Redundancy Configuration Manager Configuration and Administration Guide, Release 2025.04

First Published: 2025-10-31

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 UPs. The RCM provides N:M redundancy of UPs wherein "N" is number of Active UPs and is less than 10, and "M" is number of Standby UPs 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 the UPs and deployed as a VM. For 5G, the RCM is deployed as CN application using the SMI Cluster Manager (SMI CM).

The RCM supports the following functions:

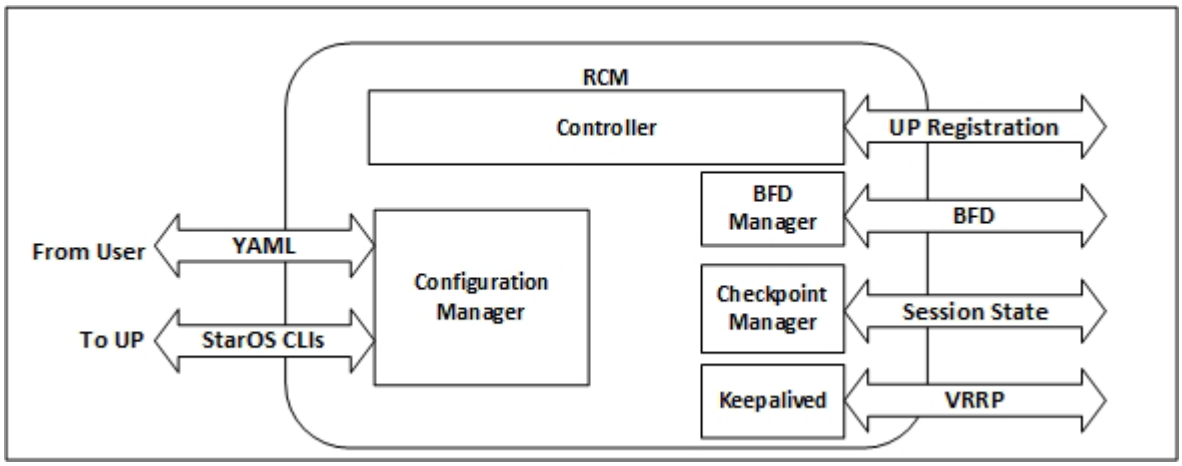
- **Redundancy Management**
  - Monitors the health of each Active UP
  - Selects a Standby UP to become Active when an Active UP becomes unreachable
  - Store session checkpoint information from Active UPs
  - Send session checkpoints to new Active UP during switchover
- **Configuration Management**
  - Decides the role of a UP; Active or Standby
  - Stores the configuration data that is sent to a UP
  - Provides the configuration data to a new UP (includes Day-0 to Day-N):
    - **Day-0**: Configuration required on UP 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.

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

Components

The following diagram illustrates the components of RCM.

Cisco Confidential



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

It is 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 22.04

Cisco Confidential

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

Deploying the RCM

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
```


Cisco Confidential

```
    groups: [admin, adm, audio, cdrom, dialout, docker, floppy, luser,
video, plugdev, dip, netdev]
    lock_passwd: false
    passwd:
$6$rounds=4096$Zl7FnVp7dYlT0bw$CFqAVNA8M1wxaEXwAVYlKfekSrTXbGUne1ihJ11xdNk
29bfCk2AURQG4XV.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 command 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, chartmuseum, and init_cluster services are running. You can verify by executing the following command:

      `systemctl status kubelet/chartmuseum/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.

   Note: The helm version has been upgraded from 2 to 3.

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

Cisco Confidential

- **systemctl status kubelet**
- **systemctl status kubeadm_init**
- **systemctl status calico_init**
- **systemctl status hardening**
- **systemctl status chartmuseum**
- **systemctl status init_cluster**

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

c. 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
```

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

kubectl get pods -n rcm

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

Note: Without configuring the UP credentials, 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:

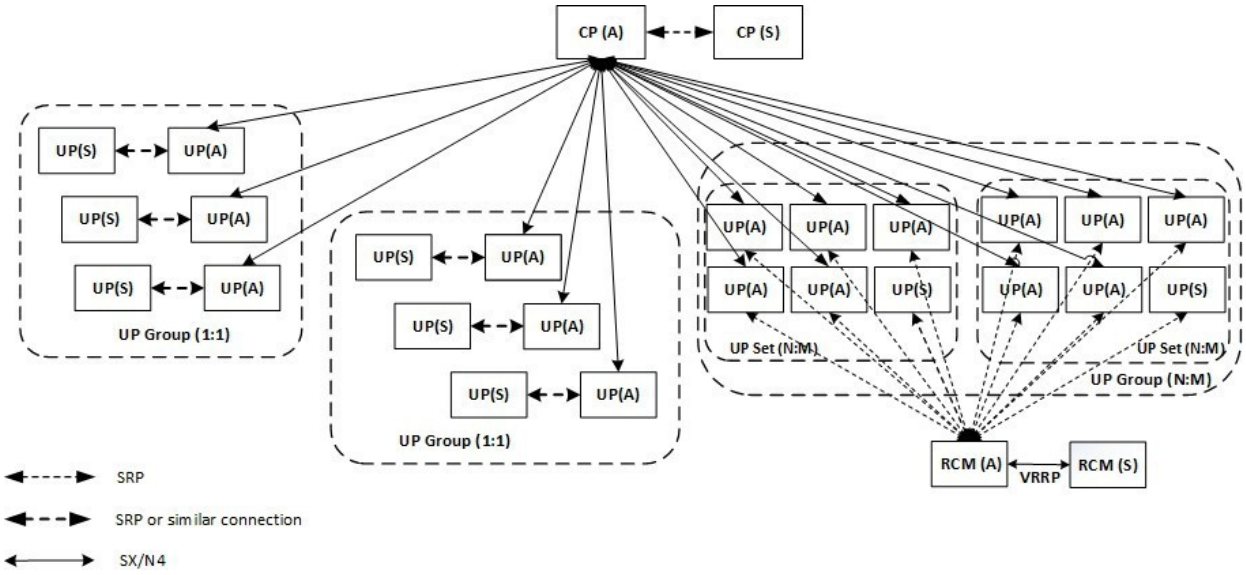
system mode running

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

RCM Deployment Architecture

The following diagram illustrates a typical RCM deployment model.

Cisco Confidential



- 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

Cisco Confidential

Redundancy Management

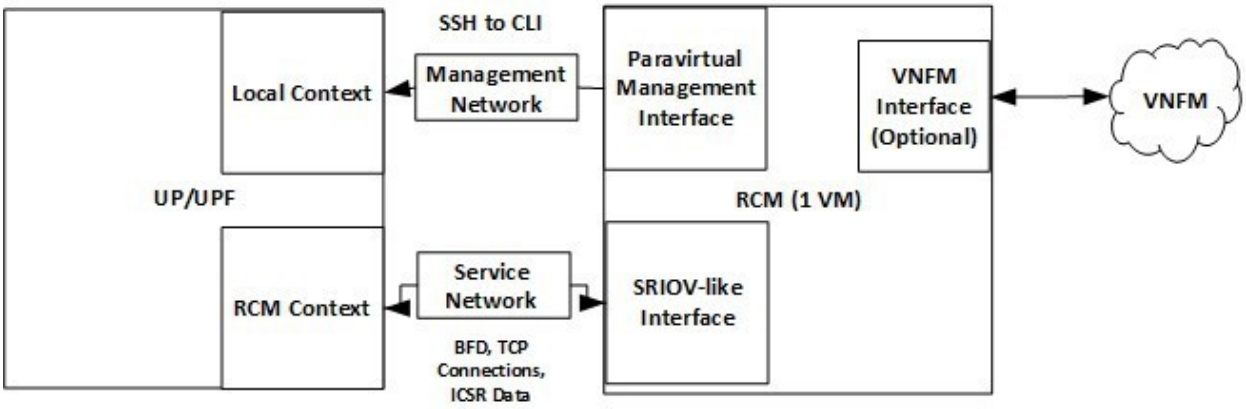
N:M redundancy using Redundancy and Configuration Manager (RCM), where N > 1 and M is at least 1, is a mechanism that is required to store all the required information at a common location. This common information is properly segregated based on “N” User Planes (UPs), and each UP contains “x” Session Managers (SessMgrs). On a switchover trigger, one of the “M” UPs 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 (ConfigMgr, RedMgr, and so on) helm charts are prepopulated in the disk image. An RCM-specific Ops Center provides the user interface to RCM. On bootup, the Ops Center starts. After Ops Center configuration, other RCM services comes up.

Cisco Confidential

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.

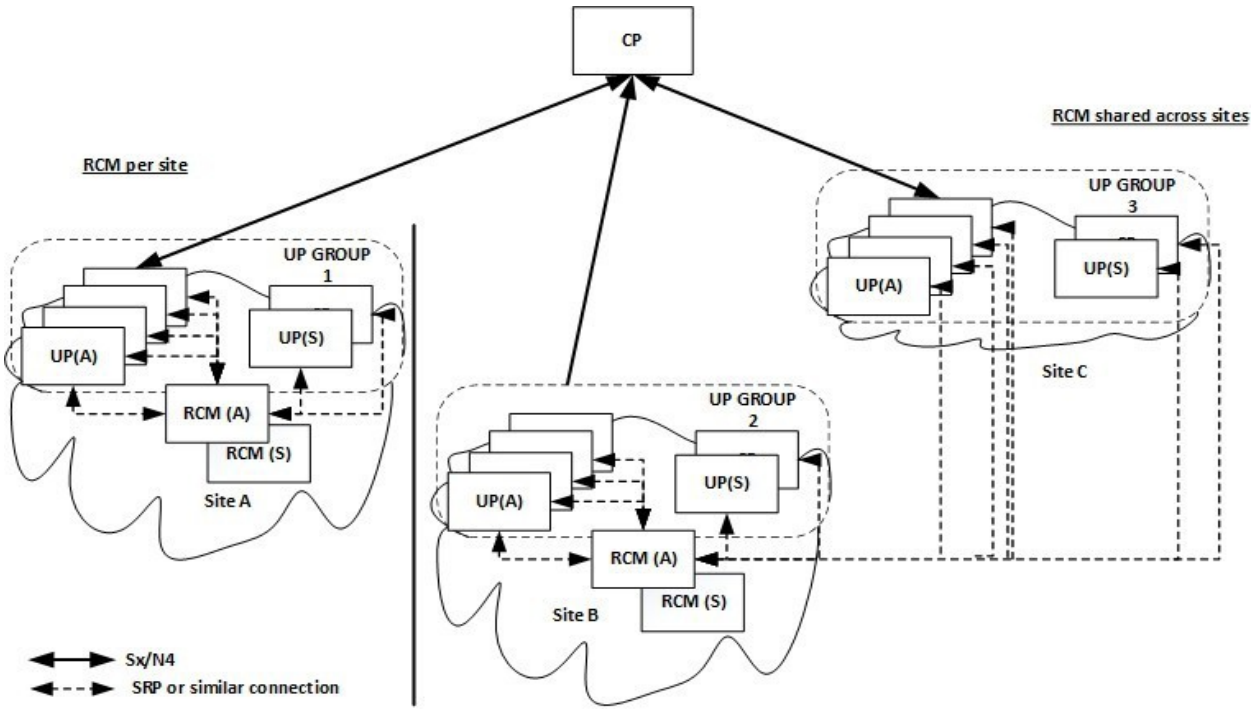
For deployment information, refer RCM Deployment section.

Upgrade Process

Rolling upgrade of components is currently not supported in the VM-based application. 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, they 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.

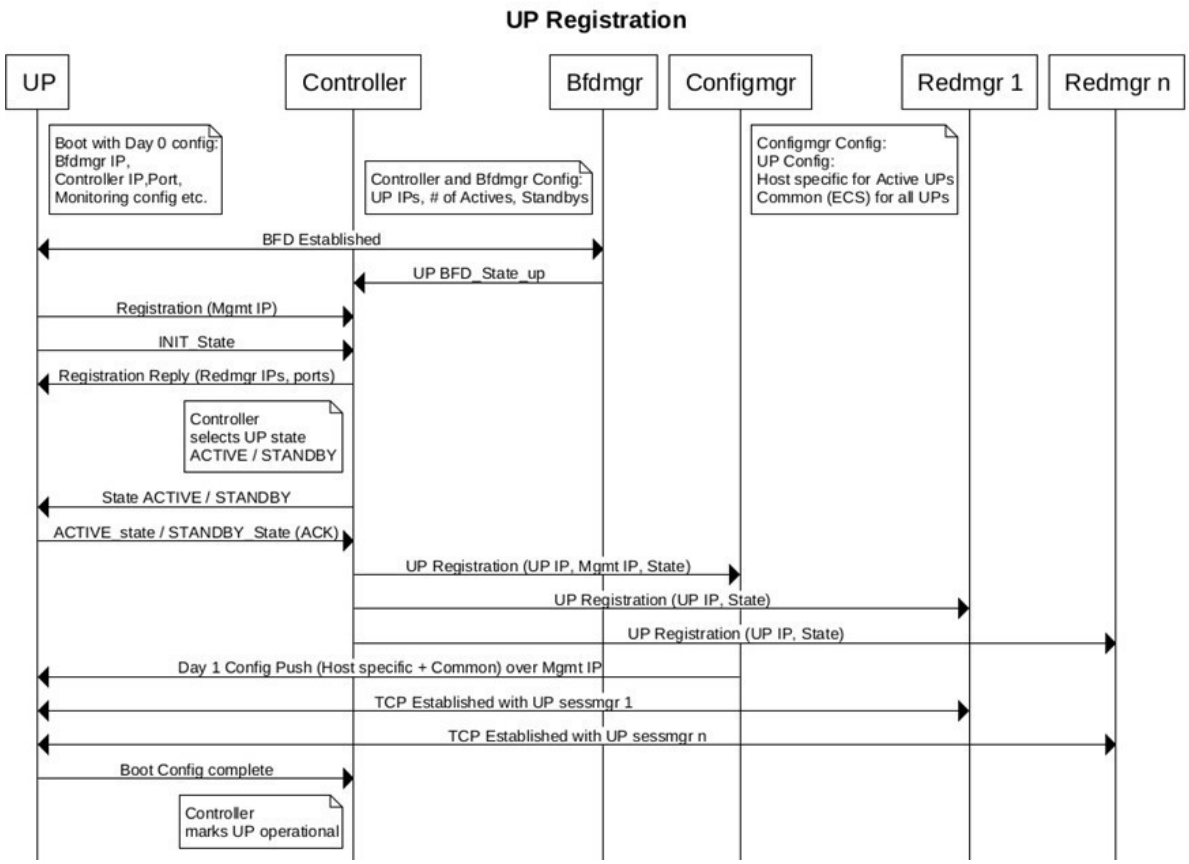
To ensure faster data retrieval, it is stored in memory. The checkpoint data that the Redundancy Manager receives is stored in the RAM. After careful segregation based on UP, session, and call, the checkpoint data is pushed to the new Active UP after a switchover from the RAM.

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.

RCM supports the following timers:

- **Init Wait timer:** The Init Wait timer defers the registration of Init state UPs and registers the active UPs first. This timer starts only when RCM controller starts or when RCM moves to HA MASTER state.

  When RCM controller starts or moves to HA MASTER state, the RCM controller has no UPF state and learns the UP state from the UPs itself. The Init state UPFs should not be assigned HostIDs that are already allocated to Active UPs.
- **Mass UPF Failure timer:** The Mass UPF Failure timer starts when all UPs lose BFD connectivity with RCM. Depending on the network deployment, there can be network connectivity issue between RCM and UPs. If RCM cannot establish BFD connectivity to any UPF within the timeout period, then RCM HA switchover is performed.

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 UPs Endpoint to Host ID. And, assigns the failed Active UPs 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 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:

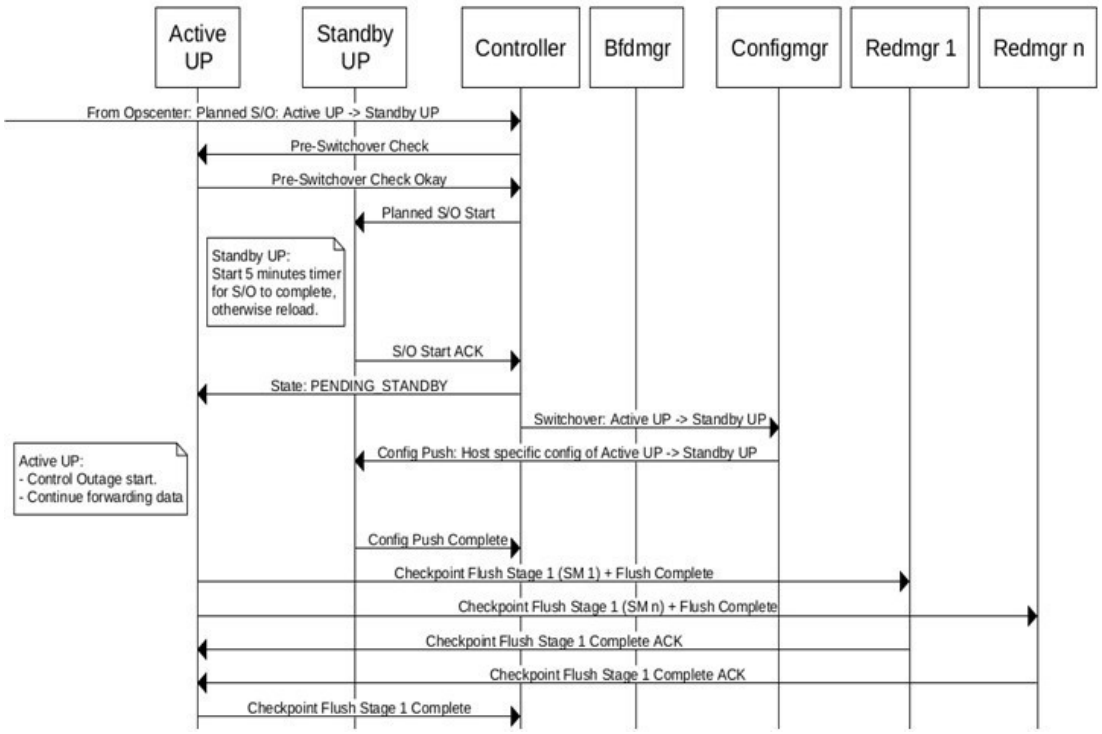
- When a UP is manually rebooted/reloaded during a maintenance window wherein the UP indicates to the RCM about the event of going down.
- When switchover is triggered from the RCM by either the Controller, Configmgr, or BFDMgr using the CLI. For information about the CLI command, refer the Performing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

UPF supports the following switchover timers that are configurable using CLI. See the Configuring Switchover Timers_section for configuration details.

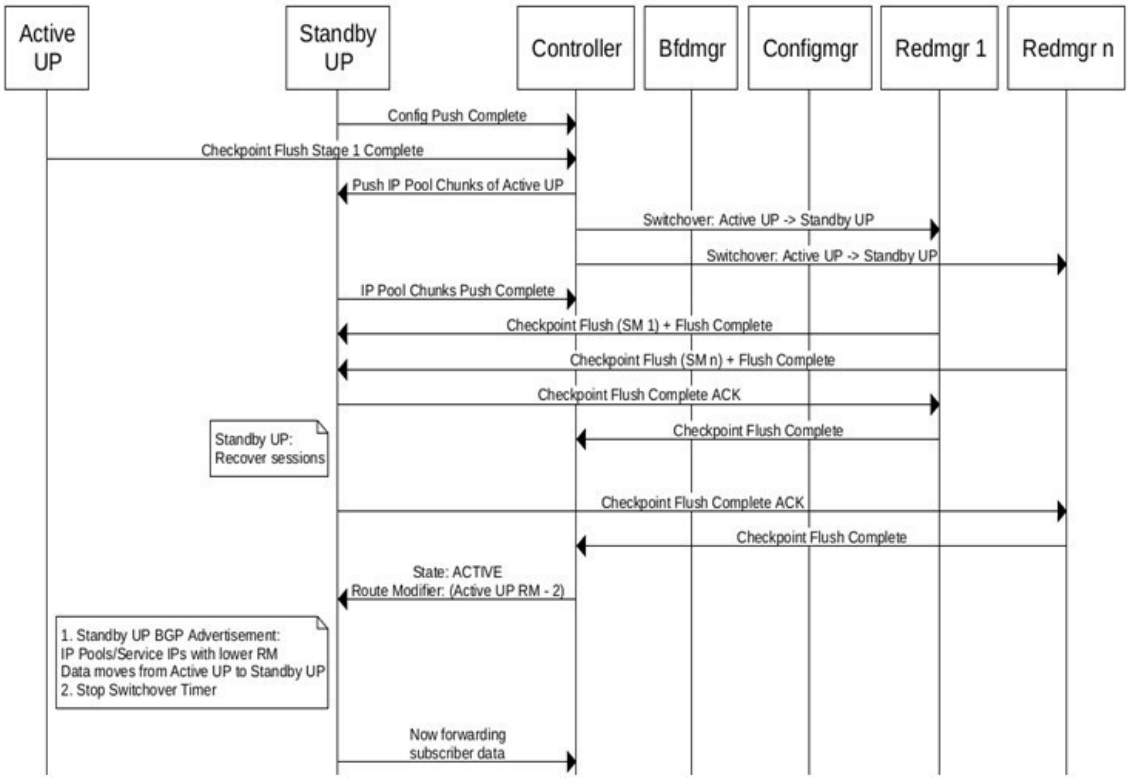
- Timer for planned switchover completion on new Active (Standby) UPF
- Timer for receipt of Standby state from the start of pending Standby state on old Active UPF

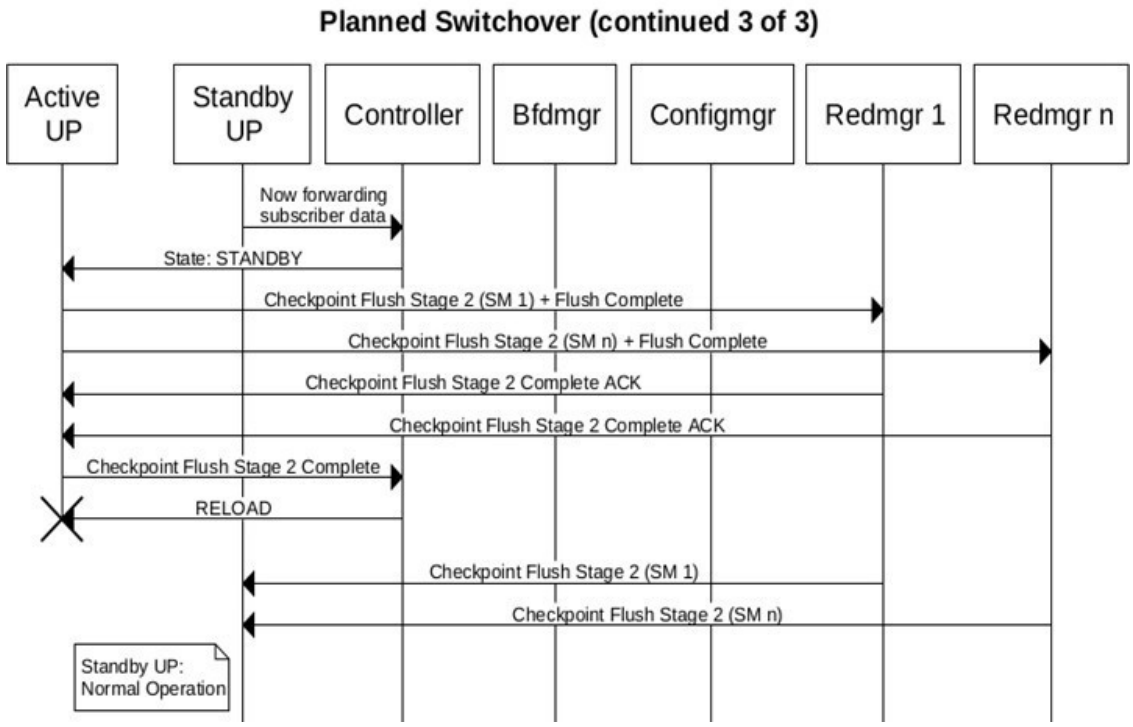
The following images illustrate the call flows for planned switchover.

Planned Switchover



Planned Switchover (continued 2 of 3)

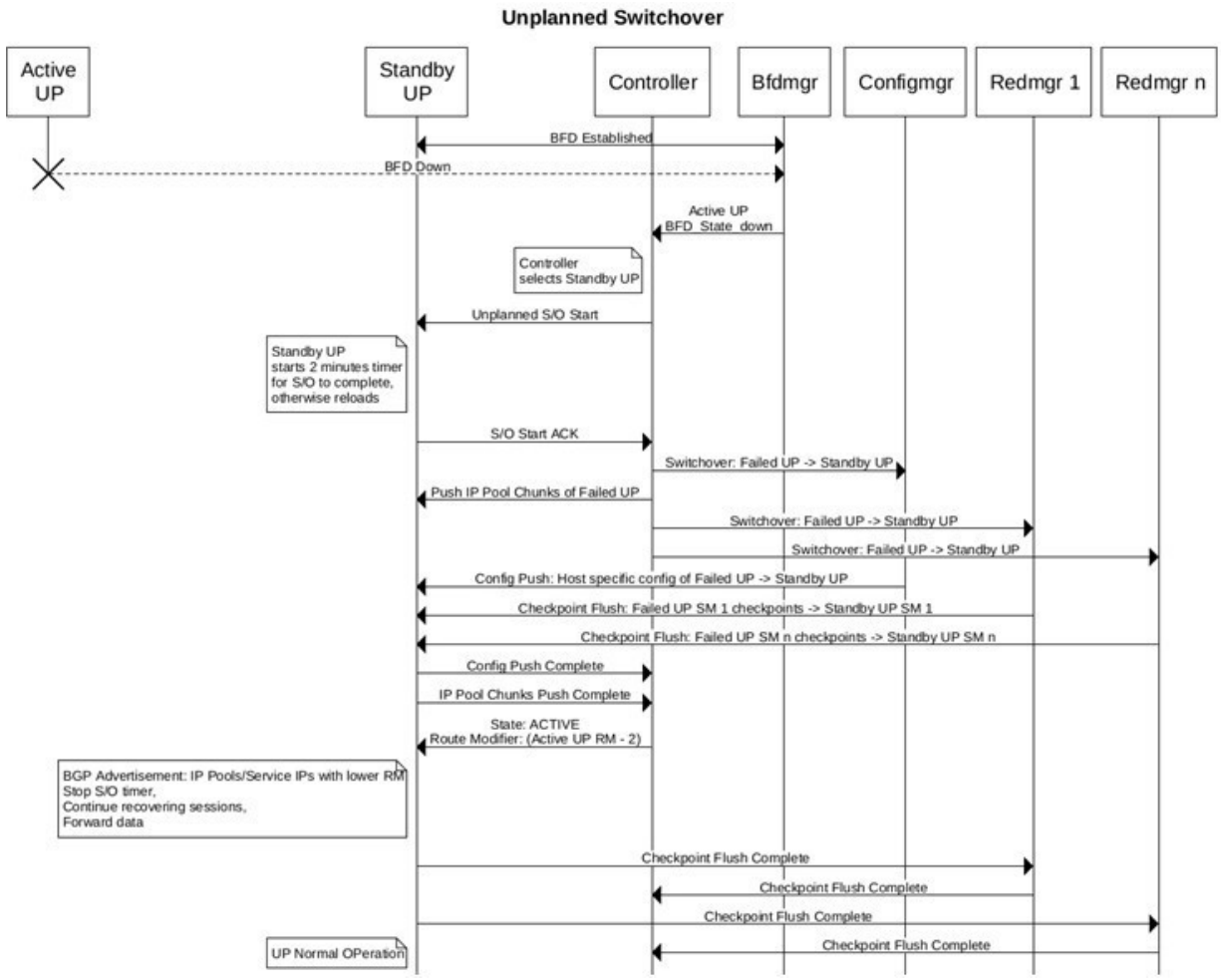




Unplanned Switchover

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, the **`switchover allow-checkpoint-processing-active`** CLI command 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.

NOTES:

- 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.

Monitoring TCP Connections using Heartbeat Communication

The application-level heartbeat mechanism between UPF and RCM allows you to monitor the health of the TCP connection. This feature resolves the half-closed TCP connections between RCM checkpoint managers and UP session managers.

RCM sends a heartbeat message every 3 seconds. It checks if it has received the heartbeat message from UPF in the last 60 seconds. RCM will close the TCP connection if it has not received the message. This behavior is applicable for both Active UP to RCM and RCM to Standby UP communication.

UPF also behaves similarly where it sends a heartbeat message every 3 seconds. UPF checks if it has received the heartbeat from RCM in the last 60 seconds. If UPF has not received the message, then it will close the TCP connection.

As part of RCM checkpoint manager hardening, UPF supports the heartbeat mechanism between UP sessmgr and RCM checkpoint manager.

MOP to Configure Heartbeat Communication

To enable heartbeat communication, perform the following steps:

1. Reload the standby UP in the build where the feature is supported.
2. Change boot priority to have the supported build in one active UP and perform an RCM-initiated planned UP switchover from this active UP to the standby UP mentioned in Step 1.
3. Upgrade the remaining active UPs by following the procedure in Step 2.
4. Upgrade RCM to the build where the feature is supported. The backup RCM must be upgraded first. Once upgraded, do a planned RCM HA from the other RCM instance, and upgrade the current backup RCM.
5. Enable heartbeat in both RCM using the following command:
   `[rcm-aio2/cndp-rcm1] rcm(config)#` **`k8 smf profile rcm-config-ep enable-up-heartbeat true`**
6. Enable the heartbeat in the standby UP by configuring the **`up-sm-heartbeat enable`** CLI command under RCM module in the boot config and then reload. After it comes up, register with RCM and config push complete. Wait till the sessmgr-checkpointmanager socket connections are in Standby Ready state.
7. Enable heartbeat in the active UPs in the boot config and the running configuration. Once the sessmgr-checkpointmanager socket connections are in Active Ready state, check if calls are getting checkpointed. Use the **`show rcm checkpoint statistics verbose`** command to verify the checkpointed connections.
8. Perform a UP switchover and check whether calls are recovered. If it works as expected, the feature is successfully enabled.

To disable heartbeat communication, perform the following steps:

1. Disable heartbeat in RCM using the following command:
   `[rcm-aio2/cndp-rcm1] rcm(config)#` **`k8 smf profile rcm-config-ep enable-up-heartbeat false`**

2. Change the boot config of the standby and active UPs with the **up-sm-heartbeat disable** command. Reload the standby UP to re-register with RCM and config push complete.
3. In the active UPs, disable heartbeat in the running config with the **up-sm-heartbeat disable** command.
4. Check whether checkpointing works from the active UPs using the **show rcm checkpoint statistics verbose** command. Using the same CLI, check if the sessmgr-sockets are in proper state in both active and standby UPs. It must be Active Ready in the active UP and Standby Ready in the standby UP.

Configuring Heartbeat Mechanism

Use the following configuration to configure the heartbeat mechanism.

Configurating Heartbeat in RCM

To enable or disable the heartbeat from RCM to active or standby UPF, use the following configuration in RCM:

```
config
   k8 smf profile rcm-config-ep enable-up-heartbeat { true | false }
   end
```

To verify the total number of heartbeat messages received and sent, use the following command:

```
rcm show-statistics checkpointmgr-endpointstats
```

Configurating Heartbeat in UPF

To enable or disable the heartbeat from UP session manager to RCM checkpoint manager, use the following configuration in UPF:

```
configure
   context context_name
      redundancy-configuration-module rcm_name
         up-sm-heartbeat { enable | disable }
         end
```

To verify the total number of heartbeat messages received and sent, use the following command:

```
show rcm checkpoint statistics sessmgr all
```

Configuring Socket Write Timeout

The TCP socket from RCM Controller or RCM CheckpointMgr towards UPF SessMgr could get blocked. The socket write timeout configuration can determine the timeout if the TCP socket gets blocked. If an error such as I/O timeout is seen in the Controller or CheckpointMgr logs, you can try increasing the timeout value. You can configure the write timeout value based on your tolerance requirements.

Use the following configuration to configure socket write timeout from RCM Controller to UPF:

```
k8 smf profile rcm-config-ep upf-write-timeout write_timeout_value
```

write_timeout_value must be an integer ranging from 300 to 60000 milliseconds. The default value is 1000 milliseconds.

Use the following configuration to configure socket write timeout from RCM CheckpointMgr to UPF:

k8 smf profile rcm-config-ep write-timeout *write_timeout_value*

write_timeout_value must be an integer from 1 to 60000 milliseconds. The default value is 10000 milliseconds.

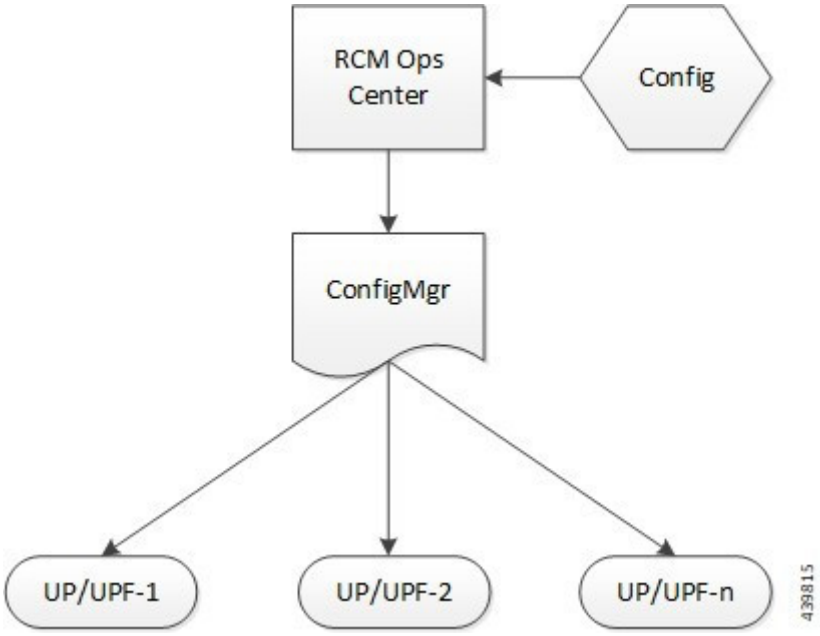
Configuration Management

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

- 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.

  NOTE: If NSO is the configurator, the Day-1 configuration is pushed by NSO and not by ConfigMgr.
- 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 UP is up, the Day-0.5 configuration is executed on UP. When UPs register with RCM, the Controller pushes the hostID and SSH IP to UP 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 UP 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 is 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.
- You can verify the status of SSH IP installation by using **show rcm info** CLI command.

IMPORTANT: 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.

Redesigning Configuration Manager to Support Large Common Configurations

When the common configuration for a redundancy group exceeds 80K lines, the configuration map that is created in the RCM exceeds 1 MB, which is the defined hard limit for the combined size of all the configuration maps created out of a single repository. When the RCM supports more than one redundancy group also, you must extend this hard limit as it turns out to be a very crucial limit. This feature extends the existing hard limit to support large common configurations.

The following CLIs are added for CLI notification support in Ops-center.

[no] k8 smf profile rcm-config-ep commo-config redundancy-group *group-id* **file** *file-name*

k8 smf profile rcm-config-ep common-config update redundancy-group *group-id*

Mount the host directory on the configuration manager pod by mounting the path as follows:

```
spec:
  template:
     spec:
       volumes:
          -name: common-config
          hostPath:
               path: /data/<rcm ns>/common_config
               type: Directory
        containers:
           VolumeMounts:
             -name: common-config
              mountPath: /config/common_config
```

Configuration Manager receives the following notifications from the RCM-Ops-center:

- To apply the configuration file entirely through the configuration (no) mode.

- To update the configuration with the day N update.

How it Works

You can use this feature by completing the following procedure.

5. A new script **apply_config_v2.sh** is available as part of the VM image's file system.
6. In the same directory, define the **connect_file** as described in the following example:

   ubuntu@bgl26-pl-cups1:~/rcm-scripts/config$ cat connect_file

   #Testing

   #VM:VMUSERNAME:VMPASSWORD:OPCENTER:OPCUSERNAME:OPCPASSWORD:OPC_EXTERNALIP

   1.2.3.4:root:starent:rcm:admin:Mitg@123:1.2.3.4
7. Create the directory **/data/<rcm_ns>/common_config/** in the RCM VM as root user. This step is very important as it enables the display of the **Configmgr** pod.
8. Use the root credentials of RCM VM specified in the **connect_file**. Root credentials are mandatory as you copy the file to RCM VM and the permissions of the encrypted configuration file must be a minimum of 644 KB in size.
9. Expose the ops-center IP using the following CLI command. This is not a mandatory step as it is executed by the script. If you run the script remotely, this step is then required.

   kubectl expose deployment ops-center-rcm-ops-center—type = LoadBalancer –name = ops-center-rcm-ops-center-cli -- port = 2024 -- target-port = 2024 -n rcm -- external-ip = *RCM_VM_IP*

RCM_VM_IP — Any IP that is reachable from the Host machine in which you can run the script.

10. Disable the string based common configuration map from the ops-center using the following CLI command.

    ```
    config
       k8 smf profile rcm-config-ep disable-cm common
       commit
    end
    ```

Applying the First time Configuration File

bash -x ./apply_config_v2.sh -g 1 -n -c UPCommon.cfg -P /data/<rcm_ns>/common_config/config4.cfg -G 4 -p connect_file

-g1 – It is a group ID and must be an integer.

-n – New configuration file. Use to change a configuration file name.

-c UPCommon.cfg – Apply the StarOS configuration file.

-P /data/<rcm_ns>/common_config/config3.cfg – Use this path in RCM VM to copy the file.

-p connect-file – This file provides all the information about the RCM VM and RCM ops-center credentials to login, copy, and apply the configuration in the RCM ops-center.

Note the password as the same password must be used for configuration update.

No Config Command

bash -x ./apply_config_v2.sh -g 1 -r -G 4 -p connect_file

-r – Specifies that it is a “no common config” for this group.

Backward Compatibility

To move back to string-based approach, re-enable the cm “common” with this configuration change.

```
config
   no k8 smf profile rcm-config-ep disable-cm common
   commit
end
```

Limitations and Restrictions

Place the configuration file in a directory that is mountable by the configuration manager. For the configuration manager to read the file, the file must be placed in the **/data/<rcm_ns>/common-config** directory.

The supported host configuration and common configuration combinations are described as follows.

The following combinations are supported with RCM as the configurator:

- Host Configuration: String-based Approach
- Common Configuration: String-based or File-based Approach

The following combinations are supported with NSO as the configurator:

- Host Configuration: Yang-based Approach
- Common Configuration: Not applicable as NSO pushes the common configuration

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

NOTES:

- **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.

k8 smf profile rcm-config-ep switchover disallow-swo-timeout *timeout_seconds*

NOTES:

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

Example Ops Center Host-specific Configuration

```
svc-type upinterface
redundancy-group 1
…
  host Active2
   host 1 config
   host 2 "  context EPC2"
   host 3 "    interface S5_SGW_PGW_loopback_up2 loopback"
   host 4 "      ip address 192.168.170.99 255.255.255.255"
   host 5 "    #exit"

svc-type cpgrp
redundancy-group 1
  host Active2
   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

   host 10 "config"
   host 11 "context rcm"
   host 12 "redundancy-configuration-module rcm"
   host 13 "monitor sx context EPC2 bind-address 192.168.170.99 peer-address
192.168.196.10"
   host 14 "end"

  exit
```

```
exit
exit
```

NOTE: The **monitor sx context EPC2 bind-address** *192.168.170.99* **peer-address** *192.168.196.10* CLI command must be part of “cpgrp svc-type” so that this configuration is pushed only to Actives during the Day1 config-push. Only at the time of switchover, this configuration is pushed to new Active.

Configuring the RCM

This section describes how to configure the RCM.

Configure IPv6 Addressing

IPv4 support is enabled by default on RCM. To enable IPv6 communication with the endpoints, use the following configuration.

k8 smf profile networking-type-dual-stack-enable { false | true }

NOTES:

- **k8 smf profile**: Specify the Kubernetes (K8s) SMF configuration with the external IP or port configuration.
- **networking-type-dual-stack-enable { false | true }**: Set this configuration to **true** to enable IPv6 configuration. The configuration is set to **false** by default.

IMPORTANT: You can configure either the IPv4 or the IPv6 address. In the running configuration, only one IP address type can be used.

Configuring the Controller Pod

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

```
config
   endpoint rcm-controller
     vip-ip ipv4_address
     vip-ipv6 ipv6_address
     exit
```

NOTES:

- **vip-ip** *ipv4_address*: Specify the host IPv4 address.
- **vip-ipv6** *ipv6_address*: Specify the host IPv6 address

NOTE: You can configure either the IPv4 or the IPv6 address. In the running configuration, only one address type can be used.

- 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.

```
config
   endpoint rcm-bfdmgr
   vip-ip ipv4_address
   vip-ipv6 ipv6_address
   exit
k8 smf profile rcm-bfd-ep bfd-monitor group group_id
k8 smf profile rcm-bfd-ep host-networking { true | false }
k8 smf profile rcm-bfd-ep node-port-enabled { true | false }
k8 smf profile rcm-bfd-ep setvar bfd-src-ip ip_address
k8 smf profile rcm-bfd-ep { infraprometheus-port | ipc-ep-port | metrics-port
| pprof-ep-port | rest-port | resthealth-port } port_number
   endpoint ip IPv4/IPv6_address
   endpoint ip IPv4/IPv6_address
   endpoint ip IPv4/IPv6_address
   min-tx-int tx_microseconds
   min-rx-int rx_microseconds
   multiplier count
   standby count
   exit
```

NOTES:

- **k8 smf profile**: Specify the Kubernetes (K8s) SMF configuration with the external IP or port configuration.
- **rcm-bfd-ep**: Specify the BFD endpoint parameters.
- **bfd-monitor**: Specify the BFD application configuration.
- **group** *group_id*: Specify the group ID.
- **bfd-src-ip**: Specify the source IP of BFD packets.
- **standby** *count*: Specify the number of required Standby UP/UPFs.
- **vip-ip** *ipv4_address*: Specify the host IPv4 address.
- **vip-ipv6** *ipv6_address*: Specify the host IPv6 address

NOTE: You can configure either the IPv4 or the IPv6 address. In the running configuration, only one address type can be used.

- **k8 smf profile rcm-bfd-ep { infraprometheus-port | ipc-ep-port | metrics-port | pprof-ep-port | rest-port | resthealth-port }** *port_number*: Specify the port number for the specific BFDMgr endpoint.
- The **k8 smf profile rcm-bfd-ep setvar bfd-src-ip** *ip_address* command is deprecated in 2023.03.0 and later releases. This command is not required to display BFDMgr statistics.
- 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 UP Credentials section.

Configuring UP Credentials

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

```
config
   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 UP 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
```

NOTES:

- **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 IPv4/IPv6 address of UP/UPF over which configuration push happens.

  NOTE: You can configure either the IPv4 or the IPv6 address. In the running configuration, only one address type can be used.
- *standby_name1* can be any name.

Sample Configuration for SSH Key-based UP 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*

NOTES:

- **upf mgmt-ip** *up_n_and_m_mgmt_ip* **upfsshinfo username** *username* **private-key** *private_key*: These credentials are required only for per-UP unique-credential scenario.
- After **private-key** keyword, press "Enter" for Multiline mode and then enter the private key.
- You must pass the Management IPv4/IPv6 address of the UPs for configuration push.

  NOTE: You can configure either the IPv4 or the IPv6 address. In the running configuration, only one address type can be used.

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.

```
config

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

   exit
```

NOTES:

- **{ chargingAction | creditCtrl | global | gtpp | gtpuService | misacs | nnrfService | 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.
- **nnrfService**: Restricts the NNRF service 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 Init Wait Timeout and Mass UPF Failure Timeout

Use the following RCM OpsCenter Configuration mode CLIs to configure the Init Wait timer and Mass UPF Failure timer.

```
k8 smf profile rcm-config-ep init-wait-timeout init_wait_timeout
k8 smf profile rcm-config-ep mass-upf-failure-timeout upf_failure_timeout
```

NOTES:

- **init-wait-timeout** *init_wait_timeout*: Specifies the Init Wait timer in seconds, as an integer from 0 to 300.

  Default: 300 seconds. A value of 0 disables the Init Wait timer.
- **mass-upf-failure-timeout** *upf_failure_timeout*: Specifies the Mass UPF Failure timer in minutes, as an integer from 0 to 60.

  A value of less than 3 minutes disables the timer. UPFs need at least three minutes to reload. When all UPFs are simultaneously reloaded as part of UPF-RCM workflow, a value of less than three minutes can potentially cause false positive alarms.

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 IPv4/IPv6 address of the UP, which the Configuration Manager uses to securely connect. *up_mgmt_addr* is the IPv4/IPv6 address of the UP.

  NOTE: You can configure either the IPv4 or the IPv6 address. In the running configuration, only one address type can be used.
- **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 Switchover Timers in UPF

Use the following configuration to configure the planned switchover timers in UPF.

```
configure
   rcm-service rcm_svc_name
      planned-standby-timeout planned_timer_value
      pending-standby-timeout pending_timer_value
      end
```

NOTES:

- **planned-standby-timeout** *planned_timer_value*: Specify the planned switchover timeout, in seconds. This timer is applicable only when UPF is in the Destination UPF role. Destination UPF will reload if the planned UPF switchover does not complete within this timeout period.

  planned_timer_value must be an integer from 300 to 3600. Default: 300 seconds.

- **pending-standby-timeout** *pending_timer_value*: Specify the pending standby timeout, in seconds. This timer is applicable only when UPF is in the Source UPF role. If the source UPF does not receive Standby state from RCM within this timeout period, it will revert back to Active from Pending Standby.

  pending_timer_value must be an integer from 300 to 3600. Default: 300 seconds.

Configuring Switchover Timers in RCM

Use the following configuration to configure the switchover timers in RCM.

```
config
   k8 smf profile rcm-config-ep swo-timeouts { pre-switchover
preswitchover_timer_value | stage1-chkpt-flush stage1flush_timer_value |
stage2-chkpt-flush stage2flush_timer_value }
   end
```

NOTES:

- **pre-switchover** *preswitchover_timer_value*: Specify the Source UPF preswitchover check timer in seconds. The planned UPF switchover will be aborted on timeout.

  preswitchover_timer_value is an integer from 15 to 3600. Default: 15 seconds.

- **stage1-chkpt-flush** *stage1flush_timer_value*: Specify the stage 1 checkpoint flush from Source UPF to CheckpointMgrs, in seconds. The planned UPF switchover will be aborted on timeout. The Source UPF will be reverted to Active and Destination UPF will be reloaded.

  stage1flush_timer_value must be an integer from 15 to 3600. Default: 15 seconds.

- **stage2-chkpt-flush** *stage2flush_timer_value*: Specify the stage 2 checkpoint flush from Source UPF to CheckpointMgrs, in seconds. Failure or timeout of stage 2 checkpoint flush does not fail the planned UPF switchover.

  stage2flush_timer_value must be an integer from 10 to 3600. Default: 10 seconds.

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.

Configuring Local Time Zone

Use the following configuration to set the local time zone in RCM Ops Center.

```
configure
   k8 smf profile timezone Europe/Warsaw
   exit
```

NOTES:

- **timezone Europe/Warsaw**: Specify to change the time zone for all RCM pods.
- This command causes pod restart for the value to take effect.
- It is recommended to configure this command as part of the day 0 configuration.

IP Pool Audit Configuration

Triggering IP Pool Audit Message

Use the following command in the Exec mode to trigger the IP pool audit message manually.

UPF sends the audit message to RCM. It also stops and restarts any running audit timer to maintain the periodical audit of configured IP pool chunks.

```
send-ip-pool-audit
```

Configuring IP Pool Audit Timer

Use the following configuration to configure the time interval between successive audits:

```
configure
   context context_name
      redundancy-configuration-module rcm_name
         [ no ] ip-pool-audit interval audit_timer
         end
```

NOTES:

- **ip-pool-audit interval** *audit_timer*: Specify to set a time interval to repeat the auditing of IP pool information towards RCM controller. *audit_timer* is the timer value ranging from 900 to 43200 seconds.
- The default value of the timer is 1 hour.
- The **no ip-pool-audit interval** command disables the timer.

Prompting Wrong Route Modifiers

UPF and RCM controller will not accept out-of-limit route modifiers. If UPF or RCM controller receives an invalid route modifier, it disconnects TCP. On reconnecting, UPF sends the available route modifier.

The route modifier 50 is applicable to a standby UPF or RCM node only. Hence, the Active or Pending standby UPF will not accept the standby route modifier value.

Use the following configuration to prompt wrong route modifiers in both RCM and UPF:

```
configure
   context context_name
      redundancy-configuration-module rcm_name
         rcm route-modifier-destination rcm_vpnmgr_ip val <2..32,50> dry-run
{ true | false }
         end
```

NOTES:

- When **dry-run** is set to **true** (default value), it only prints the route modifier value and the UPF state will be sent to UPF.
- To send the route modifier to UPF, set **dry-run** to **false**.

Configuring LZ4 Compression Algorithm

In RCM solutions, you can choose either LZ4 or zlib compression algorithm for session checkpoints. The compression algorithm is configured in UP under RCM configurations.

For detailed configuration steps, see the *UPC CUPS User Plane Administration Guide*. Complete MOP is provided below.

In 21.23.26, 21.28.0 and later releases, perform the following steps to change the compression algorithm from zlib to LZ4:

MOP at RCM System level:

11. On RCM Ops Center, use the **rcm pause switchover true** CLI command to prevent UP(F) switchover.
12. On all UPs, update the compression algorithm to LZ4 (in Day-0.5 config and running-config) across the redundancy group level.

    Use the **show config context** *context_name* or **show config url** *url* CLI command to verify if **checkpoint session compression lz4** CLI command is enabled.
13. Restart all the CheckpointMgr containers and wait for all the checkpoints to resync, or perform RCM HA.

    For example,

    kubectl -n rcm get pod rcm-checkpointmgr-0 -o yaml | grep -i "containerID: docker

```
- containerID:
docker://3f7e6b10a1be3005424eae148cca2905df8e24e0a549069dfacba533c7b57bf3
```

sudo docker restart *3f7e6b10a1be3005424eae148cca2905df8e24e0a549069dfacba533c7b57bf3*

```
[sudo] password:
3f7e6b10a1be3005424eae148cca2905df8e24e0a549069dfacba533c7b57bf3
```

In case of RCM HA, execute the **rcm migrate primary** CLI command on the primary RCM Ops Center.

14. Use the **rcm pause switchover false** CLI command to revert the **rcm pause switchover** value to **false.**

MOP at Redundancy Group level:

15. On RCM Ops Center, use the **rcm pause switchover true red-group** *red_group_number* CLI command to prevent UP(F) switchover.
16. On all UPs, update the compression algorithm to LZ4 (in Day-0.5 config and running-config) across the redundancy group level.

    Use the **show config context** *context_name* or **show config url** *url* CLI command to verify if **checkpoint session compression lz4** CLI command is enabled.
17. On the UP, bring down the RCM interface, and then bring it up.

    The following is a sample configuration to bring down the RCM interface:

```
Configure
   port ethernet 1/10
      vlan 2199
         shutdown
```

18. On RCM Ops Center, use the **rcm pause switchover false red-group** *red_group_number* CLI command to revert the **rcm pause switchover** value to **false.**

NOTE: You must follow the same MOP to change the compression algorithm from LZ4 to zlib, replacing the keyword **lz4** with **zlib**.

Changing RCM State from Pending-Active to Active

Use the following configuration to change the RCM state from Pending-Active to Active.

```
configure
   context context_name
      redundancy-configuration-module rcm_name
         force-pactv-to-actv-timeout value_seconds
         end
```

NOTES:

- *value_seconds*: Specifies the timeout value in seconds. You must set the timeout value to *1* for immediate switchover to Active state. Immediate switchover prevents the switchback from Active to Pending-Active state.

RCM Configuration to Enable NSO as Configurator

The Cisco Network Service Orchestrator (NSO) based configuration management for 4G CUPS supports:

- Onboarding of Cisco Virtual Network Function (VNF) devices—CP, UP, and RCM
- Centralized configuration management of 4G-based CPs, UPs, and RCMs for Day-N, Day-1, and Day-0.5 CUPS configuration push.
    - Day-0.5 applies to N:M UP redundancy scheme that uses RCM. The Day-0.5 configuration is intended for the UP to communicate to the RCM, so that its role can be defined, and suitable configuration be pushed subsequently.

Managing customer configuration management for 4G CUPS deployments using NSO automation also exhibits reusability, standard notification management, and systematic device configuration governance.

For details about NS0-based Configuration Management and NSO Orchestration for 4G CUPS, refer to the *Ultra Packet Core CUPS Control Plane Administration Guide* or the *Ultra Packet Core CUPS User Plane Administration Guide.*

The RCM OpsCenter Configuration mode CLIs must be configured as follows:

```
k8 smf profile rcm-config-ep config-mode NSO

k8 smf profile rcm-config-ep switchover deployment false
```

When NSO is the configurator, the Common and Host-specific configuration with string-based approach must not be configured on RCM.

The following is an example RCM (NSO-specific) and Host-specific configuration to be configured in RCM:

```
config

k8 smf profile rcm-config-ep switchover deployment false

k8 smf profile rcm-config-ep config-mode NSO
```

The following is an example configuration that is required in RCM. The SSH IP can be random IP addresses and not any working IPs. The IP address can be an IPv4 or an IPv6 address.

NOTE: You can configure either the IPv4 or the IPv6 address. In the running configuration, only one address type can be used.

```
k8 smf profile rcm-config-ep disable-cm apn gtpp creditCtrl packetFilter
urrList ruledef rulebase miscacs global chargingAction lawfulIntercept
apnprofile common

k8 smf profile rcm-config-ep rcm-upfinfo group 1

 host-id active Active1 ssh-ip 1.1.1.1 management-ip 1.2.3.10

 host-id active Active2 ssh-ip 1.1.1.2 management-ip 1.2.3.11

 host-id standby Standby1 ssh-ip 1.1.1.3 management-ip 1.2.3.12
```

upf mgmt-ip 1.2.3.10

upf mgmt-ip 1.2.3.11

upf mgmt-ip 1.2.3.12

exit

Preventing Multiple Configuration Push Notifications

To prevent multiple configuration push notifications toward NSO, the following CLI command is introduced in 21.26.4 and later releases:

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

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

Sample Configuration

The following is an example configuration on RCM pushed from NSO that is present in NSO as a separate file per Active UP:

```
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
nnrf-nfm-service svc-up1     //only nnrf service names should be configured
  redundancy-group 1
   host Active1
   exit
  exit
exit
nnrf-nfm-service svc-up2
  redundancy-group 1
   host Active2
   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
```

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

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 SNMP event trap history. Displays details for the latest 5000 SNMP traps.	**rcm show-snmp-trap history**	RCM Ops Center
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
- UPFStateAssigned

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

  NOTE: In 21.23.x and later releases, the PodNotFound and PodNotRunning traps are obsoleted. As replacement, the RCMPodHealthCheck trap is added.
- RCMPodHealthCheck
- RCMStateChange

  NOTE: In 21.23.x and later releases, the RCMStateChange trap is obsoleted. As replacement, the RCMStateChangeToFault, RCMStateChangeToActive, RCMStateChangeToActive, and `RCMControllerState` traps are added.
- RCMStateChangeToFault
- RCMStateChangeToActive
- RCMStateChangeToStandby
- `RCMControllerState`
- RCMControllerStateUpdateFailure – This SNMP trap is raised when RCM state change through HTTP POST from RCM keepalived to RCM controller fails.

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* **| clear-dflt-traps | default-external-vip** *ip_address* **| custom-port** *port_number* **[auth { md5 [auth-key | port | priv { aes | aes192 | aes256 | des | none } | user-name] | none | sha }] |host-networking { false | true } | 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 IPv4/IPv6 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 IPv4/IPv6 address.
- **clear-dflt-traps**: Enable the default traps that are disabled by default.

  The following traps are disabled by default to prevent flooding of less significant traps in the **rcm show-snmp-trap history** command:

  - ActiveSessmgrConnected
  - ActiveSessmgrDisconnected
  - CheckpointAuditEnded
  - CheckpointAuditStarted
  - StandbySessmgrConnected
  - StandbySessmgrDisconnected
- **default-external-vip** *ip_address*: Specifies the source IP for SNMP traps. Prior to sending any trap to the trap-receiver, this IP address is used as the source of the originating trap.
- **custom-port** *port_number*: Specifies the custom port number. This option must be used only when the **default-external-vip** *ip_address* is defined.
- **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.
- **host-networking { false | true }**: Configures the host networking mode or non-host networking mode.
- **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.

NOTE: You can configure either the IPv4 or the IPv6 address. In the running configuration, only one address type can be 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

Core Dump Collection and Generation

RCM allows network operators to generate a full memory dump of the applications or pods for analysis and troubleshooting purposes. In addition to logs and statistics, the full core dump eases debugging and provides better serviceability. This feature applies to both SMI and VM-based RCM deployments.

A signal handler in RCM handles the SIGUSR1 signal that obtains the core of running applications or pods. You can generate core dump for the Controller, BFDMgr, CheckpointMgr, and ConfigMgr pods.

If the application crashes in the pods, RCM will generate the core dump automatically.

Generate Core Dump

You can use this debug command to generate core dump by sending SIGUSR1 to the application without terminating the application:

kubectl -n rcm exec -it <*pod_name*> -- kill -SIGUSR1 1

You can use the **coredumpctl** command to view the generated core dump.

Example:

```
$ kubectl -n rcm exec -it rcm-controller -- kill -SIGUSR1 1
$ coredumpctl
```

```
TIME                          PID    UID GID SIG COREFILE  EXE
Wed 2024-05-09 08:36:03 UTC 795309 303 303  6  present    /opt/workspace/rcm-controller
```

IMPORTANT: You are recommended to execute this command only upon request from your Cisco account representative.

Retrieve Core Files

For a CNDP-based deployment, RCM compresses the collected core files and stores the files on an internal Apache server. When the application crashes, you can retrieve the core dump from the Apache server using the **`tac-debug-pkg`** command from the CEE Ops-Center CLI console.

For a VM-based deployment, you must log on to the VM and retrieve the core files manually from the */var/lib/systemd/coredump/* location.

For more information on storage and retrieval of core files, see the *UCC CEE Configuration and Administration Guide* > *Common Execution Environment* chapter > *Gather TAC* section.

Troubleshoot Core Dump Generation

The core dump generation fails when RCM does not respond to the SIGUSR1 signal.

You can perform this procedure to generate the core dump:

1. Identify and kill the child process created for the core dump.

   The following is an example of core generation for the **rcm-configmgr** pod.

   ```
   $ kubectl exec -it rcm-configmgr-596dbcf76-8sff5 -- bash
   I have no name!@rcm-configmgr-596dbcf76-8sff5:/opt/workspace$ ps -elf
   F S UID          PID  PPID  C PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
   4 S 303            1     0  0  80   0 -   627 do_sig 09:43 ?        00:00:00
   /usr/bin/tini  /opt/workspace/rcm-configmgr
   4 S 303            6     1  0  80   0 - 653182 futex_ 09:43 ?        00:00:17
   /opt/workspace/rcm-configmgr
   1 S 303           34     6  0  80   0 - 616188 futex_ 09:52 ?        00:00:00
   /opt/workspace/rcm-configmgr
   4 S 303           42     0  0  80   0 -  1063 do_wai  10:50 pts/0     00:00:00 bash
   0 R 303           48    42  0  80   0 -  1476 -       10:50 pts/0     00:00:00 ps -
   elf
   I have no name!@rcm-configmgr-596dbcf76-8sff5:/opt/workspace$
   I have no name!@rcm-configmgr-596dbcf76-8sff5:/opt/workspace$ kill -9 34
   ```

   a. List all running processes on the system using the **`ps -elf`** command.

   b. Determine the child process to kill:

      1. Identify the process where the Parent Process ID (PPID) is 1. This is the original pod process.
         Also, note the corresponding PID.

         In this example, the corresponding PID is 6 for PPID 1.

      2. Identify the child process to dump core. The PPID of this process will be the PID of the original pod process.

Note the corresponding PID. In this example, the corresponding PID is 34 for PPID 6.

c. Kill the child process using the **`kill -9 <PID>`** command.

   Example: **`kill -9 34`**

2. Continue to send SIGUSR1 signal to dump core.

   `kubectl -n rcm exec -it <pod_name> -- kill -SIGUSR1 1`

   Example:

```
$ kubectl -n rcm exec -it rcm-configmgr-596dbcf76-8sff5 -- kill -SIGUSR1 1

$ coredumpctl

TIME                            PID   UID  GID SIG COREFILE  EXE

Wed 2024-06-04 10:52:03 UTC 7952309 303 303    6 present /opt/workspace/rcm-
configmgr
```

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.

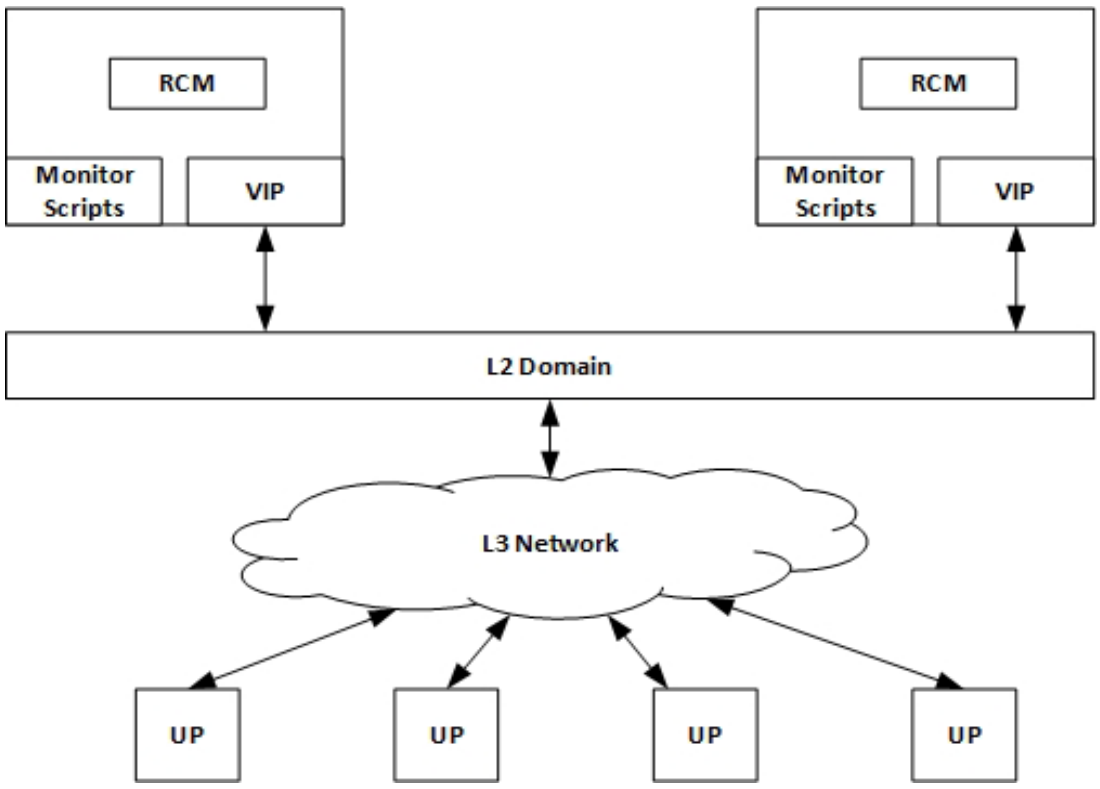
Architecture

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 (for example, 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.

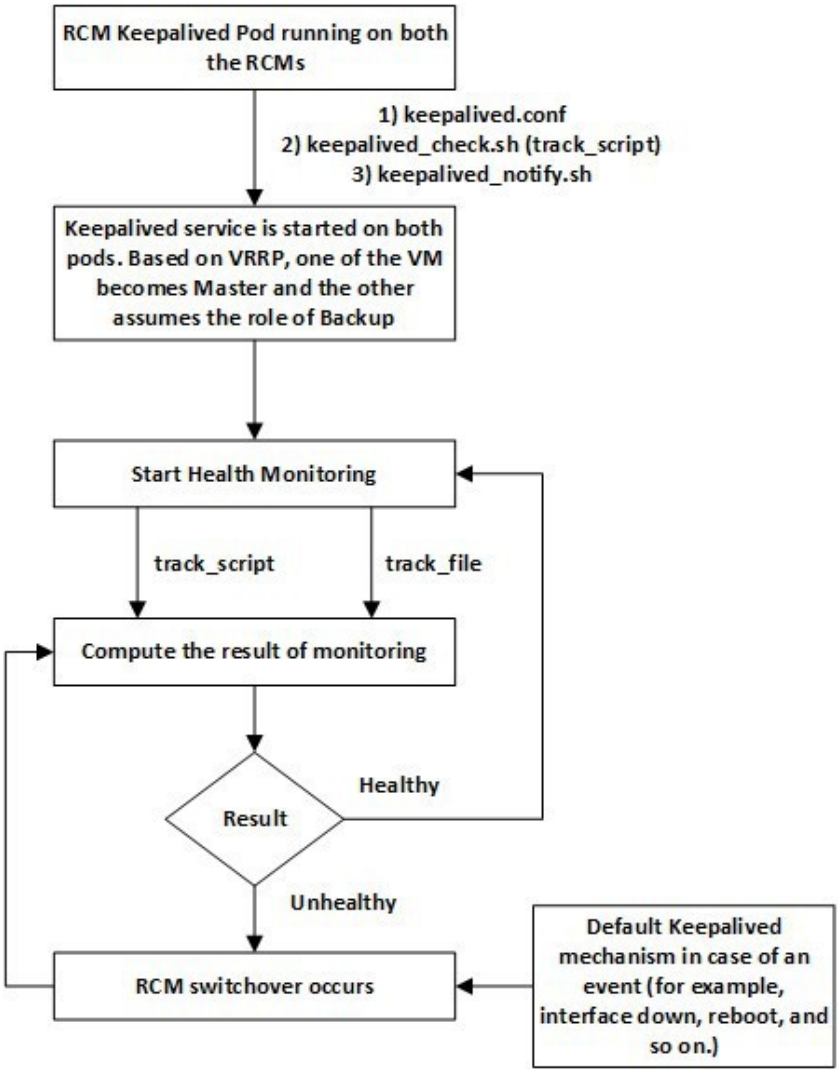
The RCM’s Controller Pod learns the state of the UPs and communicates the same to the ConfigMgr. The Controller Pod ensures consistency of operation and decides when/if to reboot a UP to restore the system to a healthy state. UP reboots are reserved for rare, corner case scenarios.

For double-fault scenarios, (e.g. when the switchover process is left in hung state and UP reboots itself after detecting it is 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.



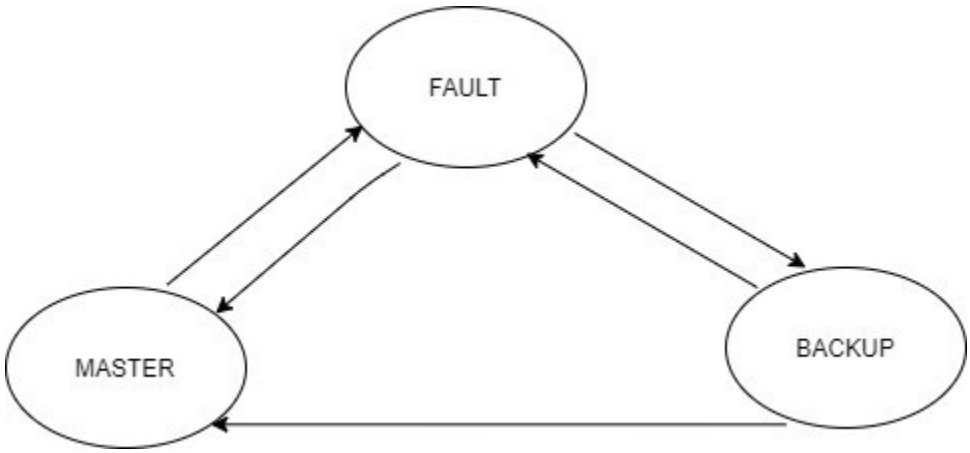
After the Keepalived pod is up and running on both the VMs, one of the RCM VMs comes up as Primary and another as Backup. Keepalived leverages the following files for proper operation:

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 an 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 21.24 and later releases:

rcm pause switchover *<true/false>* **[red-group** *red_group_number* **]**

Where:

- **red-group** *red_group_number* is an optional keyword/variable. The *red_group_number* starts from 1.

In releases prior to 21.24, 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 (for example, 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 is 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
   ipv4-route route_serial_number
      destination host_network_ipv4 mask ipv4_mask gateway host_ipv4 device
interface_name
   host priority priority
   host hostid host_id
   track-interface track_interface
   tuning-params vrrp-delay delay_timer
   init-container init-delay init_timer
   custom-port-keepalived port_number
   exit
```

NOTES:

- **vrrp interface** *vrrp_interface*: Specify the VRRP tracking interface.
- **vrrp routerId** *router_id*: Specify the VRRP router ID.

- **address** *vip_address*: Specify the Keepalived VIP IP address which must be the same in both RCM VMs.
- **mask** *address_mask*: Specify the VIP IP address mask.
- **broadcast** *broadcast_ipaddress*: Specify the VIP IP broadcast address.
- **device** *device_interface*: Specify the interface where VIP IP must be configured.
- **ipv4-route** *route_serial_number*: Specify the Keepalived IPv4 virtual routes.
- **priority** *priority*: Specify the VRRP host priority and must be different for both RCM VMs.
- **hostid** *host_id*: Specify the VRRP host ID.
- **track-interface** *track_interface*: Specify the Keepalived track interface.
- **init-container init-delay** *init_timer*: Specify the sleep time in the Init container of the Keepalived pod for startup delay. The default value is 90 seconds.
- **tuning-params vrrp-delay** *delay_time*: Specify the time in seconds to delay the start of the VRRP instance. *delay_time* must be an integer ranging from 1 to 86400.

  This command is not used to initiate startup delay; you must use the **init-container init-delay** CLI instead.
- **custom-port-keepalived** *port_number*: Specify the custom port number for the Keepalived endpoint.

Enabling Standalone RCM without Keepalived

Use the following command when you run RCM without keepalived to force MASTER status:

k8 smf profile rcm-config-ep ha-standalone { true | false }

NOTES:

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

Setting IPTables Rules for Keepalived

Set the iptables rules for Keepalived using the following 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.

Resetting Stale Connections on Standby RCM

When RCM enters the non-MASTER state, the RCM Configmgr pod does not restart in BACKUP state and cleans up its internal state. The RCM controller intimates the RCM Configmgr and Checkpointmgrs about this state transition.

Use the following command in Config mode to reset stale TCP connections from backup RCM CheckpointMgr to standby UPF:

```
k8 smf profile rcm-config-ep reset-stale-connection { true | false }
```

NOTES:

- When the command is enabled, the CheckpointMgr pods will terminate TCP connections with all UPFs and perform other clean-up when RCM enters the non-MASTER state. The CheckpointMgrs will not be restarted in BACKUP state.
- By default, the CLI command is set to **`false`**.

RCM IPSec

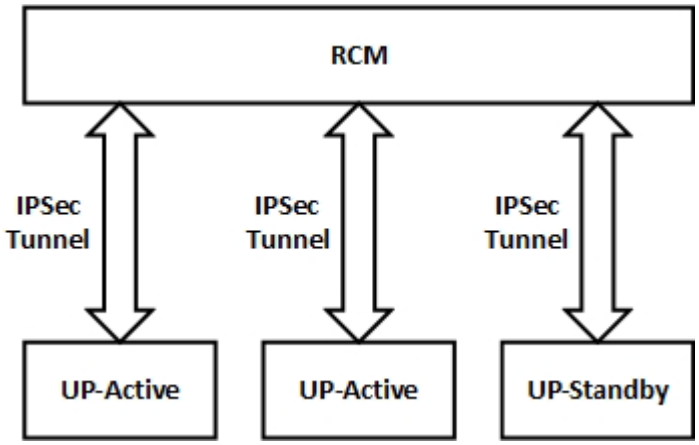
Feature Description

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 Ops Center 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)

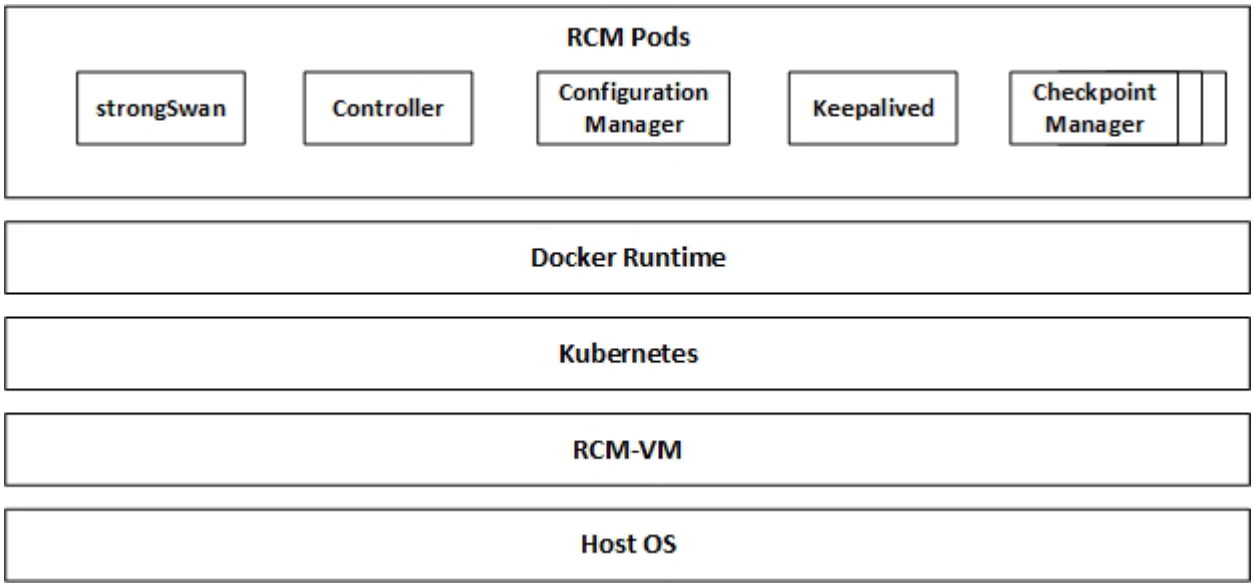
How it Works

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 **strongSwan** pod 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 Ops Center, 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.

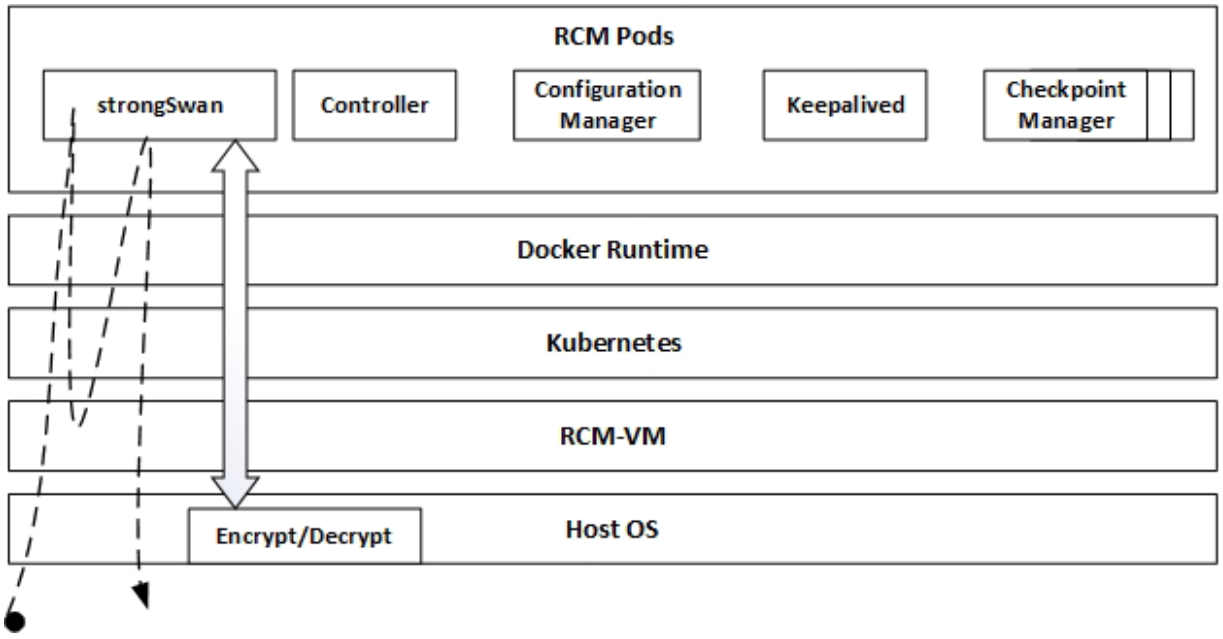


When RCM is deployed on bare metal, the RCM VM layer does not exist, and packet flow involves host kernel.

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:

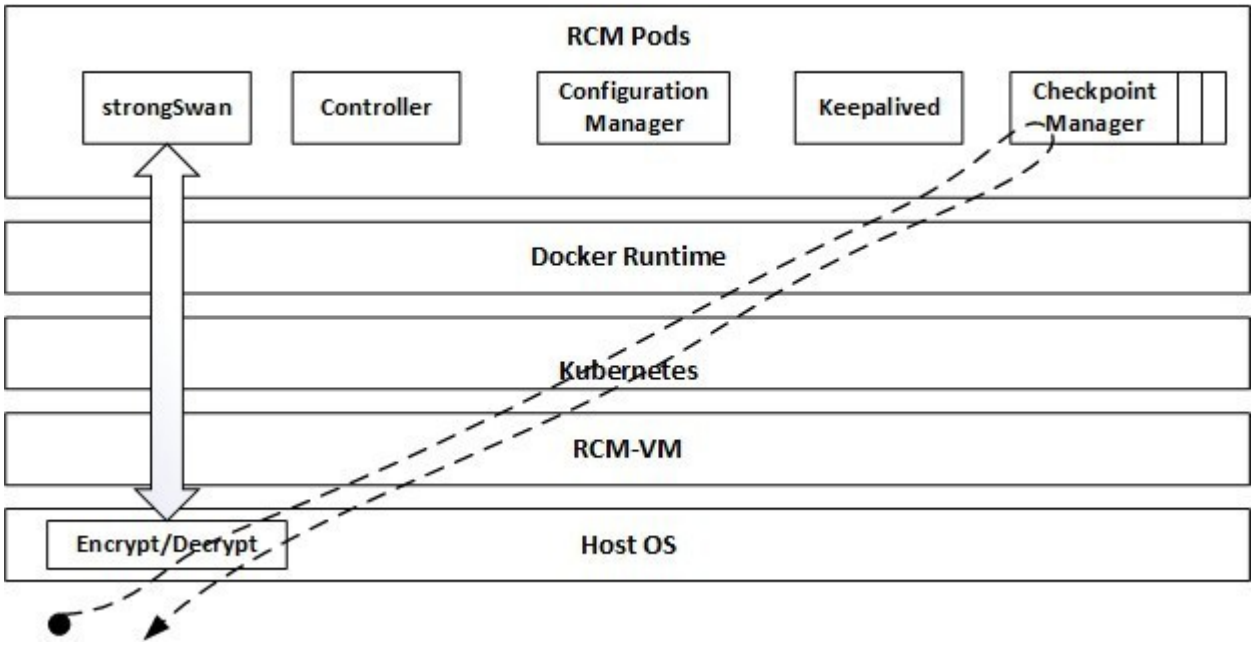
1. UP initiates IPSec tunnel establishment using IPSec control protocol such as IKEV1 or IKEV2. You may opt to choose only IKEV2.
2. Host kernel receives the packet.
3. The host kernel sends the packet to strongSwan.

   The above steps are repeated between UP and RCM till IPSec tunnel establishment concludes.
4. After the tunnel is established, strongSwan configures the host kernel with IPSec data plane parameters.
5. 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.

   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:

1. 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).
2. Host kernel receives the packet and decrypts it.
3. The host kernel sends the packet to desired checkpoint manager (essentially, this happens through Kubernetes networking).
4. Checkpoint manager processes the packet and generates a response (for example, TCP ACK).
5. The response packet is sent to the host kernel.
6. Host kernel encrypts and egresses the packet.

7. 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 and restrictions of the 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 ports are blocked for enhanced security. As part of RCM deployment, use the following steps to open the IPSec ports:

```
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
```

NOTES:

- 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 IPv4/IPv6/VIP addresses.
- **prf**: Specifies the Pseudo-Random Function (PRF) algorithm.
- **psk**: Specifies the Pre-Shared Key for peer authentication.
- **right-subnet**: [Mandatory] Specifies the RCM service IPv4/IPv6 address and port for UP.

NOTE: You can configure either the IPv4 or the IPv6 address. In the running configuration, only one address type can be used.

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.

Appendix A: Deployment Parameters

IMPORTANT: 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: 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"
   host 5 "config"
   host 6 "context <name>"
   host 7 "<LI config line1>"
   host 8 "<LI config line2>"
exit
svc-type nnrfsvc
redundancy-group 1
  host Active1
   host 2 "  config"
   host 3 "   context EPC2"
   host 4 "     nnrf-nfm-service svc-up1"
   host 5 "        bind ipv4-address  192.168.170.31"
   host 6 "         associate nnrf-mgmt-format prof1"   //Profile
configuration should be part of UP day-0 config
   host 7 "         endpoint-name end1"
   host 8 "       ipv4-address 192.168.170.32 portv4 8802"
   host 9 "    exit"
   host 10 "    end"
  exit
  host Active2
   host 2 "   config"
```

```
   host 3 "   context EPC2"
   host 4 "     nnrf-nfm-service svc-up2"
   host  5  "     bind  ipv4-address  192.168.170.33"
   host  6  "      associate  nnrf-mgmt-format  prof1"
   host 7 "         endpoint-name end1"
   host  8  "    ipv4-address  192.168.170.34  portv4  8802"
   host 9 "  exit"
   host  10  "  end"
  exit
 exit
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 • NNRF 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: Metrics

RCM Bfdmgr Controller Event Stats Category

bfdmgr_controller_event_stats

Description: Number of events sent to rcm-controller

Sample Query:

bfdmgr_controller_event_stats{service_name=\"bfdmgr\"}Labels:

- Label: `endpoint`

  Label Description: endpoint IP of UPF

  Example: 1.1.1.1

- Label: `groupid`

  Label Description: GroupId of the

  UPFExample: 1

RCM Bfdmgr UPF Event Stats Category

bfdmgr_upf_event_stats

Description: Number of events received for each UPF

Sample Query:

bfdmgr_upf_event_stats{service_name=\"bfdmgr\"}

Labels:

- Label: `endpoint`

  Label Description: endpoint IP of UPF

  Example: 1.1.1.1

- Label: `groupid`

  Label Description: GroupId of the

  UPFExample: 1

RCM checkpointmgr Current Emergency call count Category

chkptmgr_emergency_call_count

Description: Current Number of Emergency calls checkpointed to RCM

checkpointmgr Sample Query:

chkptmgr_emergency_call_count{service_name=\"rcm-checkpointmgr\"}

Labels:

- Label: `sessmgr_no`

  Label Description: Sessmgr Number of UP

  Example: 1

- Label: `up_address`

  Label Description: IP address of UP

  Example: 1.1.1.1

RCM checkpointmgr Current active VOLTE call count Category

chkptmgr_volte_active_call_count

Description: Current Number of volte active calls checkpointed to RCM

checkpointmgr Sample Query:

chkptmgr_volte_active_call_count{service_name=\"rcm-checkpointmgr\"}

Labels:

- Label: `sessmgr_no`

  Label Description: Sessmgr Number of UP

  Example: 1

- Label: `up_address`

  Label Description: IP address of UP

  Example: 1.1.1.1

RCM checkpointmgr Current call count Category

chkptmgr_total_call_count

Description: Current Number of calls checkpointed to RCM checkpointmgr

Sample Query: chkptmgr_total_call_count{service_name=\"rcm-checkpointmgr\"}Labels:

- Label: `sessmgr_no`
  Label Description: Sessmgr Number of UP
  Example: 1
- Label: `up_address`
  Label Description: IP address of UP
  Example: 1.1.1.1

RCM checkpointmgr Current non-active VOLTE call count Category

chkptmgr_volte_non_active_call_count

Description: Current Number of non-actove VOLTE calls checkpointed to RCM

checkpointmgr Sample Query:

chkptmgr_volte_non_active_call_count{service_name=\"rcm-checkpointmgr\"}

Labels:

- Label: `sessmgr_no`
  Label Description: Sessmgr Number of UP
  Example: 1
- Label: `up_address`
  Label Description: IP address of UP
  Example: 1.1.1.1

RCMcheckpointmgrCurrentnon-prioactivecallcountCategory

chkptmgr_non_prio_active_call_count

Description: Current Number of non-prio active calls checkpointed to RCM

checkpointmgr Sample Query:

chkptmgr_non_prio_active_call_count{service_name=\"rcm-checkpointmgr\"} Labels:

- Label: `sessmgr_no`

  Label Description: Sessmgr Number of UP

  Example: 1

- Label: `up_address`

  Label Description: IP address of UP

  Example: 1.1.1.1

RCM checkpointmgr Current non-prio non-active call count Category

chkptmgr_non_prio_non_active_call_count

Description: Current Number of non-prio non-active calls checkpointed to RCM

checkpointmgr Sample Query:

chkptmgr_non_prio_non_active_call_count{service_name=\"rcm-checkpointmgr\"}

Labels:

D Label: `sessmgr_no`

Label Description: Sessmgr Number of UP

Example: 1

- Label: `up_address`

  Label Description: IP address of UP

  Example: 1.1.1.1

RCM checkpointmgr UP Audit count Category

chkptmgr_up_audit_count

Description: Count related to UP Audits in RCM checkpointmgr

Sample Query: chkptmgr_up_audit_count{service_name=\"rcm-checkpointmgr\"}

Labels:

- Label: `sessmgr_no`
  Label Description: Sessmgr Number of UP
  Example: 1
- Label: `up_address`
  Label Description: IP address of UP
  Example: 1.1.1.1
- Label: `state`
  Label Description: Current
  stateExample: start/end

RCM checkpointmgr UP Flush completed count Category

chkptmgr_up_flush_completed_count

Description: Count related to UP flush completed in checkpointmgr

Sample Query: chkptmgr_up_flush_completed_count{service_name=\"rcm-checkpointmgr\"} Labels:

- Label: `sessmgr_no`
  Label Description: Sessmgr Number of UP
  Example: 1
- Label: `up_address`
  Label Description: IP address of UP

Example: 1.1.1.1

RCM checkpointmgr UP connection count Category

chkptmgr_up_connection_count

Description: Count related to UP connections to RCM checkpointmgr

Sample Query: chkptmgr_up_connection_count{service_name=\"rcm-

checkpointmgr\"}

Labels:

- Label: `sessmgr_no`
  Label Description: Sessmgr Number of UP
  Example: 1
- Label: `up_address`
  Label Description: IP address of UP
  Example: 1.1.1.1
- Label: `up_mode`
  Label Description: Mode in which UP is
  runningExample: active/standby
- Label: `conn_state`
  Label Description: Connection state of
  UP
  Example: connected/disconnected

RCM configmgr UP count Category

configmgr_upf_count

Description: Current count of UPs in a particular host group in rcm

configmgrSample Query: configmgr_upf_count{service_name=\"rcm-

configmgr\"} Labels:

- Label: `host_grp_name`

  Label Description: UP host group

  nameExample: any string

RCM configmgr config push complete count Category

configmgr_cfg_push_complete_count

Description: Count of config push complete in a particular group in rcm

configmgr Sample Query:

configmgr_cfg_push_complete_count{service_name=\"rcm-configmgr\"}

Labels:

- Label: `host_grp_name`

  Label Description: UP host group

  nameExample: any string

RCM configmgr config push failure count Category

configmgr_cfg_push_failure_count

Description: Count of config push failure in a particular host group in rcm

configmgr Sample Query:

configmgr_cfg_push_failure_count{service_name=\"rcm-configmgr\"}

Labels:

- Label: `host_grp_name`

  Label Description: UP host group

  nameExample: any string

RCM configmgr switchover abort count Category

configmgr_swover_abort_count

Description: Count of switchover aborts in a particular host group in rcm configmgr Sample Query: configmgr_swover_abort_count{service_name=\"rcm-configmgr\"} Labels:

- Label: `host_grp_name`
  Label Description: UP host group
  nameExample: any string

RCM configmgr switchover failure count Category

configmgr_swover_failure_count

Description: Count of switchover failures in a particular host group in rcm configmgr Sample Query: configmgr_swover_failure_count{service_name=\"rcm-configmgr\"}

Labels:

- Label: `host_grp_name`
  Label Description: UP host group
  nameExample: any string

RCM controller BFD heartbeat failure count Category

controller_bfd_hb_timeout_stats

Description: Count of total BFD heartbeat timeout received by rcm controller Sample Query: controller_bfd_hb_timeout_stats{service_name=\"rcm-controller\"}

RCM controller IPC received count Category

controller_ipc_rcvd_stats

Description: Count of total IPC message received by rcm controller

Sample Query: controller_ipc_rcvd_stats{service_name=\"rcm-controller\"} Labels:

- Label: `pod_name`
  Label Description: Name of pod with which controller is communicatingExample: rcm-configmgr or rcm-chkptmgr

RCM controller IPC sent count Category

controller_ipc_sent_stats

Description: Count of total IPC message sent from rcm controller

Sample Query: controller_ipc_sent_stats{service_name=\"rcm-controller\"}

Labels:

- Label: `pod_name`
  Label Description: Name of pod with which controller is communicatingExample: rcm-configmgr or rcm-chkptmgr

RCM controller IPC sent error count Category

controller_ipc_sent_error_stats

Description: Count of total IPC message sent error for rcm controller

Sample Query: controller_ipc_sent_error_stats{service_name=\"rcm-controller\"}

Labels:

- Label: `pod_name`

  Label Description: Name of pod with which controller is

  communicatingExample: rcm-configmgr or rcm-chkptmgr

RCM controller UPF BFD event count Category

controller_upf_bfd_event_stats

Description: Count of total BFD events received by rcm controller

Sample Query: controller_upf_bfd_event_stats{service_name=\"rcm-

controller\"}

Labels:

- Label: `grpId`

  Label Description: Group ID of UPF

  Example: any valid string

- Label: `endpoint`

  Label Description: Endpoint address of UPF
  Example: any ip address string

- Label: `event`

  Label Description: Name of bfd event
  Example: bfd_up or bfd_down

RCM controller UPF TCP connect count Category

controller_upf_tcp_connect_stats

Description: Count of total upf tcp connects received by rcm controller

Sample Query: controller_upf_tcp_connect_stats{service_name=\"rcm-

controller\"}

Labels:

- Label: `grpId`

  Label Description: Group ID of UPF

  Example: any valid string

- Label: `endpoint`

  Label Description: Endpoint address of UPF

  Example: any ip address string

RCM controller UPF TCP disconnect count Category

controller_upf_tcp_disconnect_stats

Description: Count of total upf tcp disconnects received by rcm controller

Sample Query: controller_upf_tcp_disconnect_stats{service_name=\"rcm-controller\"} Labels:

- Label: `grpId`

  Label Description: Group ID of UPF

  Example: any valid string

- Label: `endpoint`

  Label Description: Endpoint address of UPF

  Example: any ip address string

RCM controller UPF boot timer expiry count Category

controller_upf_boot_timer_expiry_stats

Description: Count of total UPF boot timer expiry in rcm controller

Sample Query: controller_upf_boot_timer_expiry_stats{service_name=\"rcm-controller\"}

Labels:

- Label: `endpoint`

  Label Description: Endpoint address of UPF

  Example: any ip address string

RCM controller UPF msg received count Category

controller_upf_msg_rcvd_stats

Description: Count of total message from UPF received by rcm controller

Sample Query: controller_upf_msg_rcvd_stats{service_name=\"rcm-

controller\"}

Labels:

- Label: `grpId`

  Label Description: Group ID of UPF

  Example: any valid string

- Label: `endpoint`

  Label Description: Endpoint address of UPF

  Example: any ip address string

- Label: `msg_type`

  Label Description: Name of msg sent/recvd

  Example: UpfMsgType_Registration, UpfMsgType_HB, UpfMsgType_State etc

RCM controller UPF msg sent count Category

controller_upf_msg_sent_stats

Description: Count of total message sent to UPF by rcm controller

Sample Query: controller_upf_msg_sent_stats{service_name=\"rcm-

controller\"}Labels:

- Label: `grpId`

Label Description: Group ID of UPF

Example: any valid string

- Label: `endpoint`

  Label Description: Endpoint address of UPF
  Example: any ip address string

- Label: `msg_type`

  Label Description: Name of msg sent/recvd

  Example: UpfMsgType_Registration, UpfMsgType_HB, UpfMsgType_State etc

RCM controller UPF registered count Category

controller_upf_registered_stats

Description: Count of total UPF registered in rcm controller

Sample Query: controller_upf_registered_stats{service_name=\"rcm-controller\"}Labels:

- Label: `grpId`

  Label Description: Group ID of UPF

  Example: any valid string

- Label: `endpoint`

  Label Description: Endpoint address of UPF
  Example: any ip address string

RCM controller last switchover duration Category

controller_last_switchover_seconds_total

Description: Duration of last switchover completion in rcm controller

Sample Query: controller_last_switchover_seconds_total{service_name=\"rcm-controller\"}

RCM controller mass UPF failure count Category

controller_mass_upf_failure_stats

Description: Count of mass UPF failures in rcm controller

Sample Query: controller_mass_upf_failure_stats{service_name=\"rcm-controller\"}

RCM controller switchover count Category

controller_switchover_stats

Description: Count of total switchover in rcm controller

Sample Query: controller_switchover_stats{service_name=\"rcm-

controller\"} Labels:

- Label: `swover_reason`

  Label Description: Reason string for switchover

  Example: BFD Failure, Planned Switchover etc
- Label: `state`

  Label Description: State of switchoverExample: started or completed

RCM controller switchover failure count Category

controller_switchover_failure_stats

Description: Count of failed switchover in rcm controller

Sample Query: controller_switchover_failure_stats{service_name=\"rcm-

controller\"}

Labels:

- Label: `failure_reason`

  Label Description: Reason string for failure

Example: Unknown Destination Endpoint <> , Pre-switchover Failed, Notification of switchover Trigger tostandby UP encountered error etc.

Example Expressions

The following is an example of an expression to get statistics of completed switchovers.

```
"targets": [
{
"expr": "sum (controller_switchover_stats{state=\"completed\"})", "interval":
"",
"legendFormat": "",
"refId": "A"
}
```

Whereas the following expression gives the total number of initiated switchovers:

```
"targets": [
{
"expr": "sum (controller_switchover_stats{state=\"started\"})", "interval": "",
"legendFormat": "",
"refId": "A"
}
```

The following expression gives the sum total of IPC sent to the controller for each pod, separately. That is, it displays two panels: First for IPC sent by ConfigMgr to the controller, and second for IPC sent by the CheckpointMgr to the controller:

```
"targets": [
{
"expr": "sum by (pod_name)(controller_ipc_sent_stats)", "interval": "",
"legendFormat": "",
"refId": "A"
}
```

Appendix E: 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 F: 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.

1. Copy the patch to */flash* of all the UPs.
2. To patch the plugin, execute the following CLI command:

   patch plugin p2p /flash/libp2p-*<plugin_filename>*
3. To install the plugin, execute the following CLI command:

   install plugin p2p patch_libp2p-*<plugin_filename>*
4. 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 G: 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: 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
```

NOTE: The script support for nnrf service type currently does exist. So, you must manually configure this service type.

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 H: RCM Deployment Known Issues

The following RCM deployment issue is seen in ESXI 6.7.0 with security patch ESXi-6.7.0-20210304001-standard.

RCA: k8s calico plugin fails to initialize on VMware deployment of the RCM VM.

Steps to Reproduce: VMware deployment of RCM VM

Fix Summary: Added the following MOP:

After the RCM VM deployment uses OVF, if the calico pods do not come up in 15-20 minutes, perform the following steps:

1. Configure the management IP and the default gateway. You can perform the same using net plan. Calico pods come up after this step.
2. Restart **`init_cluster service - systemctl restart init_cluster`**. Ops-center pods come up.