It contains a provision to configure username and password for the management IP of each UP in the redundancy group.

Configuring UP and RCM

This section explains the solution in detail and explains how requirements are met and challenges involved in meeting them. The design is involved in 3 stages as stated below:

1. Initial Deployment
2. UPF Switchover
3. Day-N config updates

The following sections discuss these stages in detail.

If RCM sends the following notifications using the “alert-notification” stream, any Netconf client should create subscription to this stream on rcm-ops-center for the changes.

1. SSH IP notification on registration during Day-0,1 deployment containing ssh-ip, management IP ... This notification to CFP.
2. Recovered notification after switchover about the successful switchover and new active containing info of SSH-IP, management IP and State to CX.
3. Config Push request notification containing SSH-IP, management IP and state to CX.

Config definitions on UPF and RCM:

Day-0 Config:

- On UPF, the day-0 config consists of local context and management IP to reach it.
- On RCM, the day-0 config consists of configs to instantiate the RCM-ops-center and the images of the rcm-products.

Day-0.5 Config: day-0.5 config of each UPF consists of RCM context that can trigger UPF registration with RCM.

Day-1 Config:

Day-1 RCM Config:

The day-1 config of RCM includes the endpoints configuration:

- Endpoints [rcm-configmgr, rcm-checkpointmgr, rcm-controller, rcm-bfdmgr, rcm-keepalived, rcm-snmp-trapper, rcm-strongswn]
- Each individual pod VIP Ips
- Mandatory configuration for each of the endpoints
- UPF endpoints that would be connected to this RCM
- Number of standbys

- Credentials for each UP or redundancy group as a whole
- Config mode of operation. i.e., configuration through RCM or NSO.

Day-1 UPF Config:

The day-1 Config of UPF includes the following:

- ECS Configuration of UPF
- Host Specific Configuration of UPF

Host Specific Configuration of UPF:

- Host specific configuration includes (based on number of services needed on UP):
  - User plane service config
  - Interfaces config
  - Sx service config
  - GTPU service config
  - LI config
  - Traffic steering config
  - IPSEC config
  - NAT and IP pool config
  - Control plane group config

Along with SSH IP of each UP and host IDs, this host specific configuration is configured as two variants.

- StarOS Clis on NSO
- RCM variant of host specific configuration that includes the redundancy group details

Day-N Config:

- The day-N config of UP includes any updates to ECS config and host specific configuration of UP. This is configured on NSO and then NSO pushes the day-N host specific config to RCM to keep the configuration for negation during switchover up to date.
- This is mostly change in some logging levels, change in some timers that can be done straightforward by NSO on RCM.

Initial Deployment

Use the following sequence of operations that occur during initial deployment. CNDP cluster manager or ESC deploys both RCM and UPF with Day-0 configs:

1. CX NSO wrapper invokes CFP package of NSO to configure and deploy the RCM.
2. The NSO CFP package configures CNDP CM for RCM and deploy the RCM image.
3. CNDP-CM/ESC creates RCM with Day-0 config (standby is also created if RCM HA is intended).
4. CFP polls CNDP CM for RCM readiness. Once it indicates that RCM ops-center is up and running, NSO CX configures the RCM.
5. NSO CX configures the RCM day-1 config and also configure UPF host specific config.

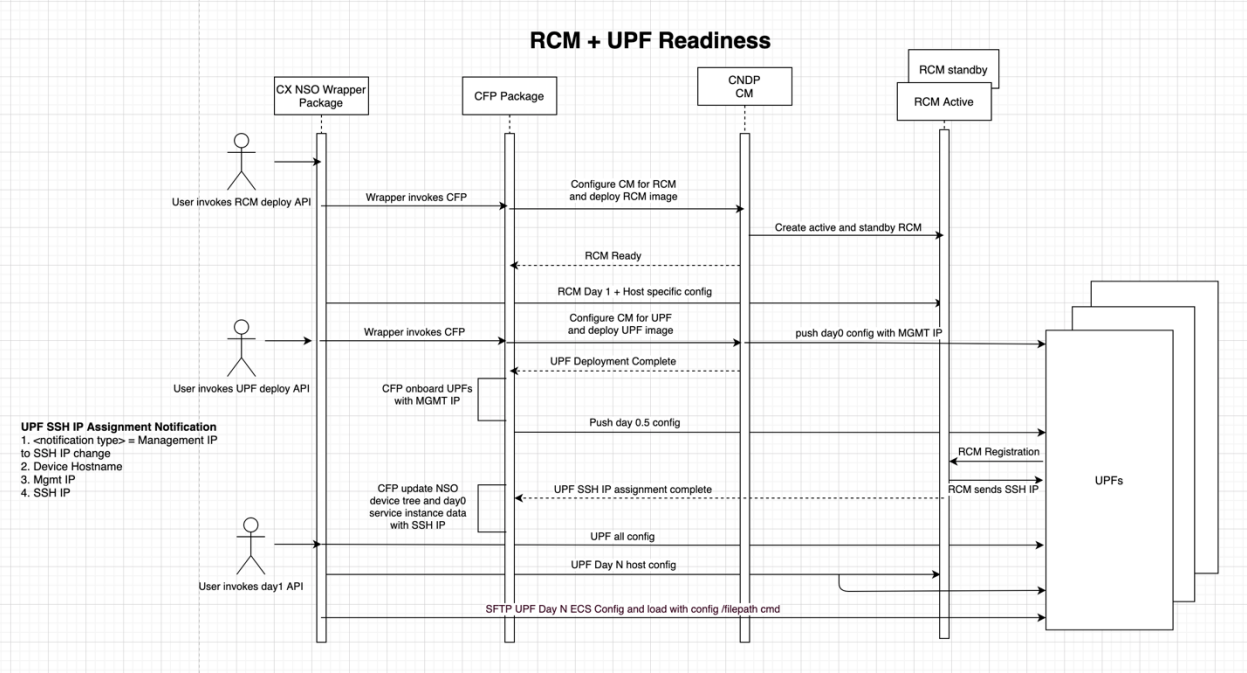
6. CX NSO wrapper invokes CFP package of NSO to configure and deploy the UPF.
7. The NSO CFP package configures CNDP CM for UPF and deploy the UPF image.
8. CNDP-CM/ESC creates UPF with Day-0 config and pushes the day 0.5 along to it.
9. Once UPF is up the CNDP CM intimates to CFP about the UPF deployment completion.
10. The CFP onboards UPF and creates a map with UPF mgmt IP.
11. Once the day-0.5 config is done on UPF, UPF registers with RCM and RCM evaluates the available redundancy options and assigns a particular state to the UPF.
12. RCM controller reads the upf endpoint details and ensures that UP is part of configured UPF endpoints and assigns either active or standby to the UPF. Controller pushes the hostID and SSH IP to UPF along with the state.
    - If "`k8 smf profile rcm-config-ep switchover mw-switchover-pause true`" is configured, the hostID and SSH-IP is chosen based on the mapping with the management IP.
    - If this CLI is not configured, any one free hostID is chosen from the pool of hostIDs . This hostID along with its corresponding SSH-IP configured is pushed to UPF and vpnmgr in UPF will activate the SSH IP received from the Controller.
    - Controller to notify Configmgr about the UPF host details (state, optionally ssh-ip and so on)
13. Controller then prepares the message containing UPF state, SSH IP, management IP (if cli "k8 smf profile rcm-config-ep switchover mw-switchover-pause true" is configured) and trigger a notification to NSO.
14. NSO on receiving the notification prepares the config and pushes to UP as follows. NSO also frames the SSH-IP to services mapping (with the help of management IP received) if not already done.
    - If the notification is about active UPF, NSO will push:
        - Host specific config of particular SSH-IP (3.1.1.3.1 a to i)
        - ECS config
    - If the notification is about standby UPF, NSO will push:
        - Host specific config (3.1.1.3.1 a to h only) of all active hosts
        - ECS config
        - Control-plane group config should not be configured on standby UPF and it is mandatory not to push this to standby UPF
    - Better to add sleep of about 60 secs between host specific and ECS config so that there is enough time for services to start before UPF consumes ECS config. We do have a CLI to wait for the user plane services to be started before we proceed. That can also be used instead. For Example: "rcm-config-push-complete timeout 60".
    - At the end of all StarOS config, a mandatory "`rcm-config-push-complete`" CLI needs to be pushed on all UPs during day-1 to mark the end of day-1 config push to UPF.

NOTE: As NSO receives UP is available notification, NSO restarts full configuration from the beginning aborting earlier Day-1/N retries. NSO has a config retry mechanism because UPs may go down even without s/o or because of network issues. Given this ability of NSO, if UP cannot configure SSH-IP within some timeout, it reloads itself. If needed (optimization) when the UP (BFD) goes down, the Controller can tell NSO that UP is down so that it can abort config retries. If UP takes time to configure SSH-IP (but configures it before timeout), NSO retry should be able to handle this situation.

15. UPF on receiving this "`rcm-config-push-complete`" marks the end of config push from external entity and notify the same to controller.

16. Controller then makes the corresponding UPF as active or standby in its data structures.
17. Once it is ensured that the desired number of Active/Standby UPFs are up and running, the cli "`k8 smf profile rcm-config-ep switchover mw-switchover-pause false`" should be flipped to "false".

The following diagram illustrates the RCM and UPF readiness.



UPF Redeployment

UPF redeployment needs Registration notification from RCM toward NSO. In production environment where DeploymentMode flag is set to FALSE, only Config-Push notification is sent from RCM toward NSO when UPF boots.

A new Exec mode CLI command is introduced that invokes RCM to send a Registration notification towards NSO when the specified UPF boots in deployment environment with DeploymentMode flag set to FALSE.

`rcm force-nso-registration management-ip` *`mgmt_ip`* **`true`**

After redeployment, you can execute the CLI with **`false`** option:

`rcm force-nso-registration management-ip` *`mgmt_ip`* **`false`**

MOP

1. Verify in RCM configuration: **`k8 smf profile rcm-config-ep switchover deployment false`** .
2. Set the CLI for the UPF (to be redeployed) to TRUE. Verify with **`rcm show-statistics controller`** CLI command.
3. Reload/Boot the UPF. RCM sends Registration notification toward NSO.

4. Let NSO complete configuration push and UPF status change from "Pending Active" to "Active" or from "Pending Standby" to "Standby" in the output of **`rcm show-statistics controller`** .
5. Set the CLI for the UPF to FALSE. Verify with **`rcm show-statistics controller`** . Now for subsequent reload of this UPF, RCM sends Config-Push notification towards NSO.

NOTE: Redeploy only one UPF at a time.

Day-N configuration updates

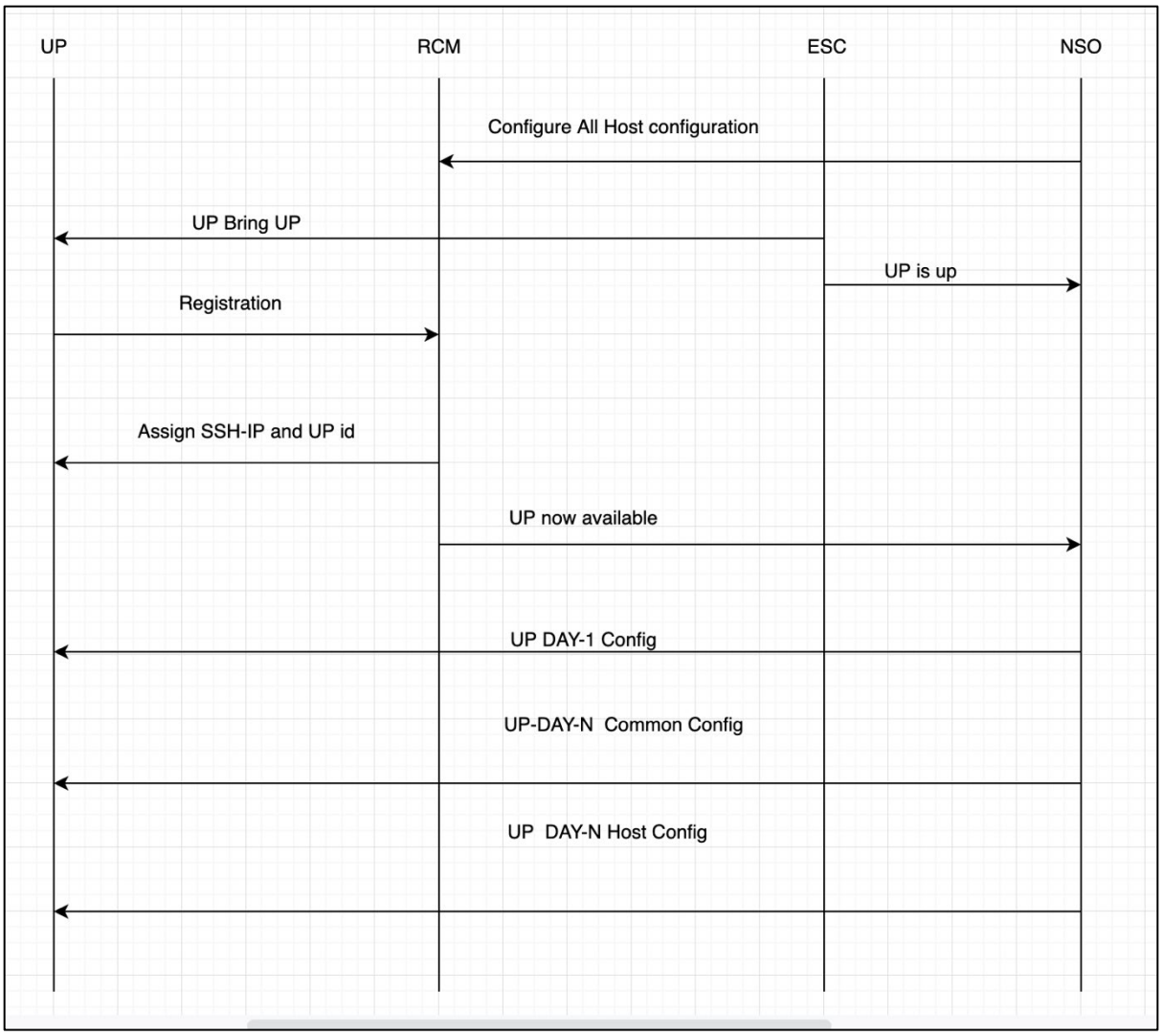
1. Day-N RCM configuration: This is mostly change in some logging levels, change in some timers that can be done straightforward by NSO on RCM.
2. Day-N ECS config update: NSO will update the config directly on all UPFs line by line similar to Day-1 config push.
3. Day-N host config update: If there is a specific host config update for a particular host, NSO can directly update the UPF through SSH IP provided by the RCM. In addition to that particular host, NSO needs to update the standby UPF (standby can be identified by the standby SSH-IP on NSO) and also the RCM by converting the host specific config into RCM config format.

Important: If the day-N host specific modification is about the control plane group, as described earlier, the standby config template does not contain this specific modification and only the active UPFs config template contains this specific service config. Additionally, RCM is also updated with the control-plane group config, since this control-plane group config must be pushed to new Active during Switchover.

While Day-N configuration is being configured by the User using NSO, in addition to pushing the updates to UPF and RCM as described above, NSO also updates the conf-D database within itself with the latest updated config and prepare a StarOS CLI config file that can be sftp'd to a newly rebooted UPF.

Upon switchover or reboot of UPFs and after notification from a controller, NSO retrieves the host specific config for that device and pushes the day-N StarOS configuration file, and uses configure <url> CLI on UPF to apply that config. NSO does not apply config line by line during the UPF reboot scenarios

The following image illustrates the call flow for Day-N configurations.



UPF Switchover

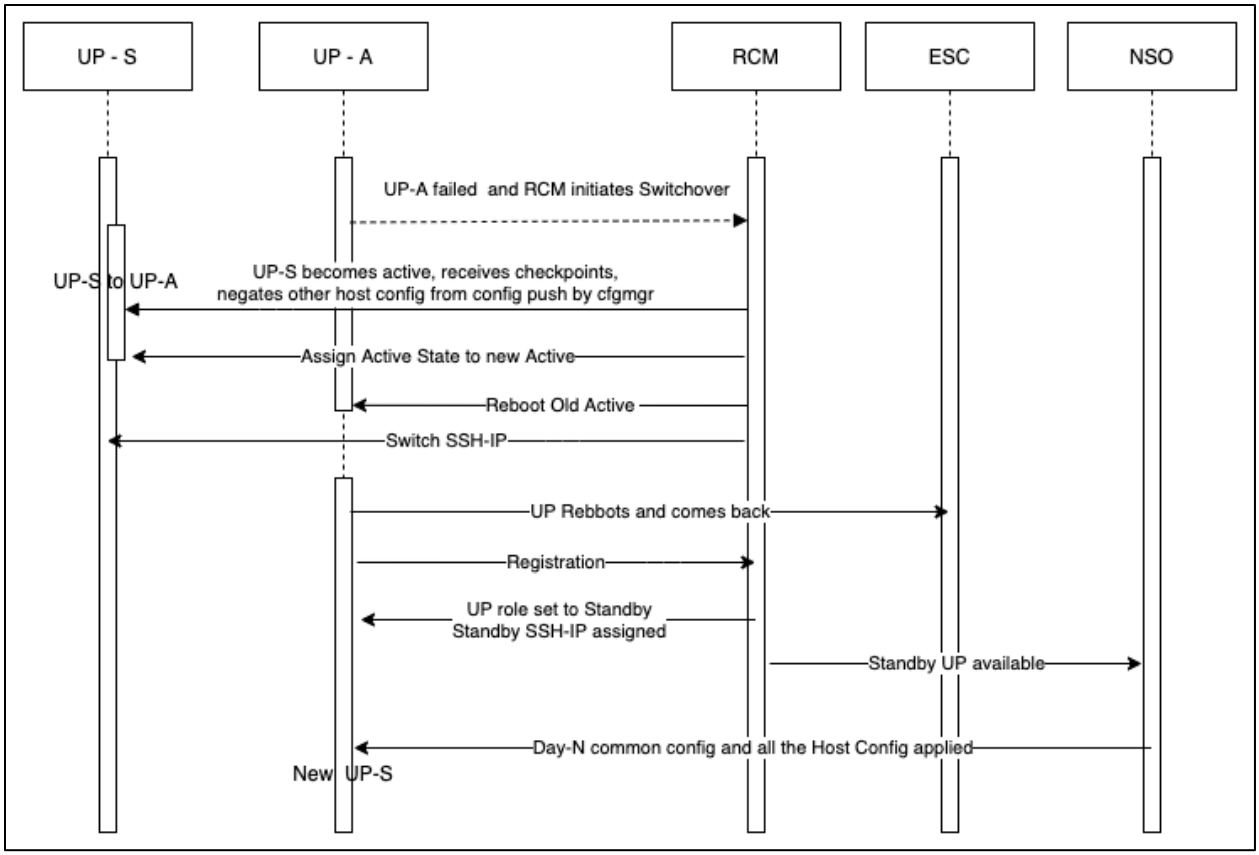
The following sequence of operations are used during UPF Switchover:

1. RCM-BFDMGR in RCM detects the UPF failure and informs controller about the same.
2. RCM-Controller initiates switchover operations through RCM-checkpointmgr and RCM-configmgr.
3. RCM-Configmgr pushes the negate config for that UPF to the new Active UPF and then UPF notifies the controller about the Config negation completion.
4. RCM-Checkpointmgrs pushes the checkpoints for that UPF to new Active UPF.
5. Notifies RCM-Controller on completion.
6. RCM-Controller flushes the IP-Pool data to the new Active UP.
7. Controller on receiving successful messages about config push, checkpoint flush, and IP pool flush updates the SSH IP of old active into new active and then revamps the UPF endpoint details accordingly.
8. When Old active properly reboots (in case of proper planned/unplanned switchover):

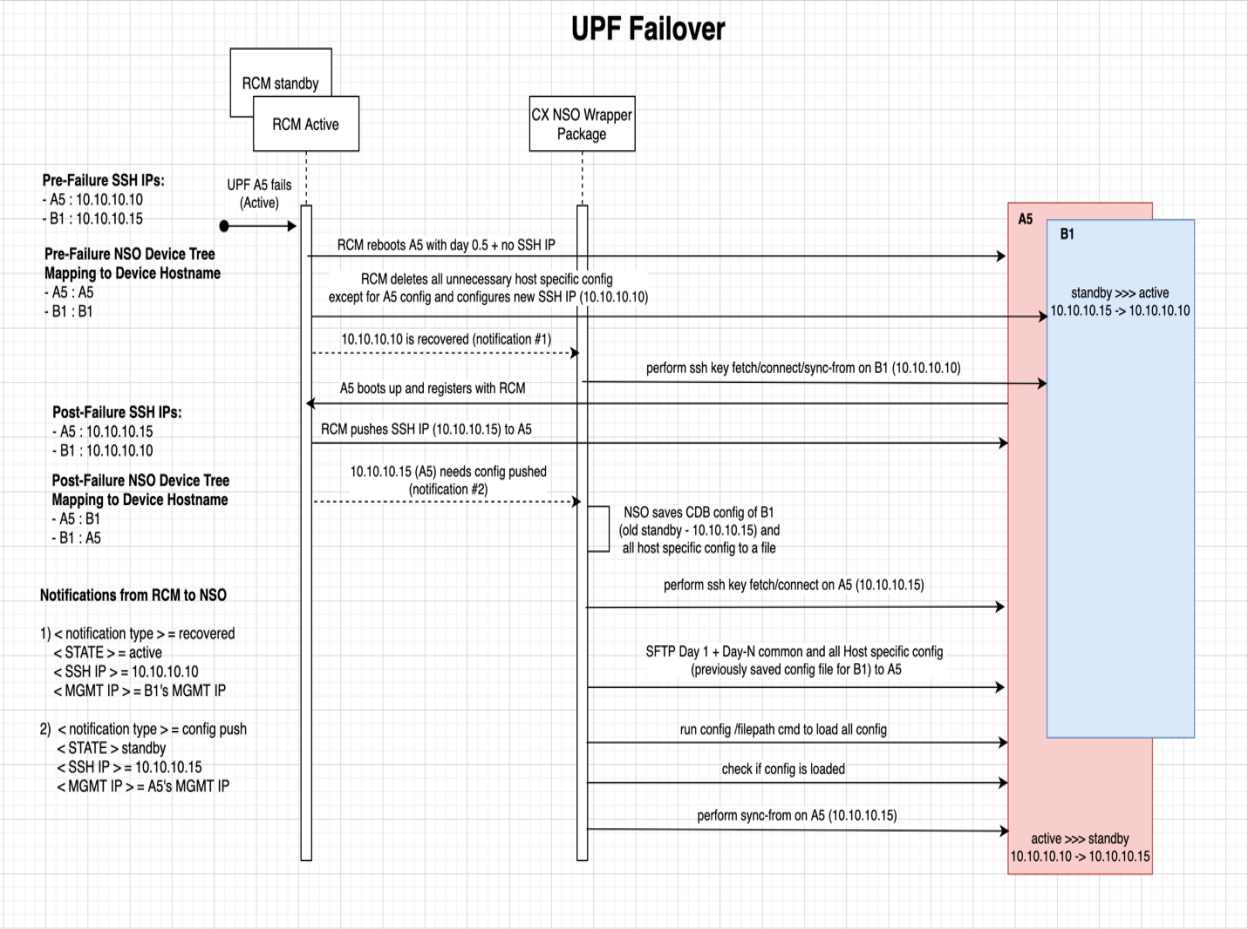
- When the old active is rebooted and comes back up, the Controller assigns to standby State and installs standby SSH-IP address into it.
- Controller then notifies NSO about the standby UP (along with SSH details) so that NSO can push the corresponding config into the standby UP.

9. When there is a switchover failure:
   - Case 1: Both old active and new active rebooted before s/o is complete, then any one of them becomes Active and the other one Standby. Active contains host id/SSHIP and Standby contains one of the free standby SSH-IP
   - Case 2: New Active reloaded before s/o is complete – Old Active State/SSHIP/HostID maintained (indefinitely), new Active is marked for Reload. If Old Active comes back as Init, then it is Case 1, if it comes back as it was before BFD down, then it is Case 2.
10. When the old active momentarily loses BFD connection and connects back before switchover is complete:
    - In this case, controller aborts the switchover and restores the SSH-IP as if switchover never occurred and reboots new Active. Old Active communicate its SSH-IP to controller. Controller then assigns to active state and assigns the active SSH IP to the old active again and informs the NSO accordingly. UP again configure the SSH-IP, which it had removed when BFD with Controller went down. New Active reboots as Standby and gets a free Standby SSH-IP (most probably what it had earlier).
11. When old active loses BFD connection and if connection restores after switchover is completed:
    - Controller then reboots the old active when it reconnects to RCM and upon reboot assigns the standby SSH connection.
12. In both case 10 and 11, when there is BFD disconnection, the UP if alive should deactivate the SSH IP so that it can properly isolate itself and not to interfere with config updates that might be occurring for new active. It reinstates the earlier SSH-IP only when controller restores its earlier state upon reconnection. Else, it would be rebooted by controller.

The following image illustrates the call flow for Sequence of operations during switchover.



In addition to the above sequence, the below sequence diagram explains the mapping update in the CX NSO package and the message communication format between RCM and NSO.

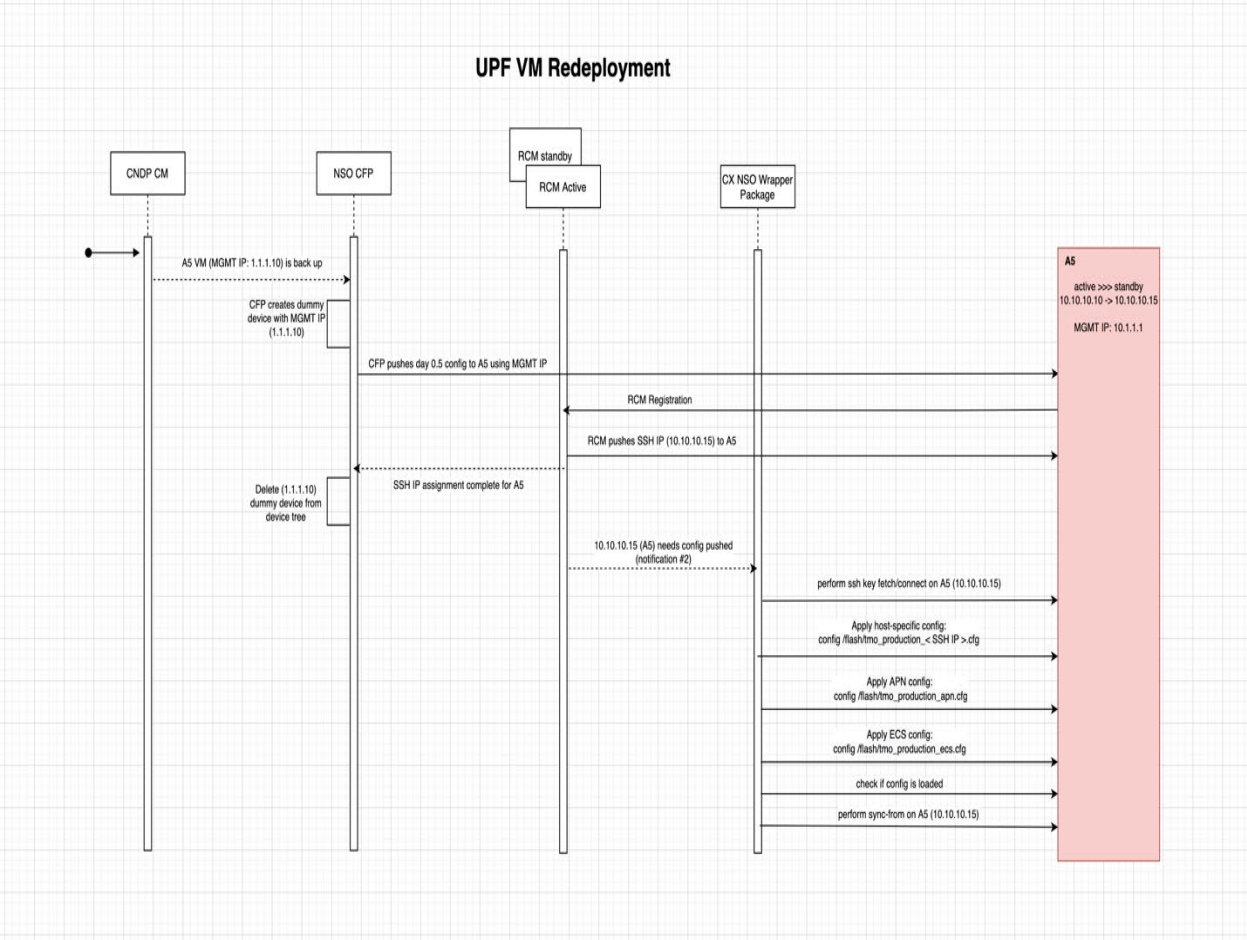


Switchover in case of Node Failure

In case of Node failure, a standby UPF takes over as Active as stated in previous section. The following set of operations take place differently in case of Node failure in terms of the UPF that went down.

1. New UPF VM is brought up and it is similar to bringing up Standby during initial deployment.
2. The UPF comes up only with the day-0 config.
3. CNDP-CM/ESC will push the day0.5 config to the new UPF VM.
4. UPF registers to RCM and then gets assigned with the Standby IP.
5. RCM then parallelly notifies NSO on the availability of Standby UPF.
6. NSO pushes the Day-N common config and all the host Config in the form of config file that is SFTP'd to the UPF.
7. Now the new UPF is a standby that is available for taking over in case of any UPF failure from now on.

The following diagram depicts UPF VM redeployment.



Configuring RCM HA

1. If RCM HA is used, NSO configures both RCMs in the same way. When there is no configuration sync between the RCMs, whatever UPF's host specific configuration that is done on RCM, it should be simultaneously (at least one after another immediately) be applied on both the active and standby RCMs.
2. RCM's day-1 configuration is applied to both RCMs.
3. UPF's host specific configuration is applied to both RCMS.
4. If in case one of the RCM is undergoing reload when host specific is being applied, it should be updated with missed config as soon as it is up.
5. In case, the node hosting RCM itself has gone bad and RCM is redeployed in a new node, RCMs config and UPFs config gets pushed by the NSO. The config should be the same as the active RCM, which did not undergo any reboot.

Upgrading RCM and UP Nodes

The upgrade of the RCM and UP node is required when there is a critical fix or new feature support is needed.

Pre-Requisite: Before any node is upgraded, confirm that both the RCM and UP software versions are compatible.

RCM Upgrade

The RCM Upgrade can be performed using the RCM HA. Use the following procedure for RCM Upgrade.

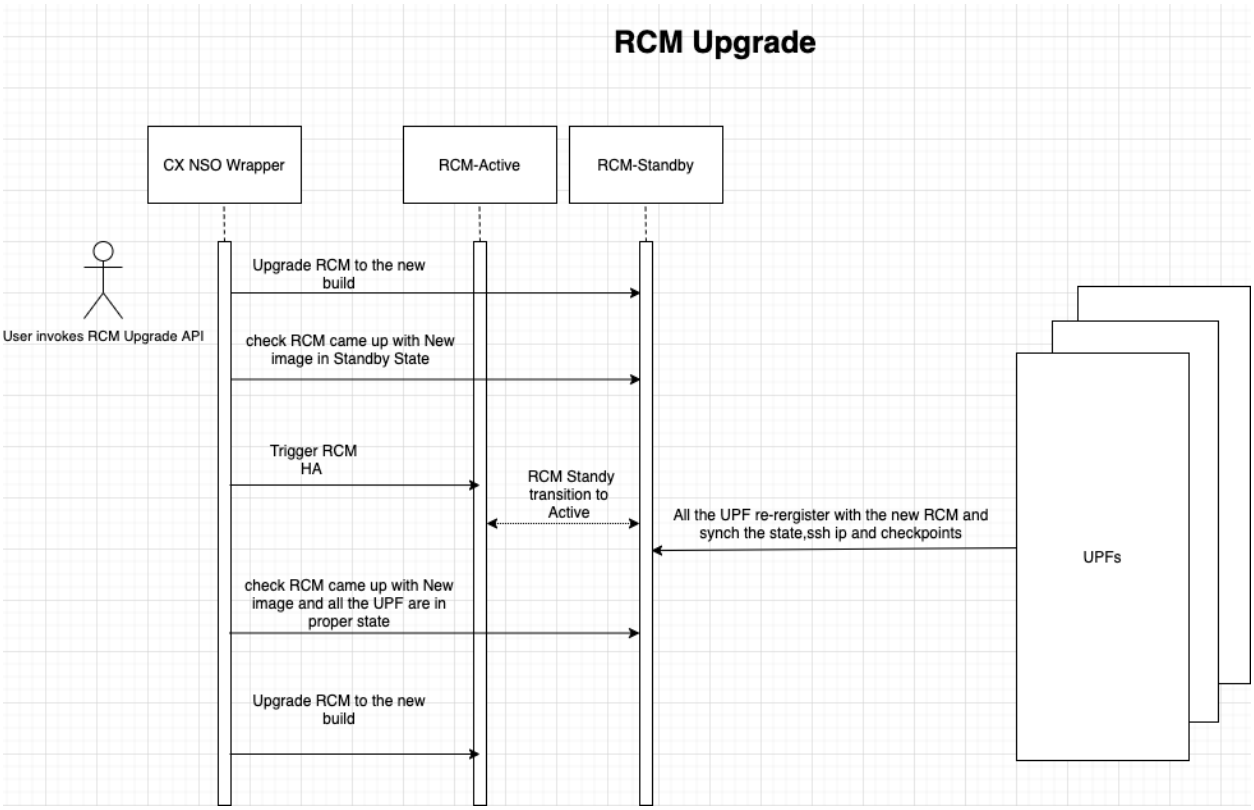
1. Upgrade the Standby RCM to the required RCM build that is compatible with the UP build. (Same procedure used to bring new PoDs similar to AMF, SMF).
2. Wait for the upgraded RCM to come to the Standby state.
   - Verify using the RCM Command about the state of the RCM.

   (opcenter CLI :rcm show-status)
3. Bring down the active RCM that brings the newly upgraded standby RCM as new Active RCM.
   - Bring down the physical interface used for VRRP communication between Active and Standby RCM.
   - Even Cluster Manager can bring down Active RCM keepalived pod to trigger RCM failover?
   - Verify using the RCM Command about the state of the RCM.

   (opcenter CLI: rcm show-status)
4. Wait for All the UPFs to reconnect to the newly upgraded active RCM and synchronize all the UP data.
   - Verify the State of ALL UP in the controller. Ensure there are configured number of UP in Active state and Standby State (N:M).

   (op-center CLI: rcm show-statistics controller)
   - Check if all the data are checkpointed to each of the checkpoint Manager from individual UP perspective.
5. Perform Step 1 and Step 2 on new Standby RCM.

The following diagram depicts RCM Upgrade.



UP UPGRADE

Approach 1- In service upgrade

This approach is the desired approach when we do not want call loss during the complete upgrade procedure. All the UP that are part of the same redundancy group is upgraded one by one.

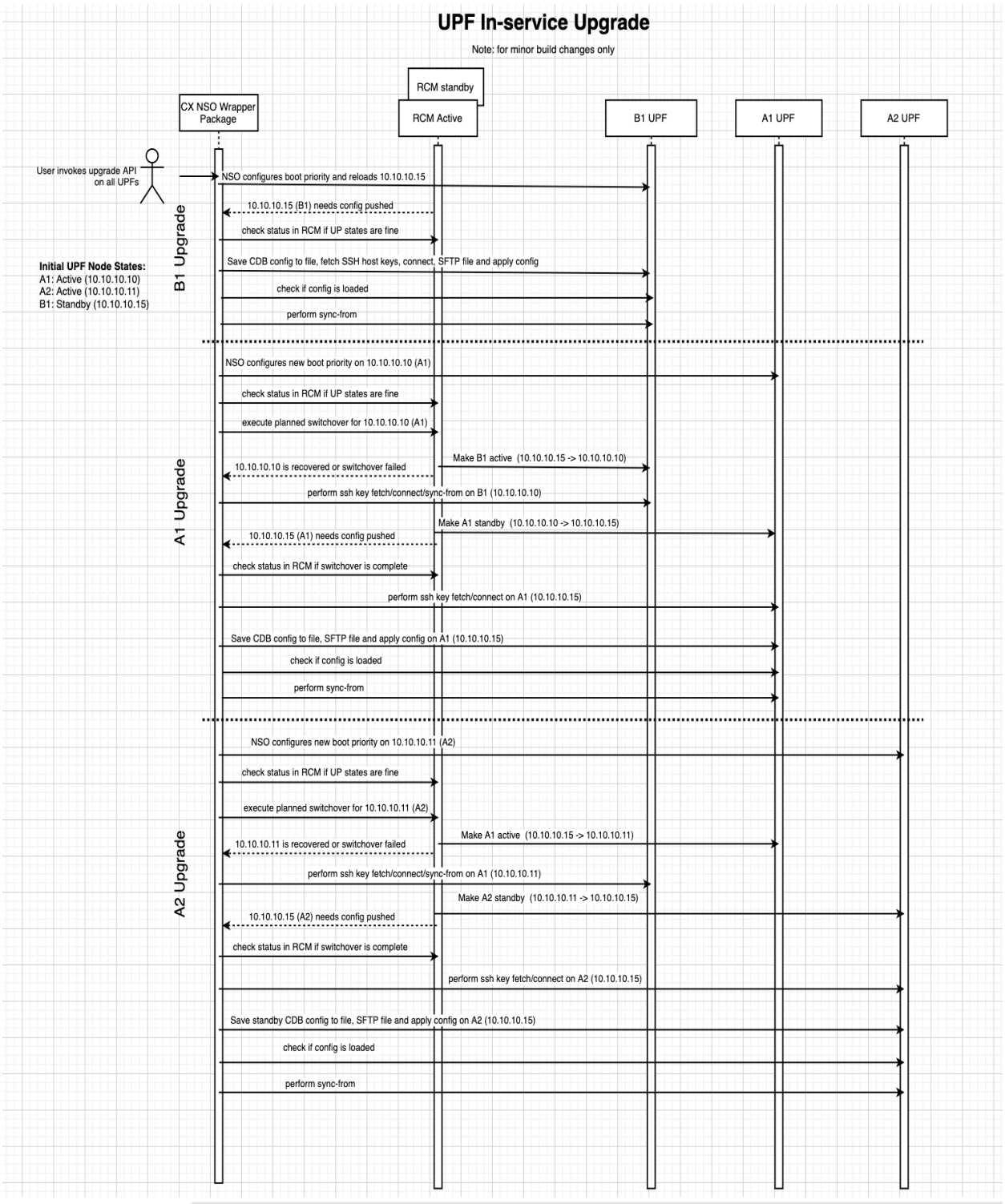
1. Identify the Standby UP from the RCM.
2. (op-center CLI: rcm show-statistics controller)
3. Upgrade the Standby UP with the required StarOS Build that is compatible with RCM by updating the boot config from NSO and trigger a reload.
4. Wait for the UP to come up and get registered with RCM and move to Standby state.
5. RCM intimates CX NSO to push the standby config.
6. CX NSO checks RCM for UP chassis Status is proper.
7. CX NSO fetches the standy UP config from CDB and update the config file along with the host key mappings and sftp the config file to the UP and applies the config.
8. NSO verifies the config is applied in the standby UP properly.
9. NSO performs a sync on the UP.
10. NSO updates the Active UP boot config with the new image to be upgraded.
11. NSO query RCM to check all the UP are in a proper state.

12. NSO instructs RCM to perform a Planned switchover from one of the Active UP to the upgraded Standby UP by triggering the planned switchover from RCM.

rcm switchover source <Source UP endpoint IP> destination <Destination UP endpoint IP>

13. On successful switchover the Standby UP takes the role of the Active UP.
14. RCM intimates the NSO about the successful recovery of old Active.
15. NSO performs a Sync-from the New Active UP.
16. Wait for the old Active UP to reboot and move to Standby State in the RCM.
17. RCM intimates NSO to do a config push for the newly came up Standby UP same as in Step 4 and follows the procedure till Step 8.
18. Perform Step 9 to Step 16 for rest of the UPs that are part of the same redundancy group.

The following call flow illustrates UPF In-service upgrade.



Approach 2 – Out of Service Upgrade

This approach is the desired approach when we do not worry about call loss during the complete upgrade procedure.

There is a new RCM-ops-center CLI added to avoid switchovers of UP during an upgrade. This helps in quickly upgrading the UP compared to a non-disruptive, round-robin approach which may take a considerable amount of time. With this approach, RCM does not perform switchover and lets the UPs reboot and come up with a new image. RCM the informs NSO about the UP and may assign the same host-id and SSH-IP similar to Day-1 deployment(not necessary if customer choose to reboot several UPs at a time), NSO will then configure the UP.

This approach is similar to the case when Standby UP is not available where we do not proceed with the switchover. Here we will additionally check for the RCM ops-center CLI. If it is configured, then we do not switchover when the UP goes down for upgrade and do nothing.

Use RCM ops-center CLI to enable this approach.

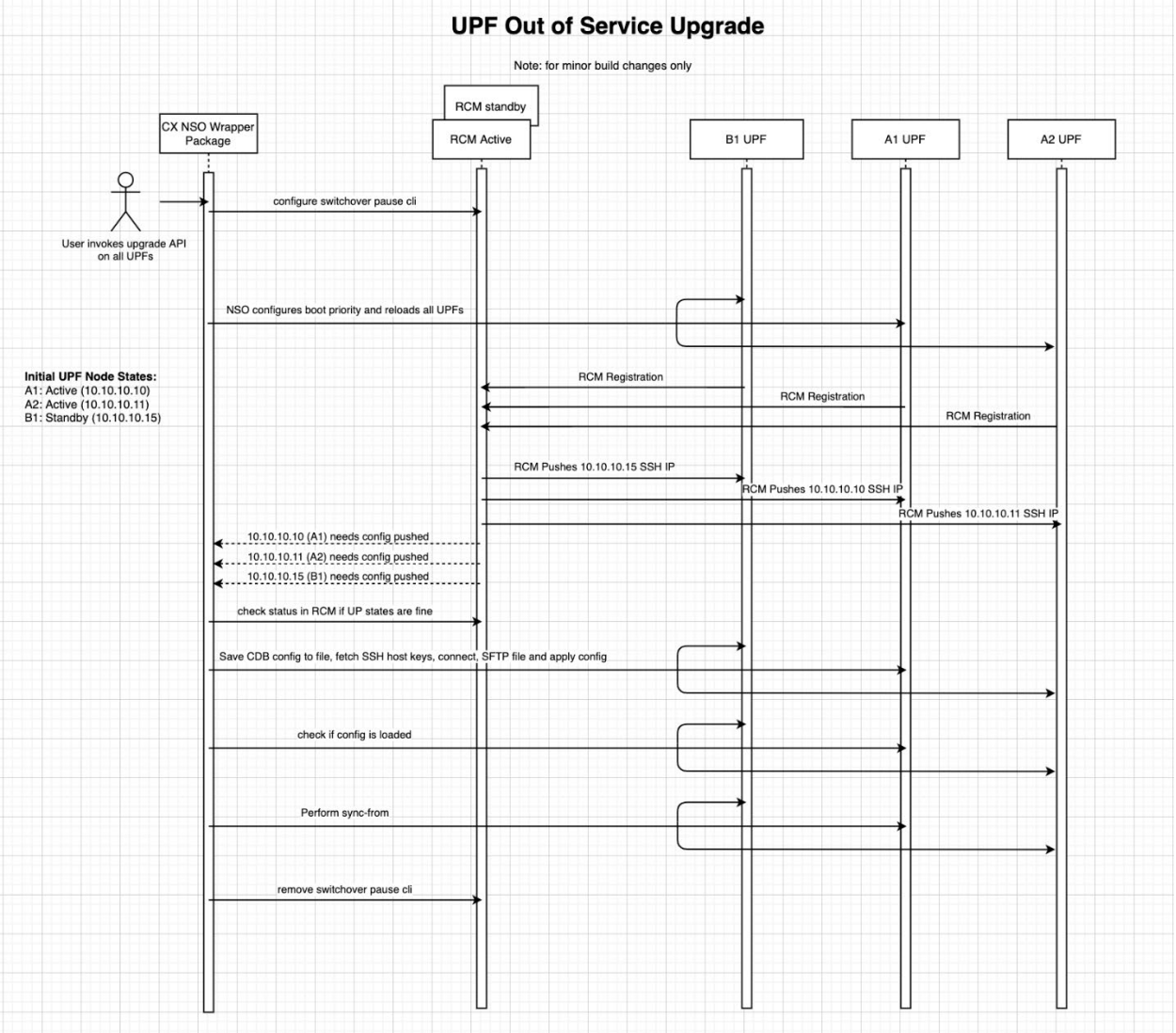
(config)# k8 smf profile rcm-config-ep switchover mw-switchover-pause <true/false>"

Once all the UP are registered, the CLI should be configured with the FALSE option to make the RCM ready for handling switchovers.

The following flow explains the Out of Service Upgrade:

1. NSO configures RCM with "switchover mw-switchover-pasue TRUE" CLI to disable the UP switchover.
2. NSO configures all the UP's boot priority with the image to be upgraded and reloads all the UP's.
3. Once the UP comes up, it registers with the RCM .
4. RCM assigns UP the role as configured in the initial config, this is same as what is explained in the Initial Deployment section.
5. The RCM sends config push notification to NSO for each of the UP.
6. NSO performs the config push and synchronization with all the UP.
7. The NSO cross checks the status of the UP with RCM.
8. NSO configures RCM with "switchover mw-switchover-pasue FALSE" CLI to enable the UP switchover.

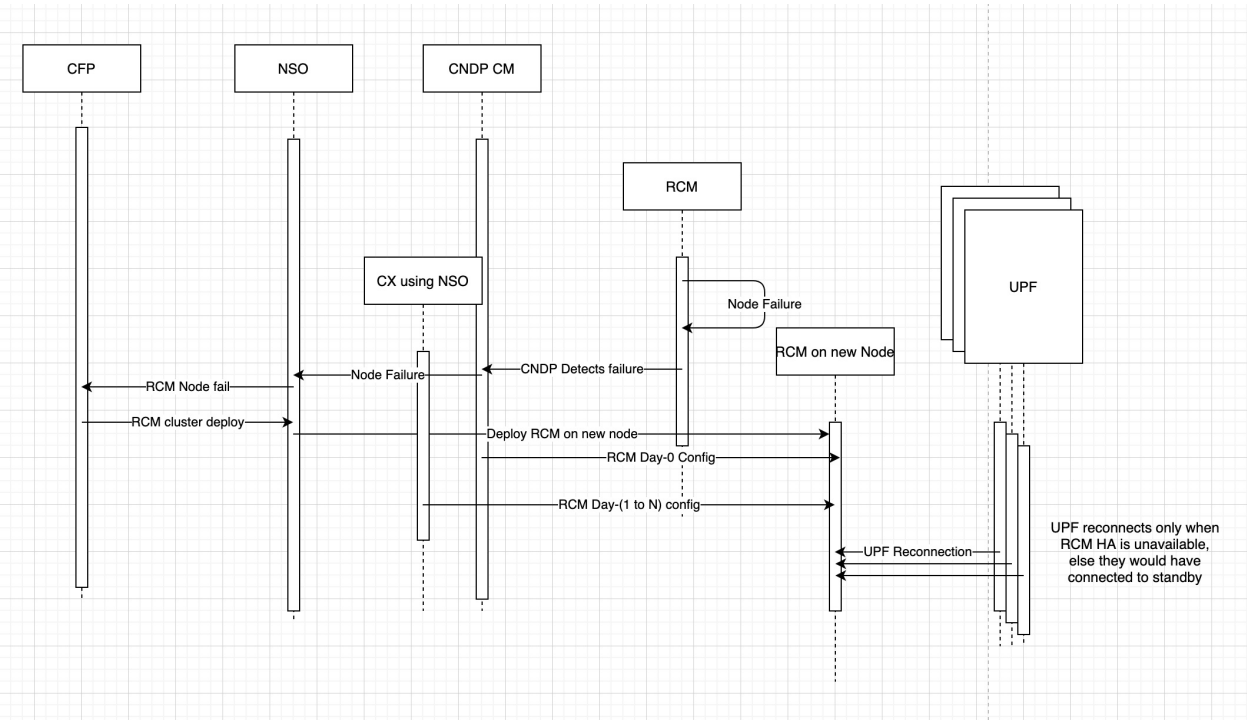
The following call flow illustrates UPF Out of Service upgrade.



RCM RECOVERY

The following sequence of events happens on RCM failure when there is no RCM HA:

1. On RCM Node failure, the Cluster Manager detects the failure and intimates to NSO.
2. NSO CFP is intimated about the RCM node failure.
3. The CFP redeploys the new RCM and waits for RCM to be ready to accept the config.
4. The CX NSO pushes Day 0 and Day 0.5 config to the RCM followed by Day-N config.
5. Since there is no RCM HA, the New RCM learns the UPF state and other essential information as part of UP registration and populates the mapping table for SSH IP and Host Id.
6. Once UPF re-registration is complete, all the checkpoint info is synched from the UP to the new RCM checkpoint manager.



Configmode Selection Config

This config mode is read by both RCM-controller and RCM-configmgr to decide on the method of config push and act accordingly during UP registration to RCM. The default config Mode is RCM.

```
configure
   k8 smf profile rcm-config-ep configMode [NSO /RCM]
```

SSH-IP and Host ID Mapping Config

RCM-Controller reads this config from the config map and frames this hostID to ssh-ip mapping. This mapping is used by RCM-controller to send the hostID and ssh-ip to UP during UP Registration. We also have the management IP of UP in the configuration. This mapping is used by NSO and RCM in the initial deployment of UPs. This is required since NSO CFP only knows the day-0 config, which is the management IP to provide the config template to NSO.

```
(config)# k8 smf profile rcm-controller-ep group 1----
(config-group-1)# k8 smf profile rcm-config-ep host-id active hos32 ssh-ip
1.1.1.3 mgmt-ip 10.10.10.1
(config-group-1)# k8 smf profile rcm-config-ep host-id active host1 ssh-ip
1.1.1.1 mgmt-ip 10.10.10.2
(config-group-1)# k8 smf profile rcm-config-ep host-id active host2 ssh-ip
1.1.1.2 mgmt-ip 10.10.10.3
(config-group-1)# k8 smf profile rcm-config-ep host-id standby sbyhost1 ssh-
ip 1.1.1.4 mgmt-ip 10.10.10.4
```

Dependent Config on StarOS side.

```
configure
   context rcm
      redundancy-configuration-module rcm
      nso-ssh-ip context local interface-name local2 mask 255.255.255.0
Info: Interface_name : local2, mask : 255.255.255.0[rcm]qvpc-si(config-
rcm-service)#
```

SSH Credentials Config

SSH Credentials for the management Ips 192.168.20.43, 192.168.20.44, 192.168.20.45, 192.168.20.46 can be provided in the following style. These credentials are solely read by RCM-configmgr from the config map and the mapping of management IP to the credentials is maintained in the RCM-configmgr. These ssh credentials are used by RCM-configmgr during switchover to push the config (negation of unwanted host config) to standby UP.

(config)# k8 smf profile rcm-config-ep rcm-upfinfo group 1

(config-group-1)# upf mgmt-ip 192.168.20.43 username staradmin password 8MNhYQ9UovHC//TG+V9Rdq/ZGEtrYKnVg0+K/8IuozNk=

(config-group-1)# upf mgmt-ip 192.168.20.44 username staradmin password 8MNhYQ9UovHC//TG+V9Rdq/ZGEtrYKnVg0+K/8IuozNk=

(config-group-1)# upf mgmt-ip 192.168.20.45 username star3 password 8brMUF9Ro7gyBbe0gNdyvOASA8dkquUWup0hL4bB/ysU=

(config-group-1)# upf mgmt-ip 192.168.20.46 username user4 password 8X1XS1ezD4tM6iGMIHigwepplwXgcm1ZtIT62Qpv2c6I=

Config for Maintenance Window

This CLI is used to indicate controller that the UPFs are in maintenance window either for initial deployment or for StarOS upgrades. This CLI can be used for StarOS upgrades only when there are no subscribers on UPFs. If this CLI is configured:

- Controller sends the UPF available notification to NSO with the management IP to SSH IP mapping.
- Controller can also handle UPF down events in a way that there is no switchover triggered for any active UPF going down with this cli configured.

```
configure
     k8 smf profile rcm-config-ep switchover mw-switchover-pause
<true/false>
```

Planned Switchover CLI on RCM

rcm switchover source 5.5.5.2destination 4.4.4.2

The above CLI is removed and two new CLIs are introduced to give the flexibility of using either management ip or endpoint ip to initiate planned switchover.

rcm switchover-mgmt-ip source 192.168.0.4 destination 192.168.0.2

The above cli is introduced to initiate planned UP switchover using the management IP.

192.168.0.4 refers to the management IP of the Source UP

192.168.0.2 refers to the management IP of the destination UP

rcm switchover-vpn-ip source 5.5.5.2 destination 4.4.4.2

This CLI is used to trigger switchover using rcm endpoint IP of the UP. This was originally "rcm switchover source <SOURCE-ENDPOINT-IP> destination <DESTINATION_ENDPOINT-IP>"

Planned Migration CLI on RCM

This CLI is used to do planned migration from active RCM to standby RCM.

rcm migrate primary

CLI to induce planned RCM migration. In case of RCM HA, this CLI initiates a migration activity of RCM. Active RCM VM reboots in case of VM mode. In CNDP mode, the keepalived is moved to fault state relinquishing the Master status. This induces RCM migration.

Save config related StarOS CLIs

Boot config Overwrite

```
configure
   [no] boot config overwrite
   End
```

By configuring this CLI as "no boot config overwrite", one can prevent from overwriting boot config file uding the cli " save configuration". But this can be forced by using -noconfirm option like below:

save configuration -noconfirm

or

autoconfirm

save configuration

CLI to Save APN Config

save configuration apn /flash/apn_config.cfg

NOTE: Unlike normal save configuration, the <URL> while saving apn configuration is mandatory. This change ensures that the boot config or already saved config is not overwritten (unless specified).

CLI to save ACS Config

save configuration acs /flash/acs_config.cfg

NOTE: Unlike normal save configuration, the <URL> while saving acs configuration is mandatory. This change ensures that the boot config or already saved config is not overwritten (unless specified).

Monitor BGP StarOS CLIs

Customer should be given flexibility through CLI to choose whether to reload the UP or not during BGP monitor failure.

```
configure
   context rcm-service-context
      redundancy-configuration-module rcm-service-name
       [no] monitor bgp failure reload standby
```

NOTES:

- **default:** no monitor bgp failure reload standby

This CLI enables / disables reload of Standby UP on BGP monitor failure. Some customers may want to reload the Standby UP to reinitialize and reconfigure the Standby UP on BGP monitor failure. If reload is enabled, then Standby UP sends BGP failure notification to RCM Controller and reload itself. If reload is disabled, Standby UP sends BGP failure notification to RCM Controller and does not reload itself. RCM Controller disconnects TCP towards the Standby UP RCM-Vpnmgr and wait for the Standby UP to re-connect TCP again. Standby UP re-establish TCP only after BGP monitor is up again.

NOTE: Standby UP is marked as unavailable for switchover in RCM Controller if TCP with the Standby UP is down. This CLI is not applicable on Active UPs.

```
configure
    context rcm-service-context
        redundancy-configuration-module rcm-service-name
         [no] monitor bgp failure reload active
```

NOTES:

- **default:** no monitor bgp failure reload active

This CLI enables / disables reload of Active UP on BGP monitor failure and if RCM could not complete a successful UP switchover. If reload is enabled, then the Active UP will send BGP failure notification to RCM Controller and reload itself. If reload is disabled, the Active UP will send BGP failure notification to RCM Controller and will not reload itself. After RCM Controller receives BGP failure notification, RCM will attempt a UP switchover. If UP switchover is successful, RCM Controller will reload the old Active UP. However if UP switchover is not successful, RCM Controller will not take any further action. Note there is no utility in disconnecting TCP in this case as in the Standby UP case where the Standby UP can be marked as unavailable. Further, data black holing is out of scope for RCM in this case. This CLI is not applicable on Standby UPs.

```
   configure
      context rcm-service-context
        redundancy-configuration-module rcm-service-name
        [no] monitor bgp bgp damping-period 1 - 120 seconds
```

NOTES:

- **default:** no monitor bgp damping-period

Time to wait after all BGP peer and peer groups are established for notifying BGP monitor as UP. Also, if within this timeout any BGP peer or BGP peer group flaps, then BGP monitor is not marked as UP and the timer is restarted.

NOTE: Any BGP peer or BGP peer group down will immediately notify BGP monitor down.

Sample combinations of RCM OpsCenter CLIs

Exec Mode

```
rcm route-modifier destination <UPF-IP-Address> val <2-32>
rcm route-modifier destination <UPF-IP-Address> # val=32
```

This CLI is used to change the route modifier (eBGP AS-PATH prepends length) advertised by a UPF. This is most useful after a switchover, when the new Active UPF is at the lowest route modifier (2) and is waiting on the old Active UPF to reboot and re-register so that its route modifier can be wrapped around. Note as long as the UPF is in the lowest route modifier value, it cannot participate in future switchovers. In case the operator observes that route modifier wrap around is not happening, then they need to ensure that the old Active UPF is shut down / restarted and use the above CLI to manually allocate the highest route modifier to the new Active UPF. If the "val" option omitted, then automatically the highest route modifier value (32) is issued.

NOTE: Issue only even numbered route modifiers.

Config Mode

```
k8 smf profile rcm-config-ep config-mode NSO
k8 smf profile rcm-config-ep switchover deployment true reboot true
```

NSO deployment mode.

```
k8 smf profile rcm-config-ep config-mode NSO
k8 smf profile rcm-config-ep switchover deployment true reboot false
```

NSO deployment mode with UPFs manually being rebooted by operator. Note all UPFs need to be rebooted and registered before returning to normal operation mode, otherwise deployment will not function properly.

```
k8 smf profile rcm-config-ep config-mode NSO # optional
k8 smf profile rcm-config-ep switchover mw-switchover-pause false
k8 smf profile rcm-config-ep switchover deployment false
```

Normal operation mode. First line may be configured depending on whether NSO is available for UPF configuration or not.

```
k8 smf profile rcm-config-ep switchover mw-switchover-pause true
k8 smf profile rcm-config-ep switchover deployment false
```

Preventing Multiple Configuration Push Notifications

To prevent multiple configuration push notifications toward NSO, the following CLI command is introduced in 2022.01.1 and later releases:

k8 smf profile rcm-config-ep disable-repeat-config-push { true | false }

By default, the CLI command is set to **false.**

New RCM Traps

- ConnectedStandbysAvailable
- NoConnectedStandbysAvailable
- RouteModifierLowest
- RouteModifierLowestCleared

The existing trap “RCMStateChange” is removed and three new traps were added in its place in this release.

- RCMStateChangeToFault
- RCMStateChangeToActive
- RCMStateChangeToStandby

Show Commands in StarOS

show rcm info

This cli has added info in its CLI Output

```
[local]qvpc-si# show rcm info
Redundancy Configuration Module:
------------------------------------------------------------------------------
--
Context: rcm
Bind Address: 5.5.5.2
Chassis State: Active
Session State: SockActive
Route-Modifier: 32
RCM Controller Address: 5.5.5.1
RCM Controller Port: 9200
RCM Controller Connection State: Connected
Ready To Connect: Yes
Management IP Address: 192.168.0.4
Host ID: active1
SSH IP Address: 2.2.2.2(Activated)


[local]qvpc-si# show rcm info
Redundancy Configuration Module:
------------------------------------------------------------------------------
```

```
--
Context: rcm
Bind Address: 4.4.4.2
Chassis State: Standby
Session State: SockStandby
Route-Modifier: 50
RCM Controller Address: 4.4.4.1
RCM Controller Port: 9200
RCM Controller Connection State: Connected
Ready To Connect: Yes
Management IP Address: 192.168.0.2
Host ID:
SSH IP Address: 2.2.2.11(Activated)
```

Verifying the RCM configuration through NSO Feature Configuration

- Backward compatibility of RCM String based Approach should be verified.
- The new Config push should be validated in all fronts like initial deployment, switchover and Day-N config updates.

RCM Configuration through NSO Feature OA&M Support

Sample RCM (NSO Specific) and Host specific config is configured on RCM

```
Configure
k8 smf profile rcm-config-ep deployment-mode VM
k8 smf profile rcm-config-ep switchover deployment true reboot true
k8 smf profile rcm-config-ep config-mode NSO
k8 smf profile rcm-config-ep disable-cm apn gtpp creditCtrl packetFilter
urrList ruledef rulebase miscacs global chargingAction lawfulIntercept
apnprofile
k8 smf profile rcm-config-ep rcm-upfinfo group 1
 host-id active Active1 ssh-ip 11.22.33.44 management-ip 10.126.82.200
 host-id active Active2 ssh-ip 11.22.33.45 management-ip 10.126.82.94
 host-id standby Standby1 ssh-ip 11.22.33.46 management-ip 10.126.82.138
exit
control-plane-group g1
 redundancy-group 1
  host Active1
   peer-node-id ipv4-address 192.168.196.10
  exit
  host Active2
   peer-node-id ipv4-address 192.168.196.10
```

```
    exit
   exit
  exit
  context EPC2
   interface-loopback S5_SGW_PGW_loopback_up1
    redundancy-group 1
     host Active1
      ipv4-address 192.168.170.77/32
     exit
    exit
   exit
   interface-loopback S5_SGW_PGW_loopback_up2
    redundancy-group 1
     host Active2
      ipv4-address 192.168.170.99/32
     exit
    exit
   exit
   interface-loopback pgw-ingress-ipv6-loopback_up1
    redundancy-group 1
     host Active1
      ipv6-address bbbb:aaaa::14/128
     exit
    exit
   exit
   interface-loopback pgw-ingress-ipv6-loopback_up2
    redundancy-group 1
     host Active2
      ipv6-address bbbb:aaaa::16/128
     exit
    exit
   exit
   interface-loopback pgw-ingress-loopback_up1
    redundancy-group 1
     host Active1
      ipv4-address 192.168.170.14/32
```

```
    exit
   exit
  exit
  interface-loopback pgw-ingress-loopback_up2
   redundancy-group 1
    host Active2
     ipv4-address 192.168.170.16/32
    exit
   exit
  exit
  interface-loopback sgw-egress-ipv6-loopback_up1
   redundancy-group 1
    host Active1
     ipv6-address bbbb:aaaa::12/128
    exit
   exit
  exit
  interface-loopback sgw-egress-ipv6-loopback_up2
   redundancy-group 1
    host Active2
     ipv6-address bbbb:aaaa::14/128
    exit
   exit
  exit
  interface-loopback sgw-egress-loopback_up1
   redundancy-group 1
    host Active1
     ipv4-address 192.168.170.51/32
    exit
   exit
  exit
  interface-loopback sgw-egress-loopback_up2
   redundancy-group 1
    host Active2
     ipv4-address 192.168.170.53/32
    exit
```

```
 exit
exit
interface-loopback sgw-ingress-ipv6-loopback_up1
 redundancy-group 1
  host Active1
   ipv6-address 2001::1:48/128
  exit
 exit
exit
interface-loopback sgw-ingress-ipv6-loopback_up2
 redundancy-group 1
  host Active2
   ipv6-address 2001::1:50/128
  exit
 exit
exit
interface-loopback sgw-ingress-loopback_up1
 redundancy-group 1
  host Active1
   ipv4-address 101.101.102.48/32
  exit
 exit
exit
interface-loopback sgw-ingress-loopback_up2
 redundancy-group 1
  host Active2
   ipv4-address 101.101.102.50/32
  exit
 exit
exit
interface-loopback sx-u-ipv6-loopback_up1
 redundancy-group 1
  host Active1
   ipv6-address bbbb:aaaa::33/128
  exit
 exit
```

```
exit
interface-loopback sx-u-ipv6-loopback_up2
 redundancy-group 1
  host Active2
   ipv6-address bbbb:aaaa::35/128
  exit
 exit
exit
interface-loopback sx-u-loopback_up1
 redundancy-group 1
  host Active1
   ipv4-address 192.168.170.131/32
  exit
 exit
exit
interface-loopback sx-u-loopback_up2
 redundancy-group 1
  host Active2
   ipv4-address 192.168.170.33/32
  exit
 exit
exit
user-plane-service up_up1
 redundancy-group 1
  host Active1
   associate control-plane-group g1
   associate fast-path service
   associate sx-service sx_up1
   associate gtpu-service pgw-gtpu_up1 pgw-ingress
   associate gtpu-service saegw-sxu_up1 cp-tunnel
   associate gtpu-service sgw-engress-gtpu_up1 sgw-egress
   associate gtpu-service sgw-ingress-gtpu_up1 sgw-ingress
  exit
 exit
exit
user-plane-service up_up2
```

```
 redundancy-group 1
  host Active2
   associate control-plane-group g1
   associate fast-path service
   associate sx-service sx_up2
   associate gtpu-service pgw-gtpu_up2 pgw-ingress
   associate gtpu-service saegw-sxu_up2 cp-tunnel
   associate gtpu-service sgw-engress-gtpu_up2 sgw-egress
   associate gtpu-service sgw-ingress-gtpu_up2 sgw-ingress
  exit
 exit
exit
gtpu-service pgw-gtpu_up1
 redundancy-group 1
  host Active1
   bind ipv4-address 192.168.170.14
  exit
 exit
exit
gtpu-service pgw-gtpu_up2
 redundancy-group 1
  host Active2
   bind ipv4-address 192.168.170.16
  exit
 exit
exit
gtpu-service saegw-sxu_up1
 redundancy-group 1
  host Active1
   bind ipv4-address 192.168.170.31
  exit
 exit
exit
gtpu-service saegw-sxu_up2
 redundancy-group 1
  host Active2
```

```
      bind ipv4-address 192.168.170.33
     exit
    exit
   exit
   gtpu-service sgw-engress-gtpu_up1
    redundancy-group 1
     host Active1
      bind ipv4-address 192.168.170.51
     exit
    exit
   exit
   gtpu-service sgw-engress-gtpu_up2
    redundancy-group 1
     host Active2
      bind ipv4-address 192.168.170.53
     exit
    exit
   exit
   gtpu-service sgw-ingress-gtpu_up1
    redundancy-group 1
     host Active1
      bind ipv4-address 101.101.102.48
     exit
    exit
   exit
   gtpu-service sgw-ingress-gtpu_up2
    redundancy-group 1
     host Active2
      bind ipv4-address 101.101.102.50
     exit
    exit
   exit
   sx-service sx_up1
    sx-protocol heart-beat interval 100
    sx-protocol heart-beat max-retransmissions 10
    sx-protocol heart-beat retransmission-timeout 20
```

```
    redundancy-group 1
     host Active1
      bind ipv4-address 192.168.170.77
      instance-type userplane
     exit
    exit
   exit
   sx-service sx_up2
    sx-protocol heart-beat interval 100
    sx-protocol heart-beat max-retransmissions 10
    sx-protocol heart-beat retransmission-timeout 20
    redundancy-group 1
     host Active2
      bind ipv4-address 192.168.170.99
      instance-type userplane
     exit
    exit
   exit
  exit
  context billing
   edr-module active-charging-service
    redundancy-group 1
     host Active1
      cdr purge storage-limit 110
      cdr transfer-mode push primary url
  sftp://ftpxfer:ftpxfer@10.135.6.32:/data0/source/SEPGW6000 source-address
  10.192.150.89
      file name SEPGW600-V9 rotation volume 60000000 rotation time 600
  storage-limit 536870912 headers edr-format-name compression gzip charging-
  service-name omit trap-on-file-delete
     exit
     host Active2
      cdr purge storage-limit 111
      cdr transfer-mode push primary url
  sftp://ftpxfer:ftpxfer@10.135.6.32:/data0/source/SEPGW6001 source-address
  10.192.150.90
      file name SEPGW600-V10 rotation volume 60000001 rotation time 600
  storage-limit 536870912 headers edr-format-name compression gzip charging-
  service-name omit trap-on-file-delete
```

```
    exit
   exit
  exit
  lawful-intercept redundancy-group 1
    host Active1
      src-ip-addr 21.21.21.21
    exit
    host Active2
      src-ip-addr 21.21.21.22
    exit
  exit
exit
ts-bind-ip redundancy-group 1
  host Active1 Active1 ipv4-address 12.12.12.12ipv6-address
2001:2345:abcd::12
  host Active2 Active2 ipv4-address 12.12.12.13ipv6-address
2001:2345:abcd::13
exit
exit
end
```

Monitoring and Troubleshooting RCM

This section provides information about monitoring and/or troubleshooting the RCM.

Show Commands and/or Outputs

The following table lists the show CLI commands that can be used to gather RCM statistics.

Statistics/Information	Show CLI commands	Node (where CLI should be executed)
Information on the status of chassis, session, RCM controller, and so on.	**show rcm info**	UP
Information on the status of BGP and BFD monitor.	**show rcm monitor { bfd | bgp | all }**	UP
RCM checkpoint details.	**show rcm checkpoint { info | statistics { active | debug-info | ipsecmgr { all | instance** *ipsecmgr_instance_number* **} | sessmgr { all | instance** *sessmgr_instance_number* **} | standby | verbose } }**	UP
Statistics of RCM Controller pod.	**rcm show-statistics controller**	RCM Ops Center
Statistics of RCM ConfigMgr pod.	**rcm show-statistics configmgr**	RCM Ops Center
Statistics of UP BFD status.	**rcm show-statistics bfdmgr**	RCM Ops Center
Statistics of RCM checkpointmgr pod.	• **rcm show-statistics checkpointmgr-endpoint-upfAddr** *ipv4_address* • **rcm show-statistics checkpointmgr-endpointstats** • **rcm show-statistics checkpointmgr-session-upfAddr** *ipv4_address*	RCM Ops Center
Statistics of Switchover	• **rcm show-statistics switchover** • **rcm show-statistics switchover-verbose**	RCM Ops Center
Summary of all RCM show commands	**rcm support-summary**	RCM Ops Center

SNMP Traps

RCM supports event-based SNMP traps to notify important events to an external Network Management System (NMS)/SNMP Manager by using “rcm-snmp-trapper” pod. This pod is based on the snmp-trapper module in smi-apps and supports outbound alerting.

Controller Pod

The following traps are generated by the Controller Pod:

- UPFRegistered
- SwitchoverTriggered
- SwitchoverFailure
- SwitchoverComplete
- BootTimerExpired
- MassUPFailure
- TCPConnect
- TCPDisconnect
- Upfbootcomplete
- UPFReloaded

Configuration Manager Pod

The following traps are generated by the Configuration Manager (ConfigMgr) Pod:

- UPFAdded
- UPFDeleted
- SwitchoverAbortedByController
- SwitchoverFailure
- UPFCfgPushComplete
- UPFCfgPushFailure

Checkpoint Manager Pod

The following traps are generated by the Checkpoint Manager (ChkpointMgr)/Redundancy Manager (RedMgr) Pod:

- ActiveSessmgrConnected
- ActiveSessmgrDisconnected
- StandbySessmgrConnected
- StandbySessmgrDisconnected
- CheckpointAuditStarted
- CheckpointAuditEnded
- CheckpointFlushCompleted

Keepalived Pod

The following traps are generated by the Keepalived Pod:

- PodNotFound

- PodNotRunning

  Important NOTE: In 21.23.x and later releases, the PodNotFound, and PodNotRunning traps are obsoleted. As replacement, the following new trap is added:

  - RCMPodHealthCheck

- RCMStateChange

  Important NOTE: In 21.23.x and later releases, the RCMStateChange trap is obsoleted. As replacement, the following new traps are added:

  - RCMStateChangeToFault
  - RCMStateChangeToActive
  - RCMStateChangeToStandby

strongSwan Manager Pod

The following traps are generated by the strongSwan Manager pod:

- TunnelsDropped
- TunnelsAdded

Configuring SNMP Traps

Use the following commands to configure SNMP Trap and various parameters.

```
configure

   endpoint rcm-snmp-trapper

   exit

k8 smf profile rcm-snmp-trapper-ep snmp-trapper { enable | disable-trap
trap_name | v2c-target ip_address { community public_string [ port
port_number ] | port port_number } | v3-engine-id id_string | v3-target
ip_address [ auth { md5 [ auth-key | port | priv { aes | aes192 | aes256 |
des | none } | user-name ] | none | sha } | port | user-name }
```

NOTES:

- **enable**: Enables SNMP trapper.
- **disable-trap** *trap_name*: Disables the desired SNMP trap.
- **v2c-target** *ip_address*: Specifies the list of SNMP v2c trap receivers. The *ip_address* specifies the SNMP Trap Receiver hostname or IP address.
- **v3-engine-id** *id_string*: Specifies the source engine ID for v3 traps as hex string, such as 80004f. *id_string* must be a string of minimum five and maximum 32 characters.
- **v3-target**: Specifies the list of SNMP v3 trap receivers.
- **community**: Specifies the SNMP Trap Receiver community.
- **port**: Specifies the SNMP Trap Receiver port.

- **auth**: Specifies the authentication protocol to be used.
- **user-name**: Specifies the SNMP trap username.
- **md5**: Specifies that HMAC-MD5-96 authentication protocol is used.
  - **none**: Specifies that no authentication protocol is used.
  - **sha**: Specifies that HMAC-SHA-96 authentication protocol is used.
  - **auth-key**: Specifies that key to authentication protocol.
  - **priv**: Specifies that privacy protocol is used.
    - **aes**: Specifies that AES-CFB (128 bits) protocol is used.
    - **aes192**: Specifies that AES-CFB (192 bits) protocol is used.
    - **aes256**: Specifies that AES-CFB (256 bits) protocol is used.
    - **des**: Specifies that CBC-DES protocol is used.
    - **none**: Specifies that no privacy protocol is used.

Sample SNMP Trap

The following is a sample SNMP trap:

```
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (43241) 0:07:12.41
SNMPv2-MIB::snmpTrapOID.0 = OID: CISCO-RCM-MIB::rcmFaultActiveNotif
CISCO-RCM-MIB::rcmFaultId = STRING: "SwitchoverTriggered"     CISCO-RCM-
MIB::rcmFaultSource = STRING: "rcm-controller"   CISCO-RCM-
MIB::rcmFaultSeverity = STRING: "Major"       CISCO-RCM-MIB::rcmFaultTime =
STRING: 2572-0-0,0:43:0.0 CISCO-RCM-MIB::rcmFaultType = INT: 3  CISCO-RCM-
MIB::rcmFaultAdditionalInfo = STRING: "Switchover Triggered"        CISCO-
RCM-MIB::rcmFaultClusterName = STRING: "k3scluster"       CISCO-RCM-
MIB::rcmFaultNamespace = STRING: "rcm"        CISCO-RCM-MIB::rcmFaultHostname
= STRING: "host1"       CISCO-RCM-MIB::rcmFaultInstance = STRING: "host1"
CISCO-RCM-MIB::rcmVnfAlias = STRING: "*"
```

For information about MIBs, refer Appendix C: MIBs

RCM High Availability

Feature Description

The Redundancy and Configuration Management (RCM) provides a High Availability (HA) solution for User Planes (UPs). RCM enables support for N:M redundancy which minimizes the number of redundant data UPs required for your deployment. To prevent UP session loss resulting from unplanned RCM outages, the RCM can be deployed in HA mode.

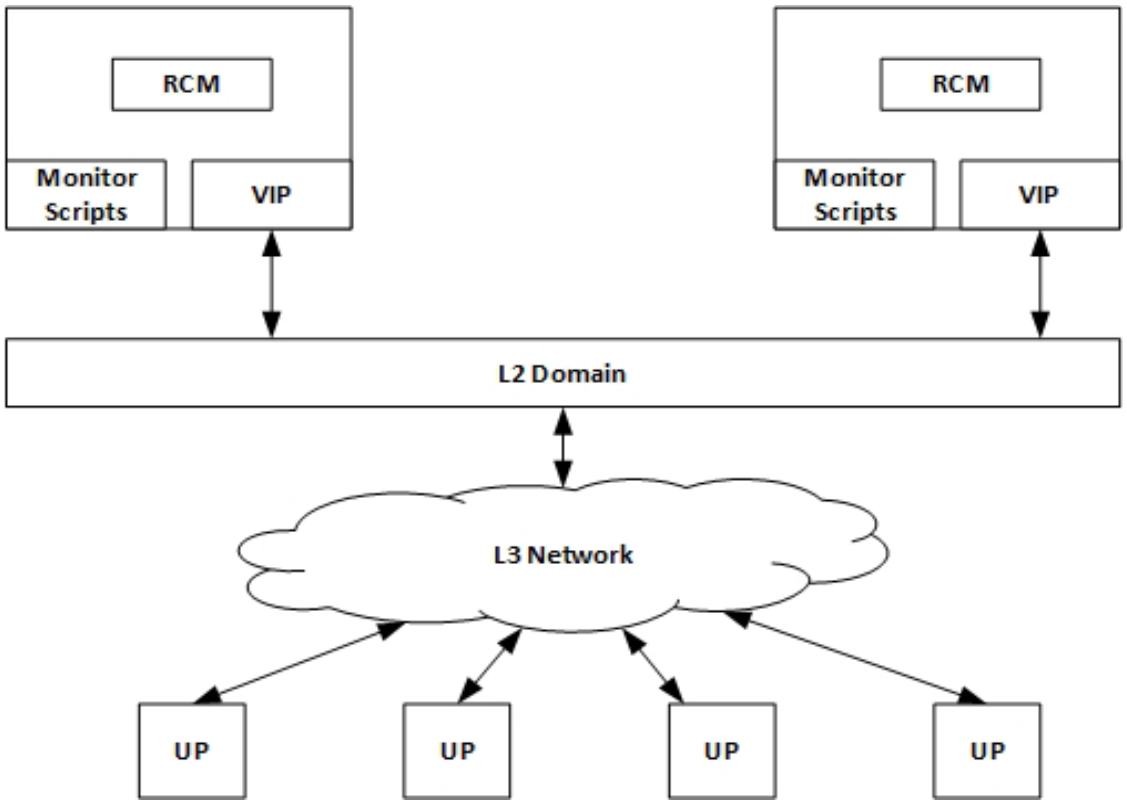
Architecture

RCM HA is achieved by deploying two instances of the RCM. The RCM instances run in active and standby mode. The external IPs for each instance are configured as Virtual IPs (VIP addresses) each running the Virtual Router Redundancy Protocol (VRRP).

Depending on the Primary RCM node (e.g., the Primary VRRP), one of the RCM instances assumes the active role and the other RCM instance assumes the Standby role.

When the active RCM is unavailable, then the VRRP selects the standby RCM as Primary, and this instance becomes the new active RCM. The UP automatically connects to the new RCM (as the IP address moves to new RCM). All UPs connected to the previously active RCM experience connection resets and, upon retry, they get connected to the new RCM.

The following image illustrates the RCM HA architecture.



RCM HA deployments require the installation of two RCM instances instead of one. To achieve high-speed communication between the UPs and RCM, and to ensure that the UP-switchover time is as minimal as possible, both RCM should be deployed in the same datacenter as the UPs for which they are providing redundancy.

How it Works

RCM HA support is based on VRRP. VRRP dynamically determines which RCM instance serves as the Primary (active) or standby (Backup) based on the assignment of the default route address.

VRRP selects the RCM instance that receives the IP address based on several factors, such as the host priority, first to be active, instance interface state (for example, “Up”), and so on. VRRP provides many options for customizing the rules for determining the Primary and Backup role for the RCM instance.

The RCM architecture is based on UP-to-RCM communication using UDP (BFD) and TCP (Controller, Checkpoint Manager) connections. The UPs are configured with BFD, Controller, and CheckpointMgr IP addresses. The UPs use these IP addresses to communicate with the RCM.

When the RCMs are deployed with HA, BFD, Controller, and Checkpoint IP addresses are configured as one IP address. This IP address, also known as external IP address, is then configured as a VIP in RCM. The VIP is managed by VRRP which determines the RCM that owns the VIP. All UPs use VIP address to communicate with the RCM. The RCM instance which owns the VIP becomes active and the other RCM instance becomes Standby.

When the active RCM reboots or relinquishes its Primary role, the VIP moves to the standby instance resulting in the standby RCM becoming active and the UPs reconnecting to newly active RCM.

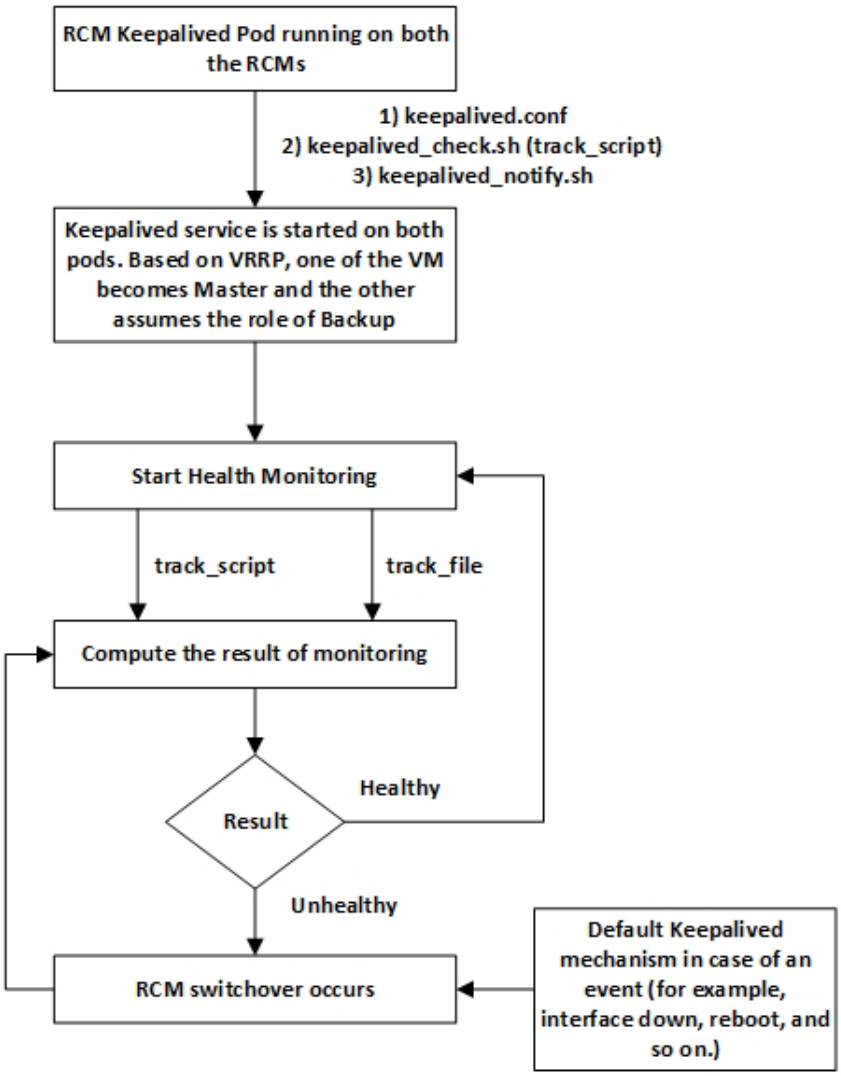
The RCM’s Controller Pod learns the state of the UPs and communicates the same to the Configmgr. The Controller Pod ensures consistency of operation and decides when/if to reboot a UP to restore the system to a healthy state. UP reboots are reserved for rare, corner case scenarios.

For double-fault scenarios, (for example, when the switchover process is left in hung state and UP reboots itself after detecting it’s in hung state), the Controller sends a START_SWITCHOVER message to the UP and starts the rest of switchover process after getting an Ack from UP.

Keepalived

Keepalived is a software implementation of VRRP on Linux that runs as a pod on the RCM VM. VRRP uses the concept of a VIP. One or more hosts (routers, servers, and so on) participates in an election to determine the host that will control the VIP. Only one host (Primary) controls the VIP at a time. If the Primary host fails, the VRRP provides mechanisms for detecting that failure and quickly failing over to a Standby host.

The following flow diagram illustrates how RCM achieves HA using Keepalived.



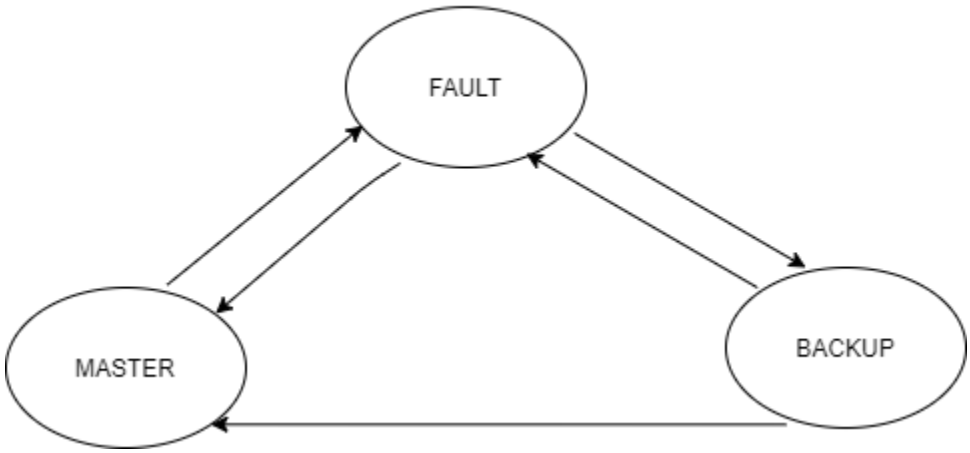
After the Keepalived pod is up and running on both the VMs, one of the RCM VMs comes up as Primary and another as Backup. Keepalived leverages the following files for proper operation:

1. **keeplived.conf**—This is the Keeplalived service configuration file. It contains various Keepalived keywords to tune the service and to track VRRP (using track_file and track_script options) and notify script (keepalived_notify.sh) to run when VRRP instance changes the State. VIP details are part of the configuration applied through the RCM Ops Center.
2. **keepalived_notify.sh**—This is specified using the notify_script keyword mentioned in keepalived.conf file. This script takes actions when the RCM VM changes any state. For example, it specifies what action must be taken when the RCM VM transitions from Primary to a Fault State.

Both keepalived.conf and keepalived_notify.sh are a part of the Keepalived container inside the Keepalived pod. Keepalived also tracks a file using track_file option to keep a check on overall system health. When BFD detects all UPs are down, the RCM controller notifies Keepalived through track_file, the RCM is then deemed unhealthy using keepalived mechanism, and an RCM switchover occurs.

There is a possibility for events other than what is explicitly specified as part of track_file to occur (such as, VM reboot, interface going down, and so on). During such unspecified events, an RCM switchover event is triggered.

The following figure illustrates the States that the VRRP instance in RCM VM can transition to.



After the Keepalived pod is up and running, one of the RCM VM instances comes up as Primary and the other as Backup. When Keepalived health monitoring fails on Primary, it transitions to Fault state first and then to Backup state. At the same time, Backup transitions to Primary role. Whenever the state transitions to Fault state, the VM is rebooted. This action is done by keepalived_notify script.

RCM Upgrade and Downgrade Procedure in HA

In the old RCM VM:

- Take a backup of the running configuration.
- Take a backup of the configuration folders—master, host, svc-type—from */var/rcm/scripts/config*

In the new RCM VM:

To upgrade the RCM image in case of qcow2, delete the old vol-image, create a new image, and restart the VM.

- Configure the back-up ops-center configuration
- Copy the backed-up contents of master/host/svc-type folders to their respective folders
- Perform system mode running

RCM Pause Switchover Command per Redundancy Group

RCM Pause Switchover functionality is used when there is a UP(F) reload/upgrade due to maintenance, or any other reason, and we don't want UP(F) switchover to occur.

RCM Pause Switchover Command from RCM Ops Center is applicable for all Redundancy Groups managed by the RCM. To pause UP(F) switchover only for a particular Redundancy Group, the RCM Pause Switchover Command must take the Redundancy Group number as an argument.

To achieve this functionality, the following command is introduced in Exec Mode starting 2022.01.2 and later releases:

rcm pause switchover { true | false } [red-group *red_group_number* **]**

Where:

- **red-group** *red_group_number* is an optional keyword/variable. Arbitrary redundancy group numbers are allowed (subject to minimum value of 1). If you do not use this keyword, or if the value of *red_group_number* is 0, switchover is paused for all redundancy groups.
- If **rcm pause switchover** is set to **true** for any maintenance activity or other reasons, it should be reverted to **false** immediately after the activity. If the value is set as **true**, any RCM switchover is paused. The **rcm show-statistics controller** CLI command displays the current **pause_switchover** value. For example,

```
rcm# rcm show-statistics controller
Fri May  20 11:04:21.603 UTC+00:00
message :
{
  "keepalive_version":
"94595c576dda83ba078fd581747ebc25726c233f237255781dcc5361d6405a63",
  "keepalive_timeout": "20s",
  "num_groups": 1,
  "groups": [
    {
      "groupid": 1,
      "endpoints_configured": 2,
      "standby_configured": 1,
      "pause_switchover": false,
      "active": 1,
      "standby": 0,
      "endpoints": [
...
```

In earlier releases, the following was the command in Configuration Mode:

k8 smf profile rcm-config-ep switchover mw-switchover-pause { true | false }

Prerequisites and Assumptions

RCM HA operation is based on the following prerequisites/assumptions:

- The Operator must reconfigure any delta configurations that was done on one RCM while the other RCM was in rebooting state.
- The Operator must configure both RCM instances in an identical way for updates.
- The Operator need not track which RCM instance is Active and which one is Standby.
- Only one VIP is configured per RCM. This VIP serves as IP for Controller, BFD, and Checkpoint Manager.
- All VRRP requirements, as specified in RFC3768, are needed to implement RCM HA solution. It includes:

  - The interface on which the VIP is defined for both RCM instances are part of same L2 domain.
  - The L2 switch to which the VIP interfaces are connected are multicast enabled.
- The UP is configured with five minutes BFD timeout by default. If the UP does not find RCM within the timeout period, the UP gets rebooted.
- The VRRP protocol ensures that RCM does not end up in active-active state. RCM Backup-Backup state is possible if both VIP interfaces are down.
- A new or rebooted UP may have to wait up to five minutes before getting registered with the RCM. This is expected during initial RCM bring-up and during RCM switchover.
- RCM HA does not support hot-standby wherein checkpoints are synchronized between RCM instances. RCM HA supports cold-standby. That is, upon switchover, new RCM re-learns checkpoints from the UPs.
- RCM HA deployments do not assume the availability of external persistent storage.

Operational Flow

This section describes the scenarios and operation of RCM HA solution.

RCM HA Deployment

The following points describe the operational flow within RCM HA deployments:

1. The Operator brings up the RCM VMs and applies the required configurations (common and host-specific).
- **NOTE**: The Operator reconfigures the RCM whenever it reboots.
2. One of the RCM VMs becomes active because the VRRP selects that RCM instance as Primary for the following reasons:
   - The RCM instance that comes up first becomes the Primary (provided VIP interface is up) and the other RCM that comes up later becomes Backup, OR
   - If both RCM instances come up simultaneously, then the VRRP uses a tie-breaker algorithm to select one of the RCM instances to be the Primary.
3. The active RCM starts receiving connection requests (e.g. BFD messages, Controller Request, etc.) from the UPs.
4. All UPs start registering with the active RCM.
5. After registration is complete, the active RCM starts assigning active/standby roles to the UPs and configures them based on their role.
6. Active UPs start the checkpoint operation with the active RCM, and the active RCM starts managing the standby UPs.
7. When the active RCM crashes or relinquishes its role as Primary, the Backup RCM becomes active. The old active RCM either reboots or restarts the Controller Pod.

   Should an RCM switchover occur, the UP detects a BFD failure and brings down the TCP connection with the Controller and Checkpoint Manager.

   NOTE: The RCM switchover is triggered by moving the VIP. For UPs, the switchover is merely detected as connection failure and UPs retry the connection.
8. The UPs initiate BFD sessions with the newly active RCM and re-establish the controller TCP connection with it.
9. The UPs share information with the newly active RCM:

- Their role (for example, Active or Standby)
- The host-ID allocated to it (in case it was active)
- Their configuration state (for example, whether a UP received the configuration or not)

10. The newly active RCM builds the state information with the information supplied by the UPs.
11. The newly active RCM resumes operation based on state information:
    - For active UPs, the it starts a full checkpoint operation.
    - For registering UPs, it determines the role and host configuration.
    - For UPs that did not receive the full configuration, it reconfigures the UP and it determines the role for new RCM.

RCM Switchover During UPs Registration/Configuration

While UP registration/Day-1 configuration is in progress, one or more of State, Host ID, and "Config Pushed" flag might not be updated on the UP. As such, the UP publishes the saved State, Host ID, and "Config Pushed" flag to the new Active RCM:

- If the State is Active, the Host ID is not null, and the Pushed flag is true –> The RCM Controller accepts the information and updates the Database. It notifies the Configuration Manager service with this information and the Redundancy Manager about the state.
- If the State is active, the Host ID is not null, and the Pushed flag is false –> The RCM Controller accepts the State of the UP and informs the Configuration Manager to push the configuration again.
- If the State is Active and the Host ID is null –> The RCM Controller informs the UP to reboot.
- If the State is Standby and the Pushed flag is true –> The RCM Controller accepts the information and updates the database. (For Standby, the Host ID is ignored, and the Standby cannot represent an Active host configuration.) It then notifies the Configuration Manager and the Redundancy Manager about the State.
- If the State is Standby and the Pushed flag is false –> The RCM Controller either informs the Configuration Manager to push the configuration to the UP again or reboots the UP.

RCM Switchover During UP Switchover

1. Active UP failure is detected but switchover is not initiated: This is a scenario wherein the Active RCM detects the UP failure, however, before the UP switchover can be initiated, the Active RCM fails. In such scenarios, the new Active RCM receives Init request from the failed UP (after it reboots), and the new Active RCM assigns the appropriate role.

- **NOTE**: Loss of sessions are expected in this scenario as the failed UP could not be switched over to standby UP.

2. Active UP failure is detected, and switchover is initiated: This is a scenario wherein the Active RCM detects the UP failure and it starts the switchover. However, Active RCM fails before switchover is complete. In such scenarios, the new Active RCM receives Init request from failed UP (after it reboots), and the new Active RCM assigns the appropriate role to the UP. However, the standby UP which received partial checkpoints eventually detects that it's in hung state and so, it reboots. Upon completion of reboot, it registers with new Active RCM and the new Active RCM assigns appropriate role to the UP.

   NOTE: Total loss of sessions is expected for the failed UP as switchover is not completed.

RCM Switchover During Provisioning of UP

Scaling up and scaling down of UPs is not supported for RCM.

RCM Switchover During Decommissioning of UP

Scaling up and scaling down of UPs is not supported for RCM.

Reboot of both RCMs

When both the RCMs are unavailable, the BFD restart timers on the UPs reset to a value such that UPs can continue serving until one of the RCMs is available.

RCM Switchover during Loss of Connectivity to Active UP

The Active UP loses connectivity to the RCM before the RCM switchover, the UP switchover completes, and the old Active UP reconnects to new Active RCM. The new Active UP and old Active UP have the same Host ID. The new Active RCM chooses to keep the UP with lower Route Modifier as Active and reboots the UP with higher Route Modifier.

RCM Switchover during Route Modifier Wraparound

If an Active UP registration is received with lowest Route Modifier, the RCM waits for five minutes (UP self-reboot time):

- If the time period is exceeded, the Route Modifier is wrapped around.
- If within the time period, another Active UP registration is received with same Host ID (but with higher Route Modifier), then the reboot is performed on the UP with the higher Route Modifier. The Route Modifier wraparound procedure is then started for the lowest Route Modifier UP.

Limitations

UP switchover is not supported within five minutes of RCMs switchover. If there is any switchover within this time, it is considered as double-failure and Session Recovery is not possible.

Configuring Ops Center for Keepalived Pod

Use the following parameters to configure the Keepalived pod.

```
configure
   endpoint rcm-keepalived
exit
k8 smf profile rcm-keepalived-ep vrrp-config group group_name
   vrrp interface vrrp_interface
   vrrp routerId router_id
   ipv4-addresses address vip_address
   ipv4-addresses mask address_mask
```

```
ipv4-addresses broadcast broadcast_ipaddress
ipv4-addresses device device_interface
host priority priority
host hostid host_id
exit
```

NOTES:

- *vrrp_interface*: Specifies the VRRP tracking interface.
- *router_id*: Specifies the VRRP router ID.
- *vip_address*: Specifies the Keepalived VIP IP address which must be same in both RCM VMs.
- *address_mask*: Specifies the VIP IP address mask.
- *broadcast_ipaddress*: Specifies the VIP IP broadcast address.
- *device_interface*: Specifies the interface where VIP IP must be configured.
- *priority*: Specifies the VRRP host priority and must be different for both the RCM VMs.
- *host_id*: Specifies the VRRP host ID.

Setting IPTables Rules for Keepalived

Set the iptables rules for Keepalived using the below command in each VM.

sudo ufw allow in on *interface_name* **from** *peer_vm_address*

For example:

sudo ufw allow in on ens6.2299 from 192.168.20.247

NOTES:

- *peer_vm_address* must be configured to the same value specified in the keepalived.conf file.

RCM IPSec

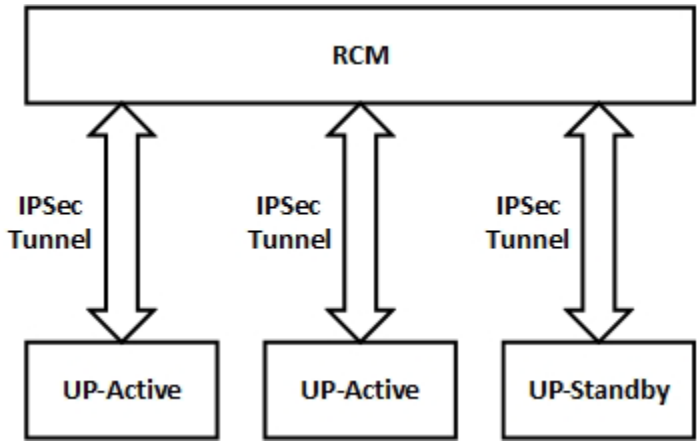
Feature Description

Security is very crucial aspect of any solution that meets vital customer requirements on security-related issues. RCM provides a secure computing environment and is part of a stable solution. RCM addresses both internal and external security aspects by protecting traffic between RCM and UP using IP Security (IPSec).

IPSec is a suite of protocols that interact with one another to provide secure private communications across IP networks. These protocols allow the system to establish and maintain secure tunnels with peer security gateways. IPSec provides confidentiality, data integrity, access control, and data source authentication to IP datagrams.

Architecture

The following diagram depicts the overall architecture of the RCM IPSec solution.



There is one IPSec (L3) tunnel between RCM and each UP. This node-level L3 tunnel between RCM and UP guarantees full protection of information exchanged between RCM and UP.

At a high-level, the RCM IPSec solution provides the following functionality:

- RCM supports multiple IPSec tunnels.
- RCM functions with or without IPSec.
- RCM supports OpsCenter CLI using which IPSec can be enabled and IPSec-related information can be configured.
- Show CLI commands display details of the IPSec tunnel and IPSec data exchanged.
- RCM generates SNMP traps when the tunnel is established, released, re-established, and so on.
- Supports configuration of strongSwan with a static routing table.
- RCM HA supports reestablishment of UP with IPSec tunnel during switchover scenario.

Supported Algorithms

The RCM IPSec supports the Internet Key Exchange version 2 (IKEv2) protocol. The following table provides information about the supported options.

Protocol	Type	Supported Options
Internet Key Exchange version 2 (IKEv2)	IKEv2 Encryption	AES-CBC-128 AES-CBC-256
	IKEv2 Pseudo-Random Function (PRF)	PRF-HMAC-SHA1 PRF-HMAC-SHA2-256
	IKEv2 Integrity	HMAC-SHA1-96 HMAC-SHA2-256-128 HMAC-SHA2-384-192 HMAC-SHA2-512-256
	IKEv2 Diffie-Hellman (DH) Group	Group 14 (2048-bit) Group 15 (3072-bit) Group 16 (4096-bit)

How it Works

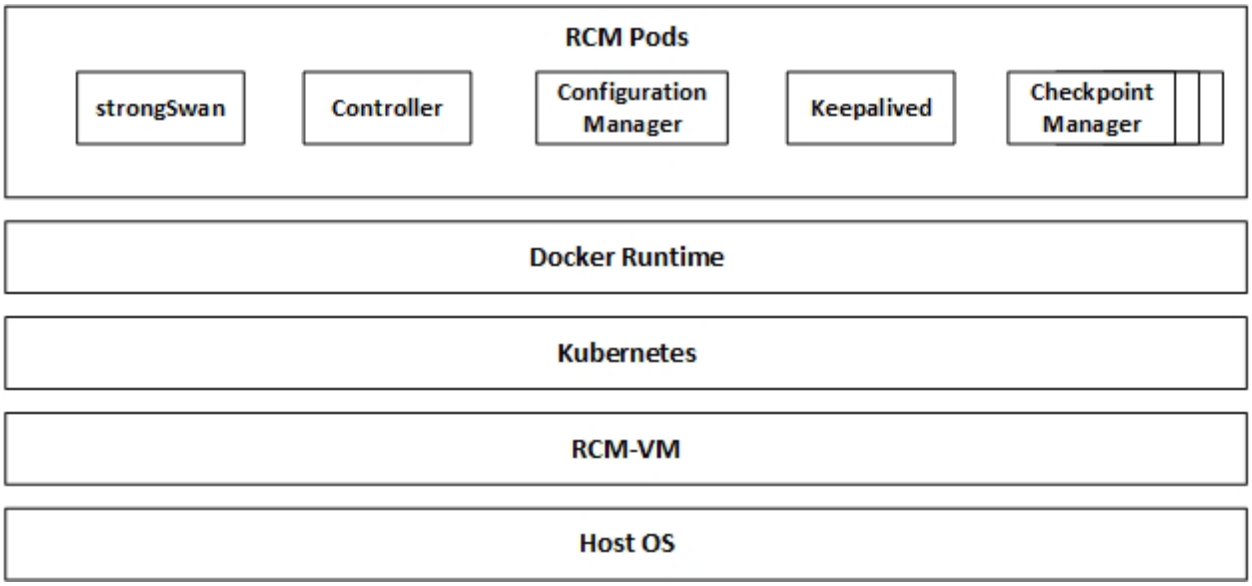
A single IPSec tunnel is established between RCM and UP. This tunnel carries checkpoint traffic, configuration files, VPNMGR traffic towards RCM, and so on. Since configuration from RCM uses SSH, there is no need to use IPSec for SSH traffic.

To support IPSec operation, the following pod is introduced:

- **strongSwan**: Establishes the IPSec tunnel (that is, the Control Plane of IPSec)
-

The strongSwan runs as a pod with Host mode networking and as a Deamon process. When you enable the IPSec using OpsCenter, then strongSwan pod gets created. IPSec tunnel initiation always happens from UP. The strongSwan handles the IPSec tunnel creation aspect. And, once the IPsec tunnel parameters are determined, it configures the host kernel for the IPSec data plane. The host kernel performs the encryption/decryption of IPSec traffic. When host kernel receives the encrypted packet, it decrypts the packet and punts it to the application listening on specific IPs. The strongSwan maintains the tunnel state between the RCM and the UP.

The following illustration depicts the system architecture of RCM VM.

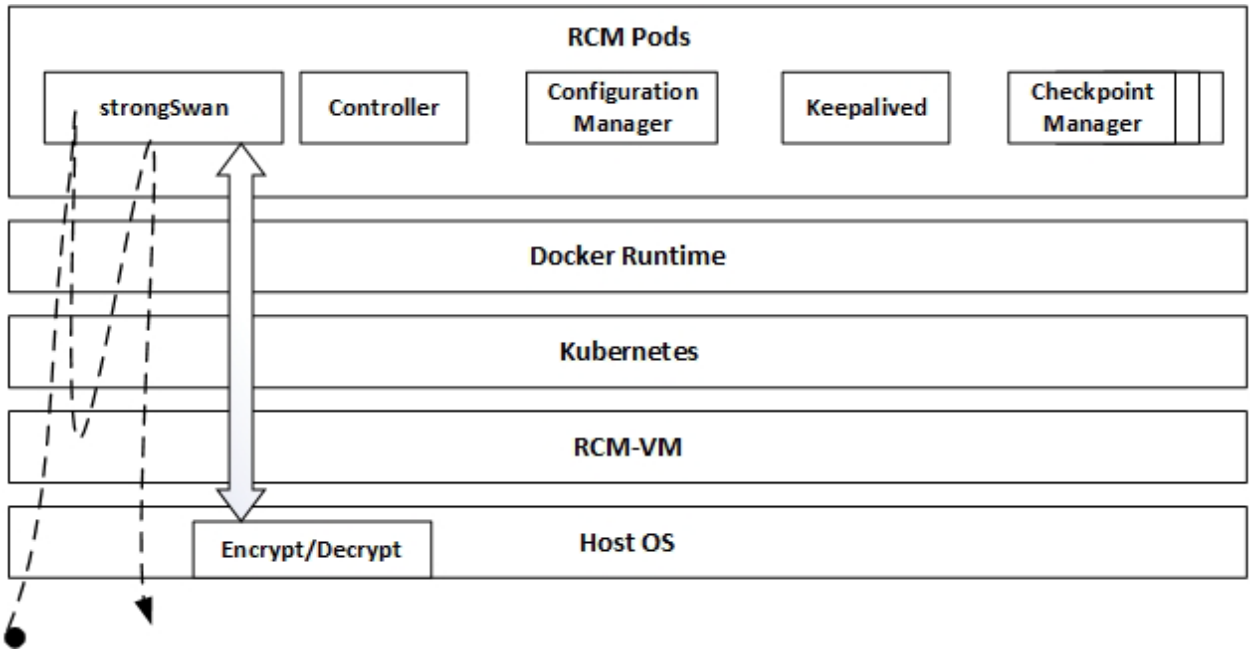


When RCM is deployed on bare metal, the RCM VM layer does not exist, and packet flow involves host kernel.

NOTE: Additional hardware for RCM is required as few cores must be dedicated for strongSwan.

RCM IPSec Tunnel Establishment

The following illustration depicts how IPSec tunnel is established between RCM and UP, and the packet flow.

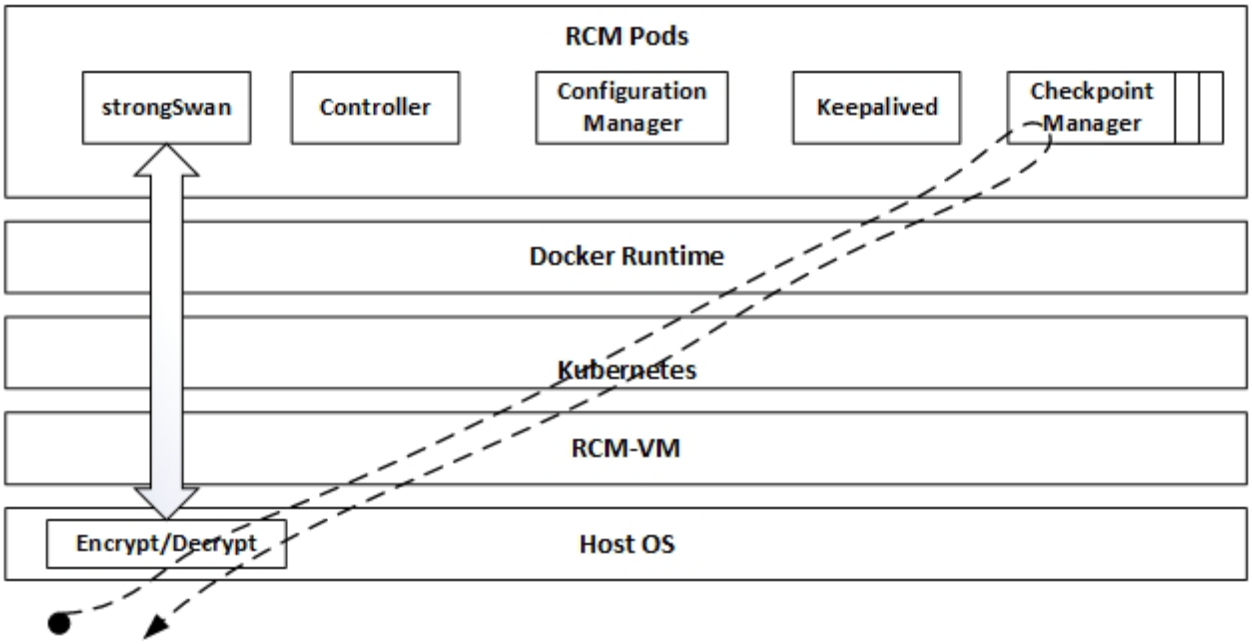


The following steps explain how IPSec tunnel is established between RCM and UP:

3. UP initiates IPSec tunnel establishment using IPSec control protocol such as IKEV1 or IKEV2. You may opt to choose only IKEV2.
4. Host kernel receives the packet.
5. The host kernel sends the packet to strongSwan.
6. The above steps are repeated between UP and RCM till IPSec tunnel establishment concludes.
7. After the tunnel is established, strongSwan configures the host kernel with IPSec data plane parameters.
8. Host kernel is now ready to encrypt/decrypt IPSec packet towards the UP, and strongSwan is ready to maintain the IPSec tunnels between host kernel and UP.
9. NOTE: Above sequence of operation is executed whenever UP establishes an IPSec tunnel.

Checkpoint Data Transfer

The following illustration depicts the checkpoint data transfer and packet flow.



The following steps explain the sequence of checkpoint data transfer and packet flow:

10. SessMgr of UP sends checkpoint over IPSec tunnel (note that all packet between RCM and UP uses the same tunnel, and there is one tunnel between UP and RCM).
11. Host kernel receives the packet and decrypts it.
12. The host kernel sends the packet to desired checkpoint manager (essentially, this happens through Kubernetes networking).
13. Checkpoint manager processes the packet and generates a response (for example, TCP ACK).
14. The response packet is sent to the host kernel.
15. Host kernel encrypts and egresses the packet.
16. The strongSwan maintains the tunnel state between RCM and UP and also keeps count of the tunnels idle-state and count of encrypted/decrypted packets.

RCM IPSec during Switchover

During RCM switchover, Standby RCM becomes Active. During this time, UPs try reconnecting with the new RCM. However, the reconnection fails as the new RCM does not have an IPSec tunnel. To resolve this reconnection failure, IPSec keepalive timer must be configured. IPSec keepalive timer expires as new RCM does not send keepalive messages. When this happens, UP triggers the establishment of new IPSec tunnel. After the IPSec tunnel is established, all UP connections toward new RCM becomes operational, and IPSec data is exchanged.

Limitations and Restrictions

The following are the known limitations/restrictions of RCM IPSec feature:

- Dynamic change of IPSec mode is not supported.

  Configuring strongSwan with static routing table is supported. Dynamic routing table update is not supported.

- By default, all IPSec ports are blocked for enhanced security. Use the following steps to open the port as part of RCM deployment:

```
echo udp:500 >> /usr/local/etc/ports_to_allow.txt
echo udp:4500 >> /usr/local/etc/ports_to_allow.txt
echo esp >> /usr/local/etc/ports_to_allow.txt
 /usr/local/bin/block_ports.sh -p /usr/local/etc/ports_to_allow.txt
```

Configuring RCM IPSec

This section provides information about the CLI commands available in support of the RCM IPSec feature.

Enabling IPSec in RCM

Use the following configuration to enable IPSec in RCM.

```
configure
   rcm-ipsec enable true
   exit
```

NOTE:

- After IPSec is enabled, mandatory parameters like **left-subnet** and **right-subnet** must be configured.

Disabling IPSec in RCM

Use the following configuration to enable IPSec in RCM.

```
configure
   rcm-ipsec enable false
```

```
   exit
```

Configuring IPSec Transform Set

The IPSec Transform Set Configuration mode is used to configure IPSec security parameters. Use the following CLI commands to configure the IPSec Transform Set Configuration mode.

```
configure

   rcm-ipsec ikev2-ikesa transform-set

   exit
```

Configuring IPSec Parameters

Use the following CLI commands to configure IPSec parameters in RCM.

```
configure

   rcm-ipsec ikev2-ikesa transform-set

      dh-group { 14 | 15 | 16 }

      encryption { aes-cbc-128 | aes-cbc-256 }

      hmac { sha1-96 | sha2-256-128 | sha2-384-192 | sha2-512-256 }

      left-subnet

      prf { sha1 | sha2-256 }

      psk aes_encrypted_string

      right-subnet

      exit

   end
```

NOTES:

- **`dh-group { 14 | 15 | 16 }`**: Specifies Diffie-Hellman group configuration. It configures the appropriate key exchange cryptographic strength and activates Perfect Forward Secrecy by applying a Diffie-Hellman group.
  - Group 14 provides 2048 bits of keying strength. This is the default option.
  - Group 15 provides 3072 bits of keying strength.
  - Group 16 provides 4096 bits of keying strength.
- **`encryption { aes-cbc-128 | aes-cbc-256 }`**: Specifies the encryption algorithm and encryption key length.

  - **aes-cbc-128**: An advanced Encryption Standard Cipher Block Chaining with a key length of 128 bits. This is the default setting for this command.
  - **aes-cbc-256**: An advanced Encryption Standard Cipher Block Chaining with a key length of 256 bits.
- **hmac { sha1-96 | sha2-256-128 | sha2-384-192 | sha2-512-256 }**: Specifies integrity algorithm using a Hash-based Message Authentication Code (HMAC).
  - **sha1-96**: Uses a 160-bit secret key and produces a 160-bit authenticator value. This is the default setting for this command.
  - **sha2-256-128**: Uses a 256-bit secret key and produces a 128-bit authenticator value.
  - **sha2-384-192**: Uses a 384-bit secret key and produces a 192-bit authenticator value.
  - **sha2-512-256**: Uses a 512-bit secret key and produces a 256-bit authenticator value.
- **left-subnet**: [Mandatory] Specifies the RCM external IP/VIP.
- **prf**: Specifies the Pseudo-Random Function (PRF) algorithm.
- **psk**: Specifies the Pre-Shared Key for peer authentication.
- **right-subnet**: [Mandatory] Specifies the RCM service IP and port for UP.

Configuring strongSwan Endpoint

Use the following CLI commands to configure the strongSwan endpoint to create the pod.

configure

endpoint rcm-strongswan vip-ip *ip_address*

end

Sample Configuration

The following is a sample configuration for your reference.

```
configure
rcm-ipsec enable true
rcm-ipsec ipsec transform-set
encryption aes-256-gcm-128
 hmac       sha1-96
 dh-group   16
exit
rcm-ipsec ikev2-ikesa timer
 ikelifetime        10000
 retransmit-tries   5
 retransmit-timeout 99
exit
```

```
rcm-ipsec ikev2-ikesa transform-set
 encryption aes-cbc-128
 hmac       sha2-256-128
 prf        sha1
 dh-group   14
 psk        test
 left-subnet 192.0.2.1 port 1111 prefix-length 32
 left-subnet 192.0.3.1 port 1111 prefix-length 32
 right-subnet 192.0.4.1 port 2222 prefix-length 24
 right-subnet 192.0.5.1 port 2222 prefix-length 24
exit
```

SNMP Traps

The following traps are generated by the strongSwan Manager pod:

- TunnelsDropped
- TunnelsAdded
- For the complete list of SNMP Traps available in RCM and configuration, see SNMP Traps under the Monitoring and Troubleshooting RCM section.

RCM and SMI Cluster Manager Integration

Feature Description

The Subscriber Microservices Infrastructure (SMI) Kubernetes (K8s) Cluster Manager allows you to deploy 5G Network Functions (NFs) which are developed based on SMI. SMI Cluster Manager (SMI CM) can also deploy UPF that is based on StarOS image.

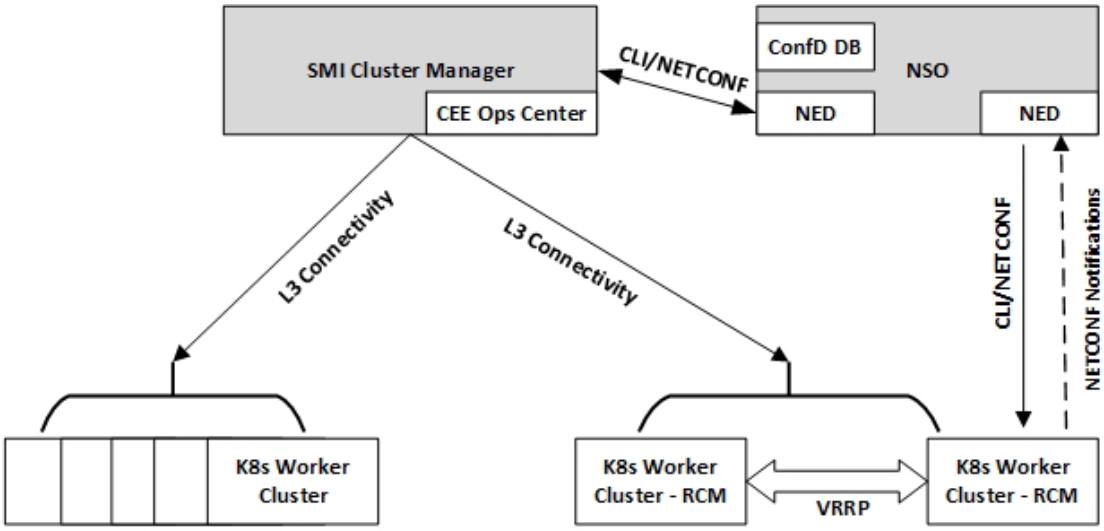
The Network Services Orchestrator (NSO) core function pack (CFP) prepares the configuration that is required to deploy all NFs including RCM, SMF, and UPF.

StarOS Network Element Drivers (NEDs) in the NSO is used to configure the UPF. The NED prepares the StarOS CLI from the configurations that are done through the Network Configuration Protocol (NETCONF) on NSO and pushes them to the UPFs. The RCM is configured using NSO through the NETCONF client on port 2022.

The K8s worker is a cluster of nodes managed by an instance of Kubernetes. However, for RCM, the K8s cluster is a single node.

How it Works

The SMI CM provisions the K8s clusters. Each K8s cluster deploys a particular NF. The RCM, a Cisco proprietary NF for UPFs that provides N:M redundancy, is allocated to one of the K8s worker clusters. The RCM must be a single-node K8s cluster. When RCM HA is used, the SMI CM brings up two RCM K8s clusters in separate nodes. These RCM clusters must be collocated in the same rack as the UPFs. The SMI CM brings up RCM pods like it does for other 5G NF components.



1. Once the SMI CM provisions the RCM pods in a K8s cluster, the rcm-ops-center starts running on that single-node RCM cluster.
2. NSO configures the Day-1 RCM configuration on each of the RCMs (Active and Standby).

   The Day-1 configuration consists:

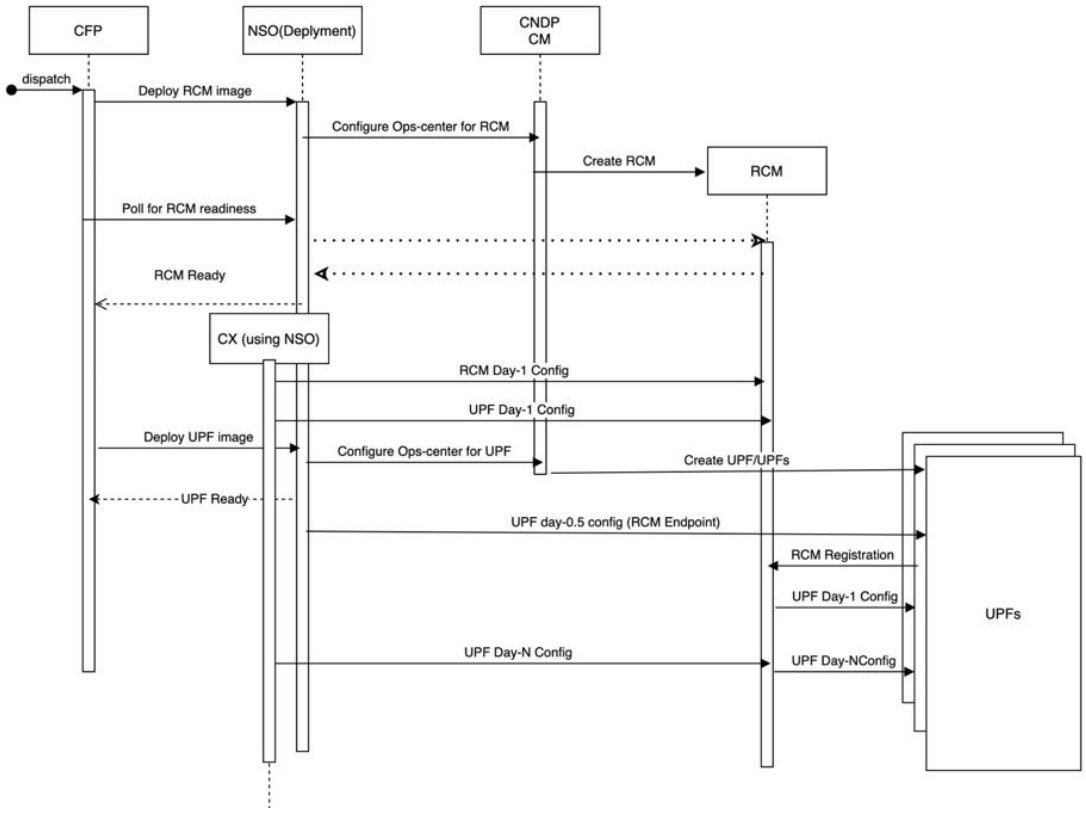
- Endpoint information: rcm-controller, rcm-bfdmgr, rcm-configmgr, replicas of rcm-checkppointmgr, rcm-keepalived and, optionally, rcm-snmp-trapper and rcm-ipsec nodes.
- IP addresses, BFD timers, and other operational parameters for logging levels, and so on.

3. When VIP is configured for rcm-keepalived pod on both RCM clusters, VRRP connection on both clusters is established. One of them is selected as master/primary and other as backup.
4. Once the required functional pods of RCM on each cluster is up and running, the NSO configures the required UPs Day-1 configuration on each RCM. For details about configuration, see the Configuring RCM and UPF for Redundancy section.
5. The SMI CM provisions the UPF on a VM. Once the UPF boots, it is configured with RCM context. Using the details on RCM endpoint IP in rcm-context configuration, the UPF establishes connection with the Active RCM.
6. The Day-1 UPF configuration is pushed by RCM/NSO based on the configuration mode.
7. The cee-ops-center in SMI CM monitors all the K8s clusters. It has pods for collecting various diagnostic information from each cluster, such as Prometheus, alert managers, log retrievers, and so on. The cee-ops-center on the SMI CM collects information from both the cluster of RCMs and displays collective information on its dashboard.

Call Flows

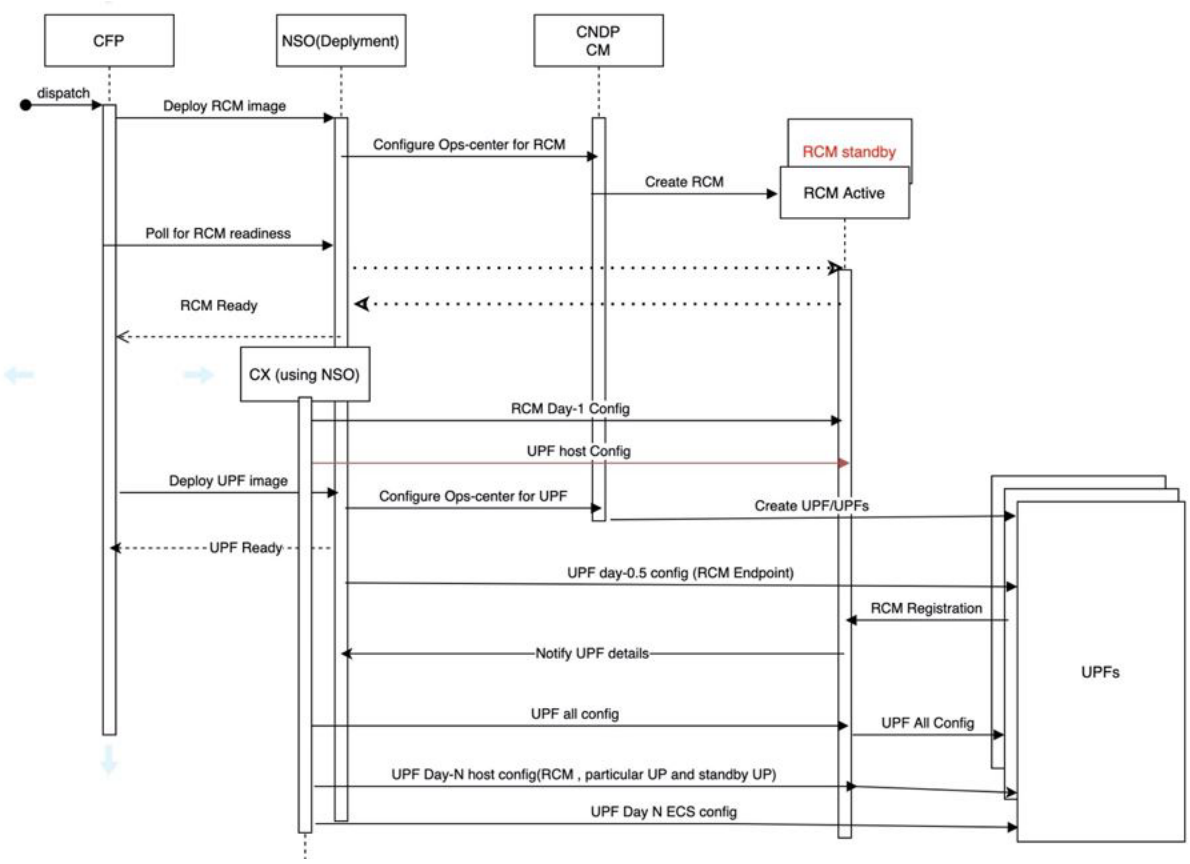
UPFs Configured by RCM

The following call flow illustrates bringing up of RCM using SMI CM and NSO core function pack (CFP) if the RCM configures the UPFs.



UPFs Configured by NSO

The following call flow illustrates bringing up of RCM when NSO configures the UPFs, and RCM needs only the host-specific configuration from NSO where it uses that configuration during switchover.



Provisioning RCM

All the RCM components can be deployed as pods using the helm charts. Each individual pod in RCM is built on SMI application infrastructure like other cloud native NFs. The RCM product is bundled as tar images.

SMI CM can only provision one RCM per single-node K8s cluster. The RCM BFDMgr runs in host networking mode. There is a possibility for port conflict if other running applications use the same port as RCM pods. Since RCM is specifically customized for StarOS UP/UPFs, we recommend isolating the RCM pods from other NFs.

RCM HA in SMI CM

The SMI CM deploys two RCMs as two separate single node K8s clusters. Both the RCMs communicate over VRRP to determine Master/Backup roles. The SMI CM is unaware of the redundancy that is achieved by the two RCM K8s clusters using the rcm-keepalived.

When rcm-keepalived pods come up after configuring with VRRP parameters, they communicate and determine the Master/Primary. The Master/Primary RCM serves all the UPFs to provide redundancy.

The health monitoring script in each RCM cluster runs periodically to check the status of the current cluster. If any issue is detected, the entire namespace must be redeployed. To trigger the RCM namespace reload, a

script is provisioned on the node that communicates with the rcm-keepalived. Redeploying only the RCM namespace avoids unnecessary reload of non-RCM NFs if they are deployed in the same cluster.

Configuring RCM and UPF for Redundancy

The NSO is leveraged for configuring various NFs, including RCM. The NSO configures RCM, required to support redundancy, using the NETCONF interface.

When configuration of UPF is done through RCM, the NSO provides the configuration required for UPFs in a format that is interpreted by the RCM.

The common configuration that must be configured by NSO for UPF, through RCM, is in the following syntax:

```
configure
   redundancy-group group_number
      common line_number "staros_cli_string"
      end
```

The host configuration that must be configured by NSO for UPF, through RCM, is in the following syntax:

```
configure
   svc-type { upinterface | sxsvc | upsvc | gtpusvc | cpgrp | li }
      redundancy-group group_number
         host line_number "staros_cli_string"
         end
```

Configuring UPF Credentials in SMI CM

For details about the configuring UPF credentials, see Configuring UPF Credentials. It is also possible to configure the default credentials for each redundancy group, wherein you need not configure the credentials for each UPF. If credentials are not explicitly configured for any UPF, the default credentials are used.

Monitoring and Troubleshooting

This section provides information about the functionality available to monitor and/or troubleshoot RCM in SMI CM.

Log Forwarding and Gather TAC

The SMI CM leverages journalD service—a Linux system service for collecting and storing log data—running on the node and uses the log-forwarder to forward the logs to the configured server on the CEE.

The Gather TAC functionality, developed to fetch the cores and logs, is reused by SMI CM to fetch cores and logs from RCM cluster like any other NFs.

For details, refer the *Log Forwarding* and *Gather TAC* sections in the *UCC SMI Common Execution Environment Configuration and Administration Guide*.

Metrics and Alerts

The metrics for each RCM pod is defined per Prometheus. The metrics are incremented/decremented using the APIs provided by the SMI App-infra.

The metrics from each RCM POD is exposed at port 8081:

```
prometheus.io/scrape: 'true'
prometheus.io/port: "8081"
```

The 8081 port is exposed in each pod's service.

```
- name: metrics
  port: 8081
```

NOTE:

- A label for a metric must be defined in *metrics/metric-labels.go*
- Add the defined label to the SMI App-infra using GetPrometheusLabelss "Add" method.
- Register the added metric using the "RegisterCounter_V1"/"RegisterGauge_V1"

method.

- All available APIs to be used on counters can be found in GIT repository.
- All the Alerts that are part of SMI App-infra of the RCM pods is available at cee-ops-center.
- For details about the counters/metrics that are added for RCM pods, see the Appendix G: RCM Metrics section.

Grafana

Grafana is an open-source data visualization tool used for displaying application metrics in interactive dashboards. A separate dashboard is created to monitor the metrics exposed by the RCM pods.

For details about Grafana, refer the *Grafana* section in the *UCC SMI Common Execution Environment Configuration and Administration Guide*.

Appendix A: Deployment Parameters

Important Note: This section provides sizing parameters and recommendations based on typical deployment setup, and its solely for your reference.

Typical Deployment

Typical deployment consists of:

- 10 Active UPs per RCM
- One or multiple redundancy group with total of 10 Active UPs across all redundancy groups
- Each UP with 10 SessMgrs
- Total UP capacity with 200k (approx.) sessions and per session size of 38 KB (approx.)
    - Approx. 7.2 GB for 200k sessions (Approx. 1.8 GB with compression)

Parameters

For a typical deployment, consider the following sizing parameters:

- Memory
- CPU
- Network
- Hard Disk

The required resources differ with:

- The number of UPs registered with RCM
- The number of SessMgrs in each UP
- The number of sessions in each SessMgr of an UP
- Call Events Per Second (CEPS)

Memory Sizing

For a Typical Deployment example given above:

- All checkpoints are stored in RAM
- Memory required for checkpoints for each UP is 2 GB (approx.)
- Application garbage collector takes time to free up old checkpoints
- Checkpoints are continuous (Added/Modified/Deleted)
- Based on CEPS
- Typically requires 4 GB for one UP with 200k sessions excluding operational overhead of minimum 10 GB.
- To support 10 UPs with 200k sessions each and 500 CEPS, recommended size of RAM is minimum of 50 GB.

CPU Sizing

The following parameters must be considered for CPU sizing:

- The number of Checkpoint Managers (ChkpointMgrs)/Redundancy Managers (RedMgrs) should be equal to the number of SessMgrs in each UP.
- All UP has same number of SessMgrs.
- During switchover, all the ChkpointMgrs work parallelly to flush checkpoints from RCM to new Active UP:
  - To achieve true parallelism, it is recommended to have as many cores as the number of ChkpointMgrs.
  - Additionally, two cores for operations involving other pods.

  Note: You can try with lesser number of cores if the total number of sessions are less.

Network Sizing

The following parameters must be considered for network sizing:

- Requires two interfaces:
  - RCM service interface where checkpoints are transferred
  - Management interface where SSH and SFTP of configurations can be performed

    You can use RCM interface and configure in RCM context of UP. However, its recommended to keep them separate.
- Additionally, operational interface to login to the Ops Center and for NETCONF communication to configure the Ops Center. Operations interface is also used to collect logs, diagnose, alert, and so on. This interface can be same as Management interface.
- For RCM High Availability (HA), an additional L2-level network for VRRP is required.
- Speed:
  - Minimum of 25Gbps link for RCM interface for checkpoints
    - (Low latency as BFD runs on this interface.)
  - A basic 1Gbps link for management interface

Hard Disk Sizing

The following parameters must be considered for hard disk sizing:

- Disk space required for logging, docker images, and so on.
- It is recommended to have at least 40 GB of disk space.
- If resource crunch occurs, the pods are evicted.
- Some persistent storage to preserve data across reboots.

The following table is a typical recommendation.

Number of UPs	SessMgrs/UP	Total Sessions/RCM	Recommended Cores	Recommended Memory
5	8	1000K	10	32 GB
6	8	1200K	10	36 GB
8	10	1600K	12	45 GB
10	12	2000K	14	55 GB
10	16	2000K	18	55 GB

Appendix B: Sample Common and Host-specific Configurations

Important Note: This section provides sample Common and Host-specific configurations based on typical deployment setup, and its solely for your reference.

Common Configuration

The following is a sample Common configuration. All ACS-related configurations are part of Common configuration.

```
redundancy-group 1
common 0 "sleep 5 "
common 1 "config "
common 2 "active-charging service ACS "
common 3 "idle-timeout udp 60 "
common 4 "statistics-collection ruledef all "
common 5 "exit"
```

NOTE: There should be an "exit" in the Common configuration for each section, such as rulebase, ruledef, host pool, port map, and so on.

Host-specific Configuration

The following is a sample Host-specific configuration.

```
svc-type upinterface
   redundancy-group 1
   host Active1
   host 1 config
   host 2 " context EPC2"
   host 3 " interface S5_SGW_PGW_loopback_up1 loopback"
   host 4 " ip address 192.168.170.77 255.255.255.255"
   host 5 " #exit"
   host 6 " interface pgw-ingress-ipv6-loopback_up1 loopback"
   host 7 " ipv6 address bbbb:aaaa::14/128"
   host 8 " #exit"
svc-type sxsvc
   redundancy-group 1
   host Active1
   host 1 config
```

```
   host 2 " context EPC2"
   host 42 " sx-service sx_up1"
   host 43 " instance-type userplane"
   host 44 " bind ipv4-address 192.168.170.77"
   host 45 " exit"
   host 46 " end"
exit
svc-type upsvc
   redundancy-group 1
   host Active1
   host 1 config
   host 2 " context EPC2"
   host 46 " user-plane-service up_up1"
   host 47 " associate gtpu-service pgw-gtpu_up1 pgw-ingress"
   host 48 " associate gtpu-service sgw-ingress-gtpu_up1 sgw-ingress"
   host 49 " associate gtpu-service sgw-engress-gtpu_up1 sgw-egress"
   host 50 " associate gtpu-service saegw-sxu_up1 cp-tunnel"
   host 51 " associate sx-service sx_up1"
   host 52 " associate fast-path service"
   host 53 " associate control-plane-group g1"
   host 54 " exit"
   host 55 " #exit"
   host 56 end
exit
svc-type gtpusvc
   redundancy-group 1
   host Active1
   host 1 config
   host 2 " context EPC2"
   host 30 " gtpu-service pgw-gtpu_up1"
   host 31 " bind ipv4-address 192.168.170.14"
   host 32 " exit"
   host 33 " gtpu-service saegw-sxu_up1"
   host 34 " bind ipv4-address 192.168.170.31"
   host 35 " exit"
   host 36 " gtpu-service sgw-engress-gtpu_up1"
```

```
   host 37 " bind ipv4-address 192.168.170.51"
   host 38 " exit"
   host 39 " gtpu-service sgw-ingress-gtpu_up1"
   host 40 " bind ipv4-address 101.101.102.48"
   host 41 " exit"
   host 42 " end"
exit
svc-type cpgrp
   redundancy-group 1
   host Active1
   host 0 config
   host 1 "control-plane-group g1"
   host 2 "peer-node-id ipv4-address 192.168.196.10"
   host 3 exit
   host 4 end
exit
```

NOTE:

1. If you have 10 active hosts, then you should have 10 active Host-specific configuration present under these services.
2. The Lawful Intercept (LI) configuration must be under “svc-type cpgrp”. There is no specific service for LI.

Appendix C: N:M Configuration Separation Table

CP	RCM	UP
1. URR List must be explicitly configured under Active Charging Service (ACS). 2. The **`sx-pfd-push`** and **`sx-reassociation`** CLI commands under UP group must be disabled.	Common configuration: • ACS • APN • GTPP Group (for URR generation) • AAA Group (for URR generation) Host Configuration: • Service loopback interfaces • Sx Service • GTPU Service • UP Service • CP Group • Lawful Intercept (LI): LI configuration must be under CP group configuration.	Day-0 configuration [All physical interfaces, contexts (including RCM context), routing configurations (including BGP), bulkstats, SNMP, Syslog, and so on.] Day-1 (Common and Host) configuration is pushed from RCM.

Appendix D: MIBs

The following is the information from *CISCO-RCM-MIB.my* file.

```
-- ****************************************************************
-- CISCO-RCM-MIB.my
-- Copyright (c) 2020 by cisco Systems Inc.
-- All rights reserved.
-- ****************************************************************

CISCO-RCM-MIB DEFINITIONS ::= BEGIN

IMPORTS
    MODULE-IDENTITY,
    OBJECT-TYPE,
    NOTIFICATION-TYPE
        FROM SNMPv2-SMI
    MODULE-COMPLIANCE,
    NOTIFICATION-GROUP,
    OBJECT-GROUP
        FROM SNMPv2-CONF
    TEXTUAL-CONVENTION,
    DateAndTime
        FROM SNMPv2-TC
    ciscoMgmt
        FROM CISCO-SMI;


ciscoRcmMIB MODULE-IDENTITY
    LAST-UPDATED    "202008120000Z"
    ORGANIZATION    "Cisco Systems, Inc."
    CONTACT-INFO
            "Cisco Systems

            Customer Service
```

```
            Postal: 170 W Tasman Drive
            San Jose, CA  95134
            USA

            Tel: +1 800 553-NETS"
    DESCRIPTION
        "The MIB module for the Cisco Redundancy Configuration Manager (RCM)
platform.

        This MIB only handles notifications from the RCM."
    REVISION        "202008120000Z"
    DESCRIPTION
        "Added rcmFaultClusterName, rcmFaultNamespace, rcmFaultHostname
         and rcmFaultInstance fields to identify the faults."
    REVISION        "201809190000Z"
    DESCRIPTION
        "Initial version of this MIB module."
    ::= { ciscoMgmt 1000}

-- Textual Conventions definition will be defined before this line

ciscoRcmMIBNotifs  OBJECT IDENTIFIER
    ::= { ciscoRcmMIB 0 }

ciscoRcmMIBFaults  OBJECT IDENTIFIER
    ::= { ciscoRcmMIB 1 }

ciscoRcmMIBConform  OBJECT IDENTIFIER
    ::= { ciscoRcmMIB 2 }

rcmFaultId OBJECT-TYPE
    SYNTAX          OCTET STRING (SIZE  (1..64))
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "Uniquely identify the fault within a monitored entity."
```

```
    ::= { ciscoRcmMIBFaults 1 }

rcmFaultSource OBJECT-TYPE
    SYNTAX          OCTET STRING (SIZE  (1..128))
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "Uniquely identify the monitored entity
         It can be a hostname or IP Address or
         human readable identity."
    ::= { ciscoRcmMIBFaults 2 }


rcmFaultSeverity OBJECT-TYPE
    SYNTAX          OCTET STRING (SIZE  (1..16))
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "Indicates the level of urgency for operator attention
         Refer 3GPP TS32.111-5 v9.0.0 section 4.3."
    ::= { ciscoRcmMIBFaults 3 }

rcmFaultTime OBJECT-TYPE
    SYNTAX          DateAndTime
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "The date and time when the fault is detected."
    ::= { ciscoRcmMIBFaults 4 }

rcmFaultType OBJECT-TYPE
    SYNTAX INTEGER {
           indeterminate(0),
           host-level(1),
           rcm-internal(2),
           userplane(3),
```

```
            pod-business-logic(4)
          }
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "Indicates the type of fault"
    ::= { ciscoRcmMIBFaults 5 }

rcmFaultAdditionalInfo OBJECT-TYPE
    SYNTAX          OCTET STRING (SIZE  (1..2048))
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "Additional Information about the fault."
    ::= { ciscoRcmMIBFaults 6 }

rcmFaultClusterName OBJECT-TYPE
    SYNTAX          OCTET STRING (SIZE  (1..128))
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "The cluster name associated to the fault."
    ::= { ciscoRcmMIBFaults 7 }
rcmFaultNamespace OBJECT-TYPE
    SYNTAX          OCTET STRING (SIZE  (1..128))
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "Identifies the namespace associated to
        the fault.  This field is not always available for
        every fault."
    ::= { ciscoRcmMIBFaults 8 }

rcmFaultHostname OBJECT-TYPE
    SYNTAX          OCTET STRING (SIZE  (1..128))
    MAX-ACCESS      not-accessible
```

```
    STATUS          current
    DESCRIPTION
        "Identifies the hostname or ip address associated
        with the fault.  This field is not always available
        for every fault."
    ::= { ciscoRcmMIBFaults 9 }
rcmFaultInstance OBJECT-TYPE
    SYNTAX          OCTET STRING (SIZE  (1..128))
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "Identifies the instance associated to
        the fault.  The instance is set by the alert rule
        creator and may not reference a host but could reference
        a process or KPI that is associated to the fault.  This
        field is not always available for every fault"
    ::= { ciscoRcmMIBFaults 10 }

rcmVnfAlias OBJECT-TYPE
    SYNTAX          OCTET STRING (SIZE  (1..128))
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "Alias for the monitored entity"
    ::= { ciscoRcmMIBFaults 11 }

-- Default Notification Type

rcmFaultActiveNotif NOTIFICATION-TYPE
    OBJECTS         {
                        rcmFaultId,
                        rcmFaultSource,
                        rcmFaultSeverity,
                        rcmFaultTime,
                        rcmFaultType,
                        rcmFaultAdditionalInfo,
```

```
                        rcmFaultClusterName,
                        rcmFaultNamespace,
                        rcmFaultHostname,
                        rcmFaultInstance,
                        rcmVnfAlias
                    }
    STATUS          current
    DESCRIPTION
        "This notification is generated by RCM
         whenever a fault gets triggered."
   ::= { ciscoRcmMIBNotifs 1 }

rcmFaultClearNotif NOTIFICATION-TYPE
    OBJECTS         {
                        rcmFaultId,
                        rcmFaultSource,
                        rcmFaultSeverity,
                        rcmFaultTime,
                        rcmFaultType,
                        rcmFaultAdditionalInfo,
                        rcmFaultClusterName,
                        rcmFaultNamespace,
                        rcmFaultHostname,
                        rcmFaultInstance,
                        rcmVnfAlias
                    }
    STATUS          current
    DESCRIPTION
        "This notification is generated by RCM
         whenever a fault gets cleared."
   ::= { ciscoRcmMIBNotifs 2 }

rcmEventNotif NOTIFICATION-TYPE
    OBJECTS         {
                        rcmFaultId,
                        rcmFaultSource,
```

```
                        rcmFaultSeverity,
                        rcmFaultTime,
                        rcmFaultType,
                        rcmFaultAdditionalInfo,
                        rcmFaultClusterName,
                        rcmFaultNamespace,
                        rcmFaultHostname,
                        rcmFaultInstance,
                        rcmVnfAlias
                    }
    STATUS          current
    DESCRIPTION
        "This notification is generated by RCM
         to notify a RCM event."
   ::= { ciscoRcmMIBNotifs 3 }

ciscoRcmMIBCompliances  OBJECT IDENTIFIER
    ::= { ciscoRcmMIBConform 1 }

ciscoRcmMIBGroups  OBJECT IDENTIFIER
    ::= { ciscoRcmMIBConform 2 }

rcmMIBCompliance MODULE-COMPLIANCE
    STATUS          current
    DESCRIPTION
        "The compliance statement for entities that support
        the Cisco RCM Managed Objects"
    MODULE          -- this module
    MANDATORY-GROUPS {
                        rcmMIBFaultGroup,
                        rcmMIBNotificationGroup
                    }
    ::= { ciscoRcmMIBCompliances 1 }

-- Units of Conformance
```

```
rcmMIBFaultGroup OBJECT-GROUP
    OBJECTS         {
                        rcmFaultId,
                        rcmFaultSource,
                        rcmFaultSeverity,
                        rcmFaultTime,
                        rcmFaultType,
                        rcmFaultAdditionalInfo,
                        rcmFaultClusterName,
                        rcmFaultNamespace,
                        rcmFaultHostname,
                        rcmFaultInstance,
                        rcmVnfAlias
                    }
    STATUS          current
    DESCRIPTION
        "The set of RCM Fault groups defined by this MIB"
    ::= { ciscoRcmMIBGroups 1 }

rcmMIBNotificationGroup NOTIFICATION-GROUP
   NOTIFICATIONS    { rcmFaultActiveNotif,
                      rcmFaultClearNotif }
    STATUS          current
    DESCRIPTION
        "The set of RCM notifications defined by this MIB"
    ::= { ciscoRcmMIBGroups 2 }

END
```

Appendix E: P2P/ADC Plugin Configuration and Update Procedure

This section provides information about P2P/ADC Plugin configuration and update procedure.

The following steps are required in all the UPs.

3. Copy the patch to */flash* of all the UPs.
4. To patch the plugin, execute the following CLI command:
   patch plugin p2p /flash/libp2p-*<plugin_filename>*
5. To install the plugin, execute the following CLI command:
   install plugin p2p patch_libp2p-*<plugin_filename>*
6. Configure the P2P module and upgrade it through RCM for all the UPs (Active and Standby in one step):

```
configure
   plugin plugin_name
      module priority number version plugin_version
      end
update module plugin_name
```

To roll back the P2P/ADC plugin in all UPs, execute the following CLI command in Exec mode:

```
rollback module plugin_name
```

NOTES:

- You must load, patch, and install the plugin on all the UPs individually.
- Configurations in Step 4 is loaded in the RCM configuration file using the Modify script. For details, see Modifying Host Configuration section.
- Ensure to save the plugin at CP and UP.

Appendix F: RCM Configuration Generation Utility

This section provides sample Host configurations that is based on typical deployment setup, explanations of the configurations, usage, and script-help.

Important Note: The information captured in this section is solely for your reference.

Host Configurations

```
host Active1 <<<----Name of the host
svc-type upinterface <<<<----For interface-specific configuration, we need to use this svc-type
config
context EPC2
interface loop1_up1 loopback
ip address 192.0.2.1 255.255.255.0
#exit
interface loop2_up1 loopback
ip address 192.0.2.2 255.255.255.0
#exit
interface loop3_up1 loopback
ip address 192.0.2.3 255.255.255.0
#exit
interface loop4_up1 loopback
ip address 192.0.2.4 255.255.255.0
#exit
interface loop5_up1 loopback
ip address 192.0.2.5 255.255.255.0
#exit
#exit
end
svc-type sxsvc <<<<----For Sx service-specific configuration, we need to use this svc-type
config
context EPC2
sx-service sx_up1
instance-type userplane
bind ipv4-address 192.0.2.5
exit
```

```
#exit
end
svc-type upsvc <<<<----For UP service-specific configuration, we need to use this svc-type
config
context EPC2
user-plane-service up_up1
associate gtpu-service pgw-gtpu_up1 pgw-ingress
associate gtpu-service sgw-ingress-gtpu_up1 sgw-ingress
associate gtpu-service sgw-engress-gtpu_up1 sgw-egress
associate gtpu-service saegw-sxu_up1 cp-tunnel
associate sx-service sx_up1
associate fast-path service
associate control-plane-group g1
exit
#exit
#exit
end
svc-type gtpusvc <<<<----For GTPU service-specific configuration, we need to use this svc-type
config
context EPC2
gtpu-service pgw-gtpu_up1
bind ipv4-address 192.0.2.2
exit
gtpu-service saegw-sxu_up1
bind ipv4-address 192.0.2.3
exit
gtpu-service sgw-engress-gtpu_up1
bind ipv4-address 192.0.2.4
exit
gtpu-service sgw-ingress-gtpu_up1
bind ipv4-address 192.0.2.1
#exit
#exit
end
svc-type cpgrp <<<<----For CP group-specific configuration, we need to use this svc-type
```

```
config
control-plane-group g1
peer-node-id ipv4-address 192.0.3.1
exit
end
exit
```

Script Help-string

~/Script/script_1302_02$ ./apply_config.sh

Usage: **./apply_config.sh -[g m i h n G] -f filename**

- -g : redundancy-group name
- -m : Append the starOS common/Host config to ops-center config
- -i : Input Common config file where starOS cli is present.
- -h : Input Host config file where starOS cli is present.
- -n : Input Namespace where ops-center is running.
- -G : Input Generation 4/5
    - For RCM VM version, input the value as 4. For RCM Cloud Native version, input the value as 5.

Applying both Common and Host-specific Configuration

```
~/Script/script_1302_02$ ./apply_config.sh -g 1 -G 5 -n smf -i
input/<path_name>/common_master.cli -h
<path_name1>/<path_name2>/host_master.cli
Validating....
Adding Commit Flag
Applying Clean Common Config
admin@10.0.0.1's password:
Commit complete.
Applying Clean Host Config
admin@10.0.0.1's password:
Commit complete.
```

Applying Common Configuration

```
~/Script/script_1302_02$ ./apply_config.sh -g 1 -G 5 -n smf -i
<path_name1>/<path_name2>/common_master.cli
Validating....
Adding Commit Flag
```

```
Applying Clean Common Config

admin@10.0.0.1's password:

% No modifications to commit.
```

Applying Host-specific Configuration

```
~/Script/script_1302_02$ ./apply_config.sh -g 1 -G 5 -n smf -h
<path_name1>/<path_name2>/host_master.cli

Validating....

Applying Clean Host Config

admin@10.0.0.1's password:

Commit complete.
```

Modifying Common Configuration

```
~/Script/script_1302_02$ ./apply_config.sh -g 1 -G 5 -n smf -m -i
<path_name1>/<path_name2>/common_

common_config_mod_2.cli common_config_mod.cli common_master.cli
common_master_mod_2.cli common_master_mod.cli

~/Script/script_1302_02$ ./apply_config.sh -g 1 -G 5 -n smf -m -i
<path_name1>/<path_name2>/common_config_mod.cli

Validating....

Applying Modified Common Config

admin@10.0.0.1's password:

Commit complete.

~/Script/script_1302_02$ ./apply_config.sh -g 1 -G 5 -n smf -m -i
<path_name1>/<path_name2>/common_config_mod_2.cli

Validating....

Applying Modified Common Config

admin@10.0.0.1's password:

Commit complete.
```

Modifying Host Configuration

```
~/Script/script_1302_02$ ./apply_config.sh -g 1 -G 5 -n smf -m -h
<path_name1>/<path_name2>/host_master_mod.cli

Validating....

preparing mod host config

Applying Modified Host Config

admin@10.0.0.1's password:

Commit complete.
```

```
~/Script/script_1302_02$ ./apply_config.sh -g 1 -G 5 -n smf -m -h
<path_name1>/<path_name2>/host_master_mod_2.cli

Validating....

preparing mod host config

Applying Modified Host Config

admin@10.0.0.1's password:

Commit complete.
```

Appendix G: RCM Metrics

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: smf-metrics-{{ .Release.Name }}
  labels:
    documentation-type: metrics
    documentation-group: smf
    documentation-group-version: {{.Chart.Version}}
data:
  documentation.yaml: |-
   metrics:
      entries:
         -  category: RCM Bfdmgr UPF Event Stats
            metric: bfdmgr_upf_event_stats
            description: "Number of events received for each UPF"
            sampleQuery: "bfdmgr_upf_event_stats{service_name=\"bfdmgr\"}"
            labels:
            -  label: endpoint
               description: "endpoint IP of UPF"
               example: "1.1.1.1"
            -  label: groupid
               description: "GroupId of the UPF"
               example: "1"
         -  category: RCM Bfdmgr Controller Event Stats
            metric: bfdmgr_controller_event_stats
            description: "Number of events sent to rcm-controller"
            sampleQuery:
"bfdmgr_controller_event_stats{service_name=\"bfdmgr\"}"
            labels:
            -  label: endpoint
               description: "endpoint IP of UPF"
               example: "1.1.1.1"
            -  label: groupid
```

```
            description: "GroupId of the UPF"
            example: "1"
        - category: RCM checkpointmgr Current call count
          metric: chkptmgr_total_call_count
          description: "Current Number of calls checkpointed to RCM
checkpointmgr"
          sampleQuery: "chkptmgr_total_call_count{service_name=\"rcm-
checkpointmgr\"}"
          labels:
          - label: sessmgr_no
            description: "Sessmgr Number of UP"
            example: "1"
          - label: up_address
            description: "IP address of UP"
            example: "1.1.1.1"
        - category: RCM checkpointmgr Current active VOLTE call count
          metric: chkptmgr_volte_active_call_count
          description: "Current Number of volte active calls checkpointed
to RCM checkpointmgr"
          sampleQuery:
"chkptmgr_volte_active_call_count{service_name=\"rcm-checkpointmgr\"}"
          labels:
          - label: sessmgr_no
            description: "Sessmgr Number of UP"
            example: "1"
          - label: up_address
            description: "IP address of UP"
            example: "1.1.1.1"
        - category: RCM checkpointmgr Current non-prio active call count
          metric: chkptmgr_non_prio_active_call_count
          description: "Current Number of non-prio active calls
checkpointed to RCM checkpointmgr"
          sampleQuery:
"chkptmgr_non_prio_active_call_count{service_name=\"rcm-checkpointmgr\"}"
          labels:
          - label: sessmgr_no
            description: "Sessmgr Number of UP"
            example: "1"
```

```
            - label: up_address
              description: "IP address of UP"
              example: "1.1.1.1"
         - category: RCM checkpointmgr Current non-active VOLTE call count
           metric: chkptmgr_volte_non_active_call_count
           description: "Current Number of non-actove VOLTE calls
checkpointed to RCM checkpointmgr"
           sampleQuery:
"chkptmgr_volte_non_active_call_count{service_name=\"rcm-checkpointmgr\"}"
           labels:
           - label: sessmgr_no
             description: "Sessmgr Number of UP"
             example: "1"
           - label: up_address
             description: "IP address of UP"
             example: "1.1.1.1"
         - category: RCM checkpointmgr Current non-prio non-active call
count
           metric: chkptmgr_non_prio_non_active_call_count
           description: "Current Number of non-prio non-active calls
checkpointed to RCM checkpointmgr"
           sampleQuery:
"chkptmgr_non_prio_non_active_call_count{service_name=\"rcm-checkpointmgr\"}"
           labels:
           - label: sessmgr_no
             description: "Sessmgr Number of UP"
             example: "1"
           - label: up_address
             description: "IP address of UP"
             example: "1.1.1.1"
         - category: RCM checkpointmgr Current Emergency call count
           metric: chkptmgr_emergency_call_count
           description: "Current Number of Emergency calls checkpointed to
RCM checkpointmgr"
           sampleQuery: "chkptmgr_emergency_call_count{service_name=\"rcm-
checkpointmgr\"}"
           labels:
           - label: sessmgr_no
```

```
            description: "Sessmgr Number of UP"
            example: "1"
          - label: up_address
            description: "IP address of UP"
            example: "1.1.1.1"
        - category: RCM checkpointmgr UP connection count
          metric: chkptmgr_up_connection_count
          description: "Count related to UP connections to RCM
checkpointmgr"
          sampleQuery: "chkptmgr_up_connection_count{service_name=\"rcm-
checkpointmgr\"}"
          labels:
          - label: sessmgr_no
            description: "Sessmgr Number of UP"
            example: "1"
          - label: up_address
            description: "IP address of UP"
            example: "1.1.1.1"
          - label: up_mode
            description: "Mode in which UP is running"
            example: "active/standby"
          - label: conn_state
            description: "Connection state of UP"
            example: "connected/disconnected"
        - category: RCM checkpointmgr UP Flush completed count
          metric: chkptmgr_up_flush_completed_count
          description: "Count related to UP flush completed in
checkpointmgr"
          sampleQuery:
"chkptmgr_up_flush_completed_count{service_name=\"rcm-checkpointmgr\"}"
          labels:
          - label: sessmgr_no
            description: "Sessmgr Number of UP"
            example: "1"
          - label: up_address
            description: "IP address of UP"
            example: "1.1.1.1"
```

```
        - category: RCM checkpointmgr UP Audit count
          metric: chkptmgr_up_audit_count
          description: "Count related to UP Audits in RCM checkpointmgr"
          sampleQuery: "chkptmgr_up_audit_count{service_name=\"rcm-
checkpointmgr\"}"
          labels:
          - label: sessmgr_no
            description: "Sessmgr Number of UP"
            example: "1"
          - label: up_address
            description: "IP address of UP"
            example: "1.1.1.1"
          - label: state
            description: "Current state"
            example: "start/end"
        - category: RCM configmgr UP count
          metric: configmgr_upf_count
          description: "Current count of UPs in a particular host group in
rcm configmgr"
          sampleQuery: "configmgr_upf_count{service_name=\"rcm-
configmgr\"}"
          labels:
          - label: host_grp_name
            description: "UP host group name"
            example: "any string"
        - category: RCM configmgr switchover abort count
          metric: configmgr_swover_abort_count
          description: "Count of switchover aborts in a particular host
group in rcm configmgr"
          sampleQuery: "configmgr_swover_abort_count{service_name=\"rcm-
configmgr\"}"
          labels:
          - label: host_grp_name
            description: "UP host group name"
            example: "any string"
        - category: RCM configmgr switchover failure count
          metric: configmgr_swover_failure_count
```

```
            description: "Count of switchover failures in a particular host
group in rcm configmgr"
            sampleQuery: "configmgr_swover_failure_count{service_name=\"rcm-
configmgr\"}"
            labels:
            - label: host_grp_name
              description: "UP host group name"
              example: "any string"
         - category: RCM configmgr config push failure count
            metric: configmgr_cfg_push_failure_count
            description: "Count of config push failure in a particular host
group in rcm configmgr"
            sampleQuery:
"configmgr_cfg_push_failure_count{service_name=\"rcm-configmgr\"}"
            labels:
            - label: host_grp_name
              description: "UP host group name"
              example: "any string"
         - category: RCM configmgr config push complete count
            metric: configmgr_cfg_push_complete_count
            description: "Count of config push complete in a particular group
in rcm configmgr"
            sampleQuery:
"configmgr_cfg_push_complete_count{service_name=\"rcm-configmgr\"}"
            labels:
            - label: host_grp_name
              description: "UP host group name"
              example: "any string"
         - category: RCM controller switchover failure count
            metric: controller_switchover_failure_stats
            description: "Count of failed switchover in rcm controller"
            sampleQuery:
"controller_switchover_failure_stats{service_name=\"rcm-controller\"}"
            labels:
            - label: failure_reason
              description: "Reason string for failure"
              example: "Unknown Destination Endpoint <> , Pre-switchover
Failed, Notification of switchover Trigger to standby UP encountered error
etc"
```

```
        - category: RCM controller switchover count
          metric: controller_switchover_stats
          description: "Count of total switchover in rcm controller"
          sampleQuery: "controller_switchover_stats{service_name=\"rcm-
controller\"}"
          labels:
          - label: swover_reason
            description: "Reason string for switchover"
            example: "BFD Failure, Planned Switchover etc"
          - label: state
            description: "State of switchover"
            example: "started or completed"
        - category: RCM controller last switchover duration
          metric: controller_last_switchover_seconds_total
          description: "Duration of last switchover completion in rcm
controller"
          sampleQuery:
"controller_last_switchover_seconds_total{service_name=\"rcm-controller\"}"
        - category: RCM controller UPF registered count
          metric: controller_upf_registered_stats
          description: "Count of total UPF registered in rcm controller"
          sampleQuery: "controller_upf_registered_stats{service_name=\"rcm-
controller\"}"
          labels:
          - label: grpId
            description: "Group ID of UPF"
            example: "any valid string"
          - label: endpoint
            description: "Endpoint address of UPF"
            example: "any ip address string"
        - category: RCM controller mass UPF failure count
          metric: controller_mass_upf_failure_stats
          description: "Count of mass UPF failures in rcm controller"
          sampleQuery:
"controller_mass_upf_failure_stats{service_name=\"rcm-controller\"}"
        - category: RCM controller UPF boot timer expiry count
          metric: controller_upf_boot_timer_expiry_stats
```

```
            description: "Count of total UPF boot timer expiry in rcm
controller"
            sampleQuery:
"controller_upf_boot_timer_expiry_stats{service_name=\"rcm-controller\"}"
            labels:
            -  label: endpoint
               description: "Endpoint address of UPF"
               example: "any ip address string"
         -  category: RCM controller IPC sent count
            metric: controller_ipc_sent_stats
            description: "Count of total IPC message sent from rcm
controller"
            sampleQuery: "controller_ipc_sent_stats{service_name=\"rcm-
controller\"}"
            labels:
            -  label: pod_name
               description: "Name of pod with which controller is
communicating"
               example: "rcm-configmgr or rcm-chkptmgr"
         -  category: RCM controller IPC sent error count
            metric: controller_ipc_sent_error_stats
            description: "Count of total IPC message sent error for rcm
controller"
            sampleQuery: "controller_ipc_sent_error_stats{service_name=\"rcm-
controller\"}"
            labels:
            -  label: pod_name
               description: "Name of pod with which controller is
communicating"
               example: "rcm-configmgr or rcm-chkptmgr"
         -  category: RCM controller IPC received count
            metric: controller_ipc_rcvd_stats
            description: "Count of total IPC message received by rcm
controller"
            sampleQuery: "controller_ipc_rcvd_stats{service_name=\"rcm-
controller\"}"
            labels:
            -  label: pod_name
               description: "Name of pod with which controller is
communicating"
```

```
            example: "rcm-configmgr or rcm-chkptmgr"
        - category: RCM controller UPF msg sent count
          metric: controller_upf_msg_sent_stats
          description: "Count of total message sent to UPF by rcm
controller"
          sampleQuery: "controller_upf_msg_sent_stats{service_name=\"rcm-
controller\"}"
          labels:
          - label: grpId
            description: "Group ID of UPF"
            example: "any valid string"
          - label: endpoint
            description: "Endpoint address of UPF"
            example: "any ip address string"
          - label: msg_type
            description: "Name of msg sent/recvd"
            example: "UpfMsgType_Registration, UpfMsgType_HB,
UpfMsgType_State etc"
        - category: RCM controller UPF msg received count
          metric: controller_upf_msg_rcvd_stats
          description: "Count of total message from UPF received by rcm
controller"
          sampleQuery: "controller_upf_msg_rcvd_stats{service_name=\"rcm-
controller\"}"
          labels:
          - label: grpId
            description: "Group ID of UPF"
            example: "any valid string"
          - label: endpoint
            description: "Endpoint address of UPF"
            example: "any ip address string"
          - label: msg_type
            description: "Name of msg sent/recvd"
            example: "UpfMsgType_Registration, UpfMsgType_HB,
UpfMsgType_State etc"
        - category: RCM controller UPF BFD event count
          metric: controller_upf_bfd_event_stats
```

```
            description: "Count of total BFD events received by rcm
controller"
            sampleQuery: "controller_upf_bfd_event_stats{service_name=\"rcm-
controller\"}"
            labels:
            - label: grpId
              description: "Group ID of UPF"
              example: "any valid string"
            - label: endpoint
              description: "Endpoint address of UPF"
              example: "any ip address string"
            - label: event
              description: "Name of bfd event"
              example: "bfd_up or bfd_down"
         - category: RCM controller BFD heartbeat failure count
           metric: controller_bfd_hb_timeout_stats
           description: "Count of total BFD heartbeat timeout received by
rcm controller"
           sampleQuery: "controller_bfd_hb_timeout_stats{service_name=\"rcm-
controller\"}"
         - category: RCM controller UPF TCP disconnect count
           metric: controller_upf_tcp_disconnect_stats
           description: "Count of total upf tcp disconnects received by rcm
controller"
           sampleQuery:
"controller_upf_tcp_disconnect_stats{service_name=\"rcm-controller\"}"
           labels:
           - label: grpId
             description: "Group ID of UPF"
             example: "any valid string"
           - label: endpoint
             description: "Endpoint address of UPF"
             example: "any ip address string"
         - category: RCM controller UPF TCP connect count
           metric: controller_upf_tcp_connect_stats
           description: "Count of total upf tcp connects received by rcm
controller"
           sampleQuery:
"controller_upf_tcp_connect_stats{service_name=\"rcm-controller\"}"
```

```
    labels:
    -  label: grpId
       description: "Group ID of UPF"
       example: "any valid string"
    -  label: endpoint
       description: "Endpoint address of UPF"
       example: "any ip address string"
```