

AMF Rolling Software Upgrade

- Feature Summary and Revision History, on page 1
- Feature Description, on page 1
- Upgrading AMF, on page 2
- Rolling Upgrade Optimization, on page 15

Feature Summary and Revision History

Summary Data

Table 1: Summary Data

Applicable Product(s) or Functional Area	AMF
Applicable Platform(s)	SMI
Feature Default Setting	Disabled – Configuration required to enable
Related Documentation	Not Applicable

Revision History

Table 2: Revision History

Revision Details	Release
CDL 1.10.3 related updates.	2023.01.0
First introduced.	2022.01.0

Feature Description

The AMF consists of a three-tier architecture as the following:

- Protocol
- Service
- Session

Each tier from this list includes a set of microservices (pods) for a specific functionality. Within these tiers, a Kubernetes Cluster exists. It comprises K8s or Kubernetes nodes such as master node and worker node (which also includes OAM nodes).

For high availability and fault tolerance, each tier requires a minimum of two K8s worker nodes. Each worker node can have multiple replicas for each worker node. Kubernetes orchestrates the pods using the StatefulSets controller. The pods require a minimum of two replicas for fault tolerance.

The following is a list of 12 nodes in the AMF K8s cluster:

- Three master nodes
- Three OAM worker nodes
- Two protocol worker nodes
- Two service worker nodes
- Two session (data store) worker nodes

The K8s cluster supports the following nodes:

- OAM worker nodes—Hosts the Ops Center pods for configuration management and metrics pods for statistics and Key Performance Indicators (KPIs).
- Protocol worker nodes—Hosts the AMF protocol-related pods for the following interfaces:
  - Service-based interfaces (such as N8, N11, N12, N14, N15, NRF)
  - UDP-based protocol interfaces (such as N26)
- Service worker nodes—Hosts the AMF application-related pods that help in processing the perform session management.
- Session worker nodes—Hosts the database-related pods that store the data for the subscriber session.

Upgrading AMF

This section describes how to upgrade the rolling software for AMF.

Rolling Software Upgrade for AMF

The rolling software upgrade uses one of the following processes:

- Upgrading or migrating the build from an older version to a newer version
- Upgrading the patch for the required deployment set of application pods

Important When performing a fresh deployment of AMF 2024.03, it's mandatory to use CDL 1.11.8.

If upgrading an existing AMF deployment, it's recommended to upgrade the CDL to latest version before proceeding with the AMF upgrade.

For more information on the supported CDL versions, contact your Cisco account representative.

The applications must be available all the time, where:

- Any new version (or even multiple newer versions) is expected to get deployed with a new build version or patch.
- Any unstable deployment upgrade is reverted to a previous stable version.
- Rolling upgrade process gets activated with a zero downtime, by incrementally updating pod instances with new ones.

Note The rolling software upgrade is supported from an older version to a newer version within the same major release.

Prerequisites

The prerequisites for upgrading AMF must not have changes to the following functions:

- Set of features supported in the old and new builds
- Addition, deletion, or modification of the existing CLI behavior
- Interface changes within the peer or across the pods

Recommendations

The following is a list of recommendations:

- Configuration changes aren't recommended during the upgrade process.
- All the required configuration changes must be performed, when the upgrade process gets completed.

Failure Handling

It's recommended to use the manual process to downgrade the system to a previous healthy build. The following are some of the failure scenarios:

- Crash, pods deployment, and others during the processes
- New events or procedures after the successful upgrade

Rolling Software Upgrade Using the SMI Cluster Manager

The AMF software upgrade or in-service upgrade procedure uses the K8s rolling strategy to upgrade the pod images. The pods of a StatefulSet are upgraded sequentially to ensure that the ongoing process remains unaffected.

Initially, a rolling upgrade on a StatefulSet causes a single pod instance to terminate. A pod with an upgraded image replaces the terminated pod. This process continues until all the replicas of the StatefulSet are upgraded.

The terminating pods exit gracefully after completing all the ongoing processes. Other in-service pods continue to receive and process the traffic to provide a seamless software upgrade.

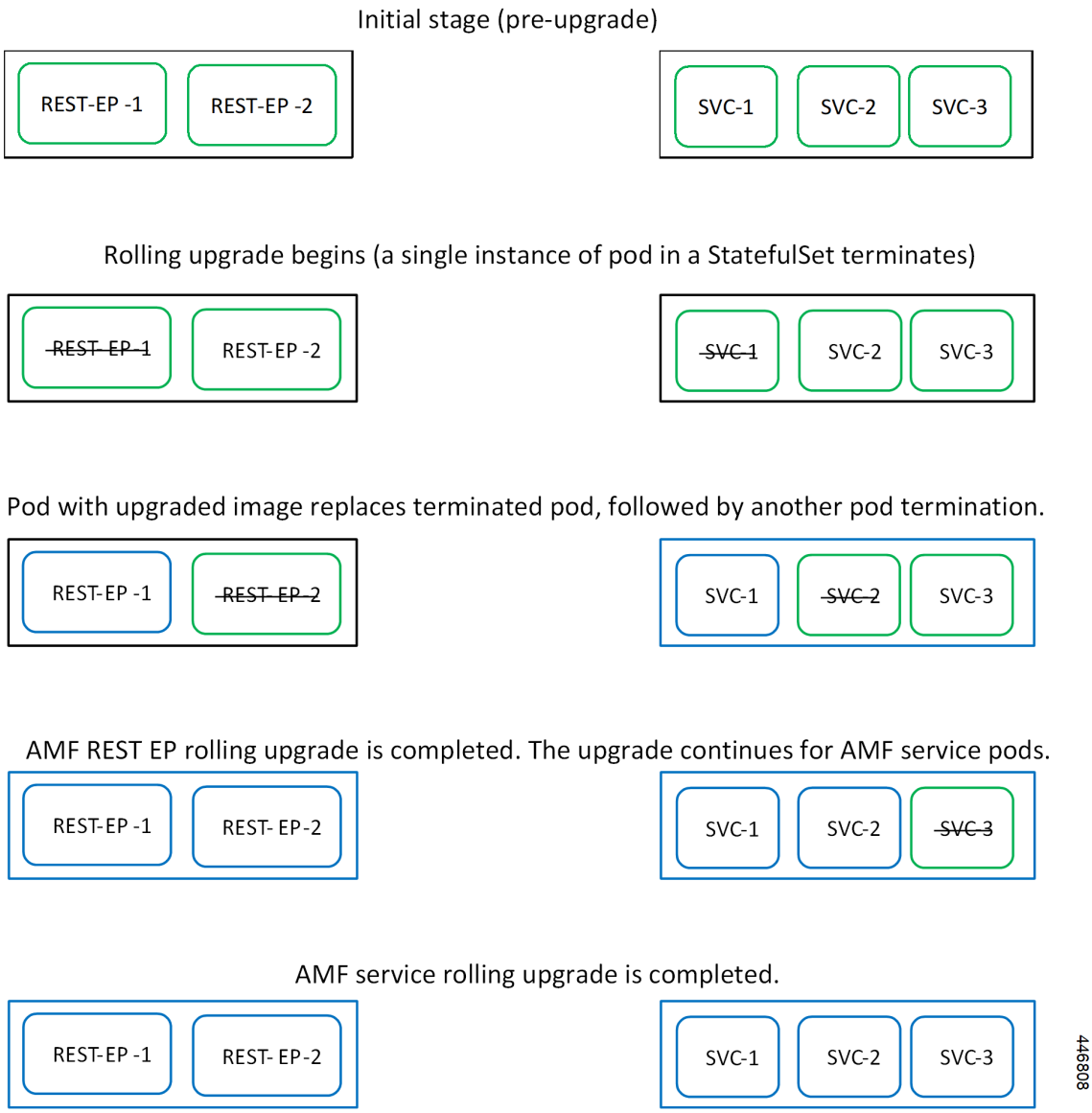
You can control the software upgrade process through the Ops Center CLI.

Note Each pod needs a minimum of two replicas for high availability. In a worst-case scenario, the processing capacity of the pod may briefly reduce to 50% while the software update is in progress.

The following figure illustrates the following:

- The AMF rolling upgrade for AMF REST endpoint pods (two replicas) on protocol worker nodes
- Along with AMF service pods (three replicas) on service worker nodes

Figure 1: AMF Rolling Upgrade



Important ETCD v3.5.x does not support in-service downgrade to 3.4.x. If you're downgrading from 2023.04.0 builds to previous releases. perform system mode shutdown before downgrade.

Prerequisites

The following is a list of prerequisites for updating AMF:

- All the nodes that include all the pods in the node must be up and running.
- A patch version of the AMF software

Note Currently, major versions don't support the rolling upgrade. The major version represents the release year, release number, and maintenance number. The version format is YYYY.RN.MN. For example: 2020.03.0

Important You can trigger rolling upgrade only when the CPU usage of the nodes is less than 50%.

AMF Health Check

To perform a health check and to ensure that all the services are running, and nodes are in the ready state:

- Log on to the Master node.
- Use the following configuration:

```
kubectl get pods -n smi
kubectl get nodes
kubectl get pod --all-namespaces -o wide
kubectl get pods -n amf-wsp -o wide
kubectl get pods -n cee-wsp -o wide
kubectl get pods -n smi-vips -o wide
helm list
kubectl get pods -A | wc -l
```

Important Ensure that all the nodes are in the ready state before you proceed further. Use the command `kubectl get nodes` to display the node states.

Performing the Deployment File Back Up

Before upgrading, back up the configuration, logs, and deployment files.

To back up the deployment files, perform the following steps:

1. Log on to the SMI Cluster Manager Node as a Ubuntu user
2. Create a new directory for deployment.

   Example:

   ```
   test@smiamf-cm01:~$ mkdir -p "temp_$(date +'%m%d%Y_T%H%M')" && cd "$_"
   ```

3. Move the `amf` deployment files into the newly created deployment directory.
4. Untar the `amf` deployment file.

   Example:

   ```
   test@smi1amf01-cm01:~/temp_08072019_T1651$ tar -xzvf amf.2020.01.0-1.SPA.tgz
   ./
   ./amf_REL_KEY-CCO_RELEASE.cer
   ```

```
./cisco_x509_verify_release.py
./amf.2020.01.0-1.tar
./amf.2020.01.0-1.tar.signature.SPA
./amf.2020.01.0-1.tar.SPA.README
```

5. Verify the downloaded image.

   Example:

```
test@smi1amf01-cm01:~/temp_08072019_T1651$ cat amf.2020.01.0-1.tar.SPA.README
```



Important Follow the procedure mentioned in the *SPA.README* file to verify the build before proceeding to the next step.

Performing the Ops Center Configuration Back Up

To back up the Ops Center configurations, perform the following steps:

1. Log on to SMI Cluster Manager node as an Ubuntu user
2. To back up the SMI Ops Center configuration to the `/home/ubuntu/smiops.backup` file, use the following command:

   ```
   ssh -p <port_number> admin@$(kubectl get svc -n smi | grep
   '.*netconf.*<port_number>' | awk '{ print $4 }') "show run | nomore"
   > smiops.backup_$(date +'%m%d%Y_T%H%M')
   ```

3. To back up the CEE Ops Center configuration to the `/home/ubuntu/ceeops.backup` file, use the following command:

   ```
   ssh admin@<cee-vip> "show run | nomore" > ceeops.backup_$(date
   +'%m%d%Y_T%H%M')
   ```

4. To back up the AMF Ops Center configuration to the `/home/ubuntu/amfops.backup` file, use the following command:

   ```
   ssh admin@<amf-vip> "show run | nomore" > amfops.backup_$(date
   +'%m%d%Y_T%H%M')
   ```

Performing CEE Back Up and AMF Ops Center Configuration

To back up the CEE and AMF Ops Center configuration, perform the following steps:

1. Log on to the Master node as an Ubuntu user
2. Create a directory to back up the configuration files as the following:

   ```
   mkdir backups_$(date +'%m%d%Y_T%H%M') && cd "$_"
   ```

3. Back up the AMF Ops Center configuration and verify the line count of the backup files as the following:

   ```
   ssh -p <port_number> admin@$(kubectl get svc -n $(kubectl get namespaces
    | grep -oP 'amf-(\d+|\w+)') | grep <port_number> | awk '{ print $3
   }') "show run | nomore" > amfops.backup_$(date +'%m%d%Y_T%H%M') && wc
    -l amfops.backup_$(date +'%m%d%Y_T%H%M')
   ```

   Example:

```
ubuntu@poamf-mas01:~/backups_09182019_T2141$ ssh -p 2024 admin@$(kubectl get svc -n
$(kubectl get namespaces | grep -oP 'amf-(\d+|\w+)') | grep <port_number> | awk '{ print
 $3 }') "show run | nomore" > amfops.backup_$(date +'%m%d%Y_T%H%M') && wc -l
amfops.backup_$(date +'%m%d%Y_T%H%M')
admin@<ipv4address>'s password: amf-OPS-PASSWORD
334 amfops.backup
```

4. Back up the CEE Ops Center configuration and verify the line count of the backup files as the following:

   ssh -p <port_number> admin@$(kubectl get svc -n $(kubectl get namespaces | grep -oP 'cee-(\d+|\w+)') | grep <port_number> | awk '{ print $3 }') "show run | nomore" > ceeops.backup_$(date +'%m%d%Y_T%H%M') && wc -l ceeops.backup_$(date +'%m%d%Y_T%H%M')

   Example:

   ```
   ubuntu@poamf-mas01:~/backups_09182019_T2141$ ssh -p <port_number> admin@$(kubectl get
   svc -n $(kubectl get namespaces | grep -oP 'cee-(\d+|\w+)') | grep <port_number> | awk
   '{ print $3 }') "show run | nomore" > ceeops.backup_$(date +'%m%d%Y_T%H%M') && wc -l
   ceeops.backup_$(date +'%m%d%Y_T%H%M')
   admin@<ipv4address>'s password: CEE-OPS-PASSWORD
   233 ceeops.backup
   ```

5. Move the SMI Ops Center backup file (from the SMI Cluster Manager) to the backup directory as the following:

   scp $(grep cm01 /etc/hosts | awk '{ print $1 }'):/home/ubuntu/smiops.backup_$(date +'%m%d%Y_T%H%M') .

   Example:

   ```
   ubuntu@poamf-mas01:~/backups_09182019_T2141$ scp $(grep cm01 /etc/hosts | awk '{ print
   $1 }'):/home/ubuntu/smiops.backup_$(date +'%m%d%Y_T%H%M') .
   ubuntu@<ipv4address>'s password: SMI-CM-PASSWORD
   smiops.backup                                                 100% 9346    22.3MB/s
      00:00
   ```

6. Verify the line count of the backup files.

   Example:

   ```
   ubuntu@poamf-mas01:~/backups_09182019_T2141$ wc -l *
     233 ceeops.backup
     334 amfops.backup
     361 smiops.backup
     928 total
   ```

Staging a New AMF Image

This section describes the procedure involved in staging a new AMF image before initiating the upgrade.

To stage the new AMF image, perform the following steps:

1. Download and verify the new AMF image.
2. Log on to the SMI Cluster Manager node as an Ubuntu user
3. Copy the images to the `uploads` directory.

   sudo mv <amf_new_image.tar> /data/software/uploads

Note The SMI uses the new image present in the `uploads` directory to upgrade.

4. Verify whether the image is picked up by the SMI for processing from the `uploads` directory.

   `sleep 30; ls /data/software/uploads`

   Example:

   ```
   ubuntu@poamf-cm01:~/temp_08072019_T1651$ sleep 30; ls /data/software/uploads
   ubuntu@poamf-cm01:~/temp_08072019_T1651$
   ```

5. Verify whether the images were successfully picked up and processed.

   Example:

   ```
   auser@unknown:$ sudo du -sh /data/software/packages/*
   1.6G /data/software/packages/cee.2019.07
   5.3G /data/software/packages/amf.2019.08-04
   16K /data/software/packages/sample
   ```

Note The SMI must unpack the images into the **packages** directory successfully to complete the staging.

Triggering the Rolling Software Upgrade

AMF utilizes the SMI Cluster Manager to perform a rolling software upgrade.

To upgrade AMF using SMI Cluster Manager, use the following configuration procedures:

Important Before you begin, ensure that the AMF is up and running with the current version of the software.

1. Log on to the SMI Cluster Manager Ops Center
2. Download the latest tarball from the URL, as the following:

   `software-packages download` *`url`*

   NOTES:

   - **software-packages download** *url*—Specify the software packages to be downloaded through HTTP/HTTPS.

   Example:

   ```
   SMI Cluster Manager# software-packages download <url>
   ```

3. Verify whether the tarball is loaded.

   `software-packages list`

   NOTES:

   - **software-packages list**—Specify the list of available software packages.

Example:

```
SMI Cluster Manager# software-packages list
[ amf-2019-08-21 ]
[ sample ]
```

4. Update the product repository URL with the latest version of the product chart.



Note If the repository URL contains multiple versions, the Ops Center automatically selects the latest version.

```
config
  cluster cluster_name
   ops-centers app_name instance_name
        repository url
         exit
       exit
```

NOTES:

- **cluster** *cluster_name*—Specify the K8s cluster name.
- **ops-centers** *app_name instance_name*—Specify the product Ops Center and instance.

  app_name is the application name.

  instance_name is the name of the AMF instance.
- **repository** *url*—Specify the local registry URL for downloading the charts.

Example:

```
SMI Cluster Manager# config
SMI Cluster Manager(config)# clusters test2
SMI Cluster Manager(config-clusters-test2)# ops-centers amf data
SMI Cluster Manager(config-ops-centers-amf/data)# repository <url>
SMI Cluster Manager(config-ops-centers-amf/data)# exit
SMI Cluster Manager(config-clusters-test2)# exit
```

5. Update the latest version of the product chart using the following command:

   clusters *cluster_name* **actions sync run**

   NOTES:

   - **actions**—Specify the actions performed on the cluster.
   - **sync run**—Triggers the cluster synchronization.

   Example:

```
SMI Cluster Manager# clusters test2 actions sync run
```

Important

- The cluster synchronization updates the AMF Ops Center, which in turn updates the application pods (through the **helm sync** command) one at a time automatically.
- When you trigger rolling upgrade on a specific pod, the AMF avoids routing new calls to that pod.
- The AMF honors in-progress calls by waiting for 30 seconds before restarting the pod where rolling upgrade is initiated. Also, the AMF establishes all the in-progress calls completely within 30 seconds during the upgrade period. The maximum call-setup time is 10 seconds.

Monitoring the Upgrade

You can monitor the status of the upgrade through SMI Cluster Manager Ops Center.

To monitor the upgrade status, use the following configurations:

```
config
  clusters cluster_name actions sync run debug true
  clusters cluster_name actions sync logs
  monitor sync-logs cluster_name
  clusters cluster_name actions sync status
  exit
```

NOTES:

- **clusters** *cluster_name*—Specify the information about the nodes to be deployed. *cluster_name* is the name of the cluster.
- **actions**—Specify the actions performed on the cluster.
- **sync run**—Trigger the cluster synchronization.
- **sync logs**—Display the current cluster synchronization logs.
- **sync status**—Display the current status of the cluster synchronization.
- **debug true**—Enter the debug mode.
- **monitor sync logs**—Monitor the cluster synchronization process.

Example:

```
SMI Cluster Manager# clusters test1 actions sync run
SMI Cluster Manager# clusters test1 actions sync run debug true
SMI Cluster Manager# clusters test1 actions sync logs
SMI Cluster Manager# monitor sync-logs test1
SMI Cluster Manager# clusters test1 actions sync status
```

Important

You can view the pod details after the upgrade through the CEE Ops Center.

For more information on pod details, see Viewing the Pod Details, on page 12 section.

Viewing the Pod Details

You can view the details of the current pods through the CEE Ops Center.

To view the pod details, use the following command in the CEE Ops Center CLI:

cluster pods *instance_name pod_name* **detail**

NOTES:

- **cluster pods**—Specify the current pods in the cluster.
- *instance_name*—Specify the name of the instance.
- *pod_name*—Specify the name of the pod.
- **detail**—Display the details of the specified pod.

The following example displays the details of the pod named *alertmanager-0* in the *amf-data* instance.

Example:

```
cluster pods amf-data alertmanager-0 detail
details apiVersion: "v1"
kind: "Pod"
metadata:
  annotations:
    alermanager.io/scrape: "true"
    cni.projectcalico.org/podIP: "<ipv4address/subnet>"
    config-hash: "5532425ef5fd02add051cb759730047390b1bce51da862d13597dbb38dfbde86"
  creationTimestamp: "2020-02-26T06:09:13Z"
  generateName: "alertmanager-"
  labels:
    component: "alertmanager"
    controller-revision-hash: "alertmanager-67cdb95f8b"
    statefulset.kubernetes.io/pod-name: "alertmanager-0"
  name: "alertmanager-0"
  namespace: "amf"
  ownerReferences:
  - apiVersion: "apps/v1"
    kind: "StatefulSet"
    blockOwnerDeletion: true
    controller: true
    name: "alertmanager"
    uid: "82a11da4-585e-11ea-bc06-0050569ca70e"
  resourceVersion: "1654031"
  selfLink: "/api/v1/namespaces/amf/pods/alertmanager-0"
  uid: "82aee5d0-585e-11ea-bc06-0050569ca70e"
spec:
  containers:
  - args:
    - "/alertmanager/alertmanager"
    - "--config.file=/etc/alertmanager/alertmanager.yml"
    - "--storage.path=/alertmanager/data"
    - "--cluster.advertise-address=$(POD_IP):6783"
    env:
    - name: "POD_IP"
      valueFrom:
        fieldRef:
          apiVersion: "v1"
          fieldPath: "status.podIP"
    image: "<path_to_docker_image>"
    imagePullPolicy: "IfNotPresent"
    name: "alertmanager"
```

```
    ports:
    - containerPort: 9093
      name: "web"
      protocol: "TCP"
    resources: {}
    terminationMessagePath: "/dev/termination-log"
    terminationMessagePolicy: "File"
    volumeMounts:
    - mountPath: "/etc/alertmanager/"
      name: "alertmanager-config"
    - mountPath: "/alertmanager/data/"
      name: "alertmanager-store"
    - mountPath: "/var/run/secrets/kubernetes.io/serviceaccount"
      name: "default-token-kbjnx"
      readOnly: true
  dnsPolicy: "ClusterFirst"
  enableServiceLinks: true
  hostname: "alertmanager-0"
  nodeName: "for-smi-cdl-1b-worker94d84de255"
  priority: 0
  restartPolicy: "Always"
  schedulerName: "default-scheduler"
  securityContext:
    fsGroup: 0
    runAsUser: 0
  serviceAccount: "default"
  serviceAccountName: "default"
  subdomain: "alertmanager-service"
  terminationGracePeriodSeconds: 30
  tolerations:
  - effect: "NoExecute"
    key: "node-role.kubernetes.io/oam"
    operator: "Equal"
    value: "true"
  - effect: "NoExecute"
    key: "node.kubernetes.io/not-ready"
    operator: "Exists"
    tolerationSeconds: 300
  - effect: "NoExecute"
    key: "node.kubernetes.io/unreachable"
    operator: "Exists"
    tolerationSeconds: 300
  volumes:
  - configMap:
      defaultMode: 420
      name: "alertmanager"
    name: "alertmanager-config"
  - emptyDir: {}
    name: "alertmanager-store"
  - name: "default-token-kbjnx"
    secret:
      defaultMode: 420
      secretName: "default-token-kbjnx"
status:
  conditions:
  - lastTransitionTime: "2020-02-26T06:09:02Z"
    status: "True"
    type: "Initialized"
  - lastTransitionTime: "2020-02-26T06:09:06Z"
    status: "True"
    type: "Ready"
  - lastTransitionTime: "2020-02-26T06:09:06Z"
    status: "True"
    type: "ContainersReady"
```

```
  - lastTransitionTime: "2020-02-26T06:09:13Z"
    status: "True"
    type: "PodScheduled"
  containerStatuses:
  - containerID: "docker://821ed1a272d37e3b4c4c9c1ec69b671a3c3fe6eb4b42108edf44709b9c698ccd"

    image: "<path_to_docker_image>"
    imageID: "docker-pullable://<path_to_docker_image>"
    lastState: {}
    name: "alertmanager"
    ready: true
    restartCount: 0
    state:
      running:
        startedAt: "2020-02-26T06:09:05Z"
  hostIP: "<host_ipv4address>"
  phase: "Running"
  podIP: "<pod_ipv4address>"
  qosClass: "BestEffort"
  startTime: "2020-02-26T06:09:02Z"
cee#
```

Rolling Upgrade Optimization

Table 3: Feature History

Feature Name	Release Information	Description
Rolling Upgrade Optimization	2024.03.0	AMF provides the following support for rolling upgrade optimization: • Retry mechanism at service and protocol pods during upgrades • Configuration-based rolling upgrade enhancements This optimization helps in reduced session and call events per second (CEPS) loss during the upgrade procedure. The configurable rolling upgrade enhancements enable smooth rollout of the changes. **Note** It is recommended that you enable the merged mode in the GTPC endpoint configuration to optimize the performance. Command introduced: • **supported-features [app-rx-retx-cache | app-tx-retx | rolling-upgrade-all | rolling-upgrade-enhancement-infra]** in AMF service configuration mode. • **interface n26 { vip-ip** *vip_ip_address* **}** in GTP endpoint configuration mode. Default Setting: Disabled – Configuration Required

Feature Description

AMF software version 2024.03.0 and higher supports rolling upgrade with additional optimizations. Rolling upgrade lets you perform graceful upgrade of all pods with minimal impact on sessions and CEPS.

This feature supports the following application-level enhancements:

- Retry mechanisms at protocol pods during service pods upgrade.
- Handling of transient sessions or transactions at service pods and protocol pods during their upgrades.
- Handling of topology and IPC mechanism changes to detect pods that are restarting or inactive. For inactive pods, the retry option is attempted toward other instances of pods.

Upgrading Software to Version with Rolling Upgrade Optimization Support

This section describes how to perform the rolling upgrade and to enable the rolling upgrade enhancements.



Important Review these important guidelines associated with the rolling upgrade procedure.

- Use the rolling upgrade optimization feature only when upgrading from Release 2024.03.0 or later.

  If you are upgrading to Release 2024.03.0 or later from a version earlier than Release 2024.03.0, first perform the shut-start upgrade.

- After the upgrade, enable the rolling upgrade enhancements using the CLI command. Then, the subsequent rolling upgrades to future releases includes the available optimizations.

Rolling Upgrade Considerations

To perform the rolling upgrade optmization, follow these steps:

1. Perform cluster synchronization through sync-phase ops-center to upgrade the HA AMF site to release 2024.03.
2. Apply the recommended configurations to enable the rolling upgrade enhancements.
   a. Go to Node 1 and log in to the AMF ops-center.
   b. Stop the ops-center.
   c. Enable the **rolling upgrade all** feature and **merge mode**.
   d. To enable the **rolling upgrade all** feature, use the following configuration.

```
config
   amf services service_name
   supported-features [ rolling-upgrade-all ]
   end
```

   e. To enable the **merge mode** feature, use the following configuration.

```
config
   instance instance-idinstance_id
      endpoint endpoint_name
         nodes nodes_count
         retransmission { timeout timeout_value | max-retryretry_count}
         internal-vip ip_address
         vip-ip ip_address
         interface interface_name
```

```
      vip-ip ip_address
    exit
```

3. Start the ops-center.

Rolling Upgrade Optimization Limitations

This feature has the following limitations:

- During a rolling upgrade, service pods restart one at a time, resulting in an uneven redistribution of sessions. The service pod that restarts first handles the highest number of sessions, while the service pod that restarts last handles the fewest. This redistribution can temporarily increase the memory requirements for some service pods. The system returns to normal operation once the sessions are removed from the local cache of the service pods.
- During the rolling upgrade, ongoing procedures in service pods continue, but a best-effort mechanism is in place to ensure their successful completion.

Configuring the Supported Features for Rolling Upgrade

To enable the supported features for a rolling upgrade, use the following sample configuration:

```
config
  amf-services service_name
    supported-features [ app-rx-retx-cache | app-tx-retx |
rolling-upgrade-all | rolling-upgrade-enhancement-infra ]
    end
```

NOTES:

- **supported-features [app-rx-retx-cache | app-tx-retx | rolling-upgrade-all | rolling-upgrade-enhancement-infra]**: Specify one of the following options to enable the supported features for the rolling upgrade.
  - **app-rx-retx-cache**: Enable retransmission cache for inbound messages at application.
  - **app-tx-retx**: Enable retransmission for outbound messages at application.
  - **rolling-upgrade-all**: Enable all the rolling upgrade features that are available through **rolling-upgrade-enhancement-infra**, **app-rx-retx-cache**, and **app-tx-retxrolling** keyword options. By default, the rolling upgrade features are disabled.

    rolling-upgrade-all is the only recommended option.
  - **rolling-upgrade-enhancement-infra**: Enable infra-level features.



Important

- It is recommended that you do not enable or disable the rolling upgrade features at run time to prevent an impact on the existing sessions.
- It is highly recommended that you use only the **rolling-upgrade-all** option as all the other command options are available only for debugging purpose.

Verifying Rolling Upgrade Optimization

Use the **show running-config amf-services supported-features** command to verify the supported features for a rolling upgrade.

The following is an example output of the **show running-config amf-services supported-features** command.

```
show running-config amf-services supported-features
amf-services am1
   supported-features [ rolling-upgrade-all ]
exit
```