Oracle® Communications

Cloud Native Core, Certificate Management Installation, Upgrade, and Fault Recovery Guide

Release 24.3.0
G10418-02
November 2024

Oracle Communications Cloud Native Core, Certificate Management Installation, Upgrade, and Fault Recovery Guide, Release 24.3.0

G10418-02

Contents

My Oracle Support

My Oracle Support (https://support.oracle.com) is your initial point of contact for all product support and training needs. A representative at Customer Access Support can assist you with My Oracle Support registration.

Call the Customer Access Support main number at 1-800-223-1711 (toll-free in the US), or call the Oracle Support hotline for your local country from the list at http://www.oracle.com/us/support/contact/index.html. When calling, make the selections in the sequence shown below on the Support telephone menu:

1. Select **2** for New Service Request.
2. Select **3** for Hardware, Networking and Solaris Operating System Support.
3. Select one of the following options:
   - For Technical issues such as creating a new Service Request (SR), select **1**.
   - For Non-technical issues such as registration or assistance with My Oracle Support, select **2**.

You are connected to a live agent who can assist you with My Oracle Support registration and opening a support ticket.

My Oracle Support is available 24 hours a day, 7 days a week, 365 days a year.

Acronyms

The following table lists the acronyms and the terminologies used in the document:

Table Acronyms

Acronym	Description
3GPP	3rd Generation Partnership Project
API	Application Programming Interface
CCA	Client Credentials Assertions
CMP	Certificate Management Protocol
CNC	Cloud Native Core
CNC Console	Cloud Native Configuration Console
OCCM	Oracle Communications Certificate Management
CA	Certification Authority is a trusted entity that issues Secure Sockets Layer (SSL) certificates. CAs are also called issuer in this document.
DNS	Domain Name Server
EE	End Entity
ECC	Elliptic Curve Cryptography
HSM	Hardware Security Module
IDP	Identity Provider
PKI	Public Key Infrastructure
RA	Registration Authority
RSA	Rivest-Shamir-Adleman
SAN	Subject Alternative Name
URI	Uniform Resource Indicator
URN	Uniform Resource Name
CMP Identity Key	Private Key used by Certificate Management to sign the CMPv2 requests and establish trust between Certificate Management and CA.
CMP Identuty Certificate	Certificate that corresponds to and certifies the CMP Identity Key. It is included in the CMPv2 requests for authentication by CA.

What's New in This Guide

This section introduces the documentation updates for Release 24.3.x.

Release 24.3.0 - G10418-02, November 2024

- **General Updates:**
  - Updated the Premier Support Details table with the release information in the Oracle Error Correction Policy section.

Release 24.3.0 - G10418-01, October 2024

- **General Updates:**
  - Updated the release number to 24.3.0 in the entire document.
- **Installation Updates:**
  - Updated the software versions in Software Requirements section.
  - Updated the architecture description and diagram to include the monitoring secrets funtionality in the OCCM Architecture section.
  - Updated the diagrams for all deployment models to include the monitoring secrets functionality in the OCCM Deployment Models section.
  - Added the OCCM Configuration Max Limits table to describe the the maximum limits defined in OCCM configuration in the OCCM Deployment Models section
  - Added a note with the annotation to enable egress network attachment for OCCM to CA communication in the Installing OCCM Package section.
  - Added the following parameters in the Configuration Options section:
    - Added the `occmConfig.cmp.config.tls.minProtocol` parameter to set the minimum supported TLS version.
    - Added the `occmConfig.cmp.config.tls.tlsNamedGroups` parameter to set the supported groups. For clients, the groups are sent using the supported groups extension.
    - Added the `occmConfig.cmp.config.tls.cipherStrings` parameter to set the available ciphers for TLSv1.2 and below.
    - Added the `occmConfig.cmp.config.tls.cipherSuites` parameter to set the available cipher suites for TLSv1.3.
    - Added the `occmConfig.cmp.config.tls.clientSignatureSchemes` parameter to set the supported signature algorithms for TLSv1.2 and TLSv1.3.
    - Added the `occmConfig.k8sSecretMonitoring` parameter to enable or disable the secret monitoring feature.
- **Upgrade, Rollback, and Uninstall Updates:**
  - Added a note in the Upgrading OCCM section to back up the OCCM configmap before performing an upgrade or rollback.
  - Updated the Supported Upgrade Paths.
  - Added a note in the Rolling Back OCCM section to back up the OCCM configmap before performing an upgrade or rollback.
  - Updated the Supported Rollback Paths.

1 Introduction

Oracle Communications Certificate Management (OCCM) is an automated solution for managing the certificates needed for Oracle 5G Network Functions (NFs). OCCM constantly monitors and renews the certificates based on their validity or expiry period.As 3GPP recommends using separate certificates based on the client or server mode and the type of workflow, automated certificate management eliminates any possibilities of network disruption due to expired certificates. In SBA network deployments, the Network Functions (NFs) are required to support multiple operator certificates for different purposes and interfaces. This amounts to hundreds of certificates in the network with varying validity periods and unwieldy to monitor and renew the certificates manually. Hence, automation of certificate management becomes important to avoid network disruptions due to expired certificates.

This guide describes how to install or upgrade Oracle Communications Certificate Management (OCCM) in a cloud native environment. It also includes information on performing fault recovery for OCCM.

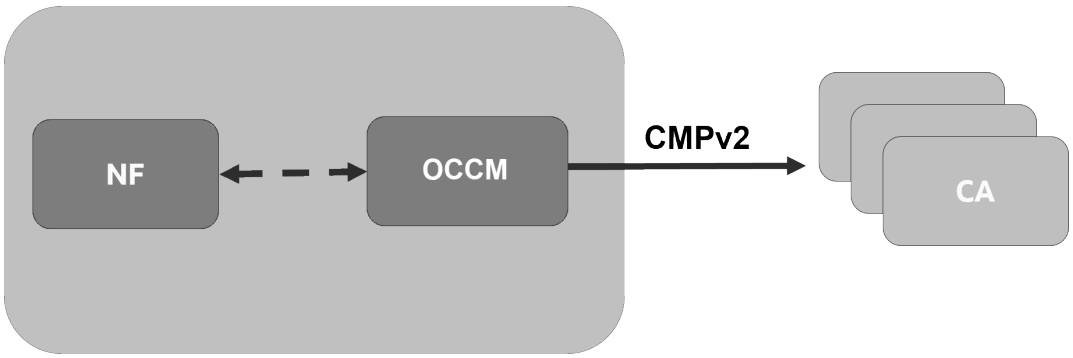
> **Note**
>
> This guide covers the installation instructions when Podman is the container platform with Helm as the Packaging Manager. For any other container platform, the operator must use the commands based on their deployed container runtime environment.

1.1 Overview

OCCM integrates with the Certificate Authority(s) using Certificate Management Protocol Version 2 (CMPv2) and RFC4210 to facilitate these certificate management operations:

- Operator-initiated certificate creation
- Operator-initiated certificate recreation
- Automatic certificate monitoring and renewal

Figure 1-1 OCCM Integration with CA



OCCM supports transport of CMPv2 messages using HTTP-based protocol.

OCCM provides the following mechanisms to establish initial trust between OCCM and CA(s):

1. Certificate-based message signing
2. Pre-shared key or MAC based authentication

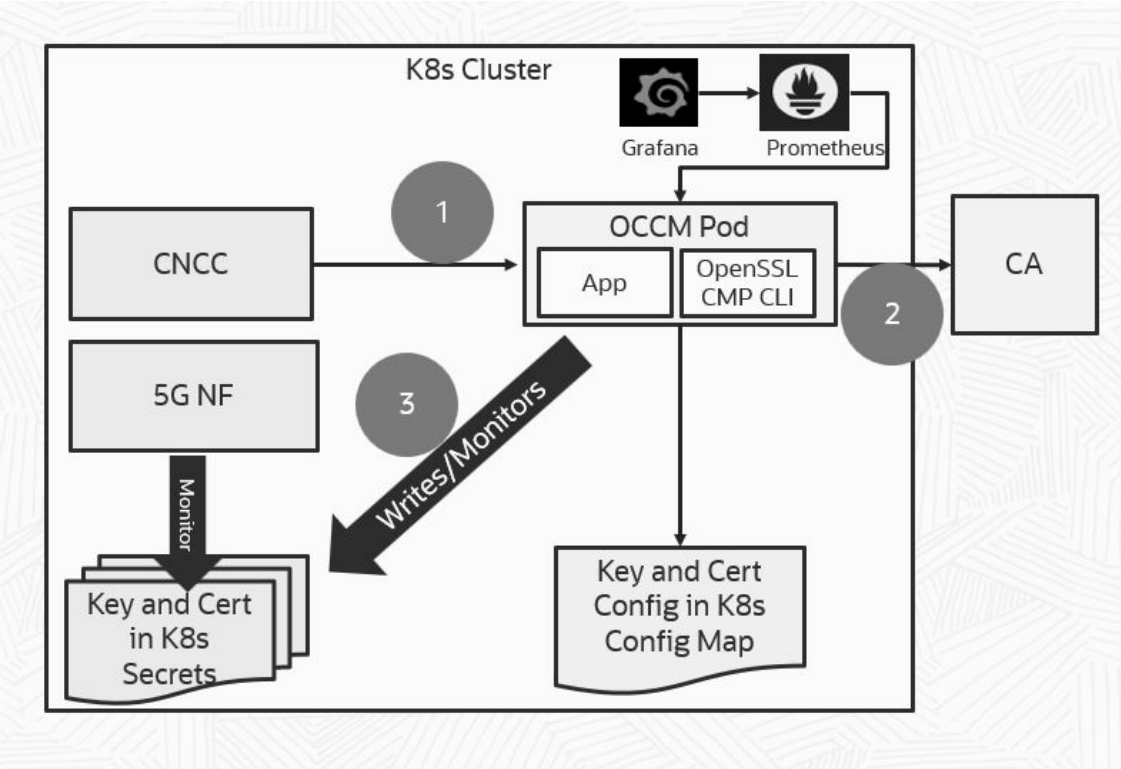
All the subsequent CMPv2 procedures are authenticated using the certificate-based mechanism in compliance with 3GPP TS 33.310.

The keys and X.509 certificates are managed using Kubernetes secrets.

1.2 OCCM Architecture

OCCM is a Cloud Native application consisting of a single microservice. OCCM is packaged and delivered as a CSAR or Helm chart.

Figure 1-2 OCCM Architecture



Architecture Description

OCCM is deployed as a single Kubernetes Pod and has a small resource footprint. The OCCM application uses a set of OpenSSL Certificate Management Protocol (CMP) CLI commands based on the provided configuration and the certificate management procedure that needs to be carried out at a point in time. The Output – Key and Certificate – is stored in configuration defined Kubernetes secret.

Operator provides the desired key and certificate configuration through Console. OCCM contacts the CA for certificate signing. After successful Certificate creation, OCCM writes the key and certificate in Kubernetes secrets.

In the diagram above:

1. Operator provides the desired Key and Certificate configuration.
2. OCCM contacts the CA for certificate signing.
3. OCCM writes the key and certificate in Kubernetes Secrets. Starts monitoring of the secret for modification or deletion.

OCCM provides the following deployment models to support certificate management for the integrated NF(s) instantiated within the same cluster:

- Dedicated deployment model - OCCM resides in the same Kubernetes namespace as the NF or Components.
- Shared deployment model - OCCM is deployed in a separate Kubernetes namespace and can manage certificates of multiple NFs or components deployed in other Kubernetes namespaces.

Appropriate permissions must be assigned to OCCM using Kubernetes Service Account, Role and Role Binding, based on the selected deployment model.

OCCM provides secret monitoring capabilities, which help the operator to monitor and manage previously created certificates. OCCM identifies and takes necessary action if certificates are modified or deleted manually, without experiencing loss of service.

Certificate monitoring is useful in the following scenarios:

- The certificate or the Kubernetes secret holding the certificate is deleted.
- The certificate is manually updated.

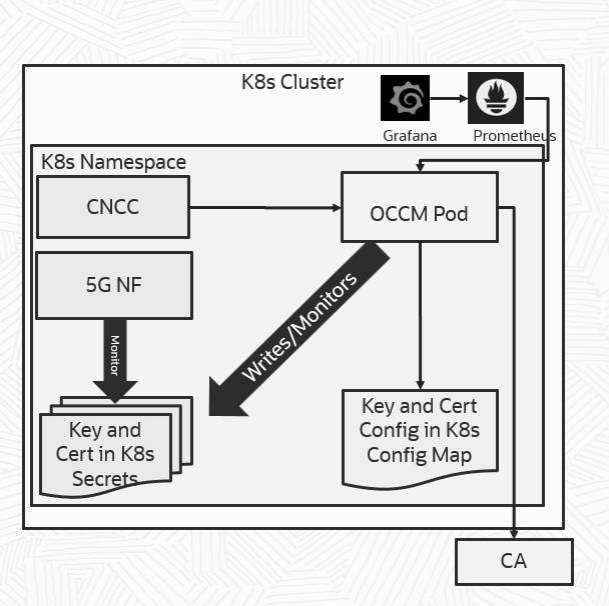
For more information, see "Monitoring Secrets for Manual Update or Delete" in the *Oracle Communications Cloud Native Configuration Console User Guide.*

1.3 OCCM Deployment Models

The following deployment models are supported by OCCM:

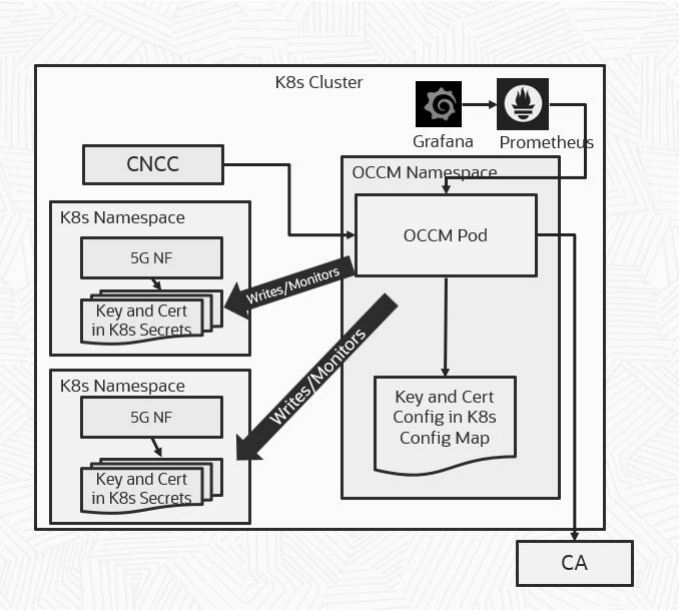
- Dedicated deployment model - OCCM resides in the same Kubernetes namespace as the NF or components.
- Shared deployment model - OCCM is deployed in a separate Kubernetes namespace and can manage certificates of multiple NFs or components deployed in other Kubernetes namespaces.

Figure 1-3 Deployment Model-1: Dedicated OCCM Deployment – Single Namespace Management



In the dedicated OCCM deployment model, OCCM is deployed in the same namespace as the component(s) managed by it.

Figure 1-4 Deployment Model-2: Shared OCCM Deployment - Multiple Namespaces Management



In the shared deployment model, OCCM is deployed in a separate Kubernetes namespace and can manage certificates of multiple NFs or components deployed in other Kubernetes namespaces. OCCM provides Key and certificate for NFs running in multiple namespaces. OCCM needs privileges to be able to create or write Kubernetes secrets in multiple

namespaces. Care must be taken to not provide Kubernetes Cluster Role access as it provides access to all namespaces in the cluster. Multiple namespace specific roles are created and then bind to OCCM service account using role binding. OCCM is deployed in a dedicated namespace different from the components' namespace managed by it. For more information, see Creating Service Account, Role, and Rolebinding.

OCCM Configuration Maximum Limits

Following table describes the maximum limits defined in OCCM configuration:

Table 1-1 OCCM Configuration Maximum Limits

Attribute Name	Maximum Limit
Maximum Number of Certificate Configuration Supported	100
Maximum Number of Issuer Configuration Supported	30
Maximum Number of Unique Namespace Configuration Supported	10

1.4 Reference

Refer to the following documents for more information:

- *Oracle Communications Cloud Native Configuration Console User Guide*
- *Oracle Communications Cloud Native Configuration Console Installation, Upgrade, and Fault Recovery Guide*
- *Oracle Communications Cloud Native Core Security Guide*
- *Oracle Communications Cloud Native Core Solution Upgrade Guide*

1.5 Oracle Error Correction Policy

The table below outlines the key details for the current and past releases, their Generally Available (GA) dates, the latest patch versions, and the end dates for the Error Correction Grace Period.

Table 1-2 Premier Support Details

Release Number	General Availability (GA) Date	Latest Patch Version	Error Correction Grace Period End Date
3.24.3	October 2024	24.3.0	October2025
3.24.2	October 2024	24.2.1	July 2025
3.24.1	April 2024	24.1.0	April 2025
3.23.4	July 2024	23.4.3	July 2025

> **Note**
>
> For a release, Sev1 and Critical Patch Unit (CPU) patches are supported for 12 months. For more information, see Oracle Communications Cloud Native Core and Network Analytics Error Correction Policy.

1.6 Oracle Open Source Support Policies

Oracle Communications Cloud Native Core uses open source technology governed by the Oracle Open Source Support Policies. For more information, see Oracle Open Source Support Policies.

2
Installing OCCM

This chapter provides information about installing OCCM in a cloud native environment.

2.1 Prerequisites

Before installing and configuring OCCM, ensure that the following prerequisites are met.

2.1.1 Software Requirements

This section lists the added or updated software required to install OCCM release 24.3.x. For more information about software requirements, see *Oracle Communications Cloud Native Core Certificate Management Installation, Upgrade, and Fault Recovery Guide.*

Install the following software before installing OCCM:

Table 2-1 Software Requirements

Software	Version
Helm	3.13.2
Kubernetes	1.30.x, 1.29.x, 1.28.x
Podman	4.4.1

To check the current Helm and Kubernetes version installed in the CNE, run the following commands:

```
kubectl version
```

```
helm version
```

The following software are available if OCCM is deployed in CNE. If you are deploying OCCM in any other cloud native environment, these additional software must be installed before installing OCCM. To check the installed software, run the following command:

```
helm ls -A
```

The following table lists the versions of additional software along with the usage:

Table 2-2 Additional Software

Software	Version
containerd	1.7.13
Calico	3.26.3
MetalLB	0.14.4

Table 2-2 (Cont.) Additional Software

Software	Version
Prometheus	2.51.1
Grafana	9.5.3
Jaeger	1.52.0
Istio	1.18.2
Kyverno	1.9.0
cert-manager	1.12.4
Oracle OpenSearch	2.11.0
Oracle OpenSearch Dashboard	2.11.0
Fluentd OpenSearch	1.16.2
Velero	1.12.0

2.1.2 Environment Setup Requirements

This section provides information on environment setup requirements for installing OCCM.

Client Machine Requirement

This section describes the requirements for client machine, that is, the machine used by the user to run deployment commands.

Client machine must have:

- Helm repository configured.
- network access to the Helm repository and Docker image repository.
- network access to the Kubernetes cluster.
- required environment settings to run the `kubectl`, `docker`, and `podman` commands. The environment must have privileges to create a namespace in the Kubernetes cluster.
- Helm client installed with the push plugin. Configure the environment in such a manner that the `helm install` command deploys the software in the Kubernetes cluster.

Network Access Requirement

The Kubernetes cluster hosts must have network access to the following repositories:

- Local helm repository, where the OCCM helm charts are available.
  To check if the Kubernetes cluster hosts have network access to the local helm repository, run the following command:

  ```
  helm repo update
  ```

- Local docker image repository: It contains the OCCM Docker images.

To check if the Kubernetes cluster hosts can access the the local Docker image repository, try to retrieve any image with tag name to check connectivity by running the following command:

```
docker pull <docker-repo>/<image-name>:<image-tag>
```

```
podman pull <Podman-repo>/<image-name>:<image-tag>
```

where:

- `<docker-repo>` is the IP address or host name of the repository.
- `<Podman-repo>` is the IP address or host name of the Podman repository.
- `<image-name>` is the docker image name.
- `<image-tag>` is the tag the image used for the OCCM pod.

> **Note**
>
> Run the `kubectl` and `helm` commands on a system based on the deployment infrastructure. For instance, they can be run on a client machine such as VM, server, local desktop, and so on.

Server or Space Requirement

For information about the server or space requirements, see the *Oracle Communications Cloud Native Core, Cloud Native Environment Installation, Upgrade, and Fault Recovery Guide.*

CNE Requirement

This section is applicable only if you are installing OCCM on Cloud Native Environment (CNE).

To check the CNE version, run the following command:

```
echo $OCCNE_VERSION
```

For more information about CNE, see *Oracle Communications Cloud Native Core, Cloud Native Environment Installation, Upgrade, and Fault Recovery Guide.*

CNC Console Requirements

OCCM supports CNC Console.

For more information about CNC Console, see *Oracle Communications Cloud Native Configuration Console Installation, Upgrade, and Fault Recovery Guide* and *Oracle Communications Cloud Native Configuration Console User Guide.*

2.1.3 Resource Requirements

This section lists the resource requirements to install and run OCCM.

Resource Profile

Table 2-3 Resource Profile

Microservice Name	Pod Replica	Limits			Requests		
		CPU	Memory (Gi)	Ephimeral Storage (Mi)	CPU	Memory (Gi)	Ephimeral Storage (Mi)
OCCM	1	2	2	1102	2	2	57
Helm Test	1	0.5	0.5	1102	0.5	0.5	57
Total		**2.5**	**2.5**	**2204**	**2.5**	**2.5**	**114**

> **Note**
>
> - Helm Test Job - This job is implemented on demand when Helm test command is run. It will run the Helm test and stops after completion. These are short-lived jobs that get terminated after the work is done. So, they are not part of active deployment resource but need to be considered only during Helm test procedures.
> - Troubleshooting Tool (Debug tool) Container - If Troubleshooting Tool Container Injection is enabled during OCCM deployment or upgrade, this container will be injected to OCCM pod. These containers will stay till pod or deployment exists. Debug tool resources are not considered in the calculation. Debug tool resources usage is per pod. If debug tool is enabled, then max 0.5vCPU and 0.5Gi memory per OCCM pod is needed.

2.2 Installation Sequence

This section describes preinstallation, installation, and postinstallation tasks for OCCM.

> **Note**
>
> In case of fresh installation of OCCM and NF on a new cluster, you must follow the following sequence of installation:
>
> 1. OCCM
> 2. cnDBTier
> 3. CNC Console
> 4. NF

2.2.1 Preinstallation Tasks

Before installing OCCM perform the tasks described in this section.

2.2.1.1 Downloading the OCCM Package

To download the OCCM package from My Oracle Support (MOS), perform the following steps:

1. Log in to My Oracle Support using your login credentials.
2. Click the **Patches & Updates** tab to locate the patch.
3. In the Patch Search Certificate Management, and click the **Product or Family (Advanced)** option.
4. Enter `Oracle Communications Cloud Native Core - 5G` in Product field and select the product from the **Product** drop-down list.
5. From the Release drop-down list, select **"Oracle Communications Cloud Native Core Certificate Management <release_number>".**
   Where, <release_number> indicates the required release number of OCCM.
6. Click **Search**.
   The Patch Advanced Search Results list appears.
7. Select the required patch from the list.
   The Patch Details window appears.
8. Click **Download**.
   The File Download window appears.
9. Click on the `<p********_<release_number>_Tekelec>.zip` file.
10. Extract the release package zip file to download the network function patch to the system where network function must be installed.
11. Extract the release package zip file.
    Package is named as follows:

    `occm_csar_<marketing-release-number>.zip`
    Example:

    `occm_csar_24_3_0_0_0.zip`

2.2.1.2 Pushing the Images to Customer Docker Registry

To push the images to the registry:

1. Untar the OCCM package to the specific repository: `tar -xvf occm_csar_<marketing-release-number>.zip`
   For example: `tar -xvf occm_csar_24_3_0_0_0.zip`

   The package consists of the following files and folders:

   a. **Files:** OCCM Docker Images file and OCCM Helm Chart
      - OCCM Helm chart: `occm-24.3.0.tgz`
      - OCCM Network Policy Helm Chart: `occm-network-policy-24.3.0.tgz`
      - Images: `occm-24.3.0.tar, nf_test-24.3.0.tar, ocdebug-tools-24.3.0.tar`

   b. **Scripts:** Custom values and Alert files:
      - OCCM Custom Values File: `occm_custom_values_24.3.0.yaml`
      - Sample Grafana Dashboard file: `occm_metric_dashboard_promha_24.3.0.json`
      - Sample Alert File: `occm_alerting_rules_promha_24.3.0.yaml`
      - MIB files: `occm_mib_24.3.0.mib, occm_mib_tc_24.3.0.mib, toplevel.mib`
      - Configuration Open API specification: `occm_configuration_openapi_24.3.0.json`
      - Network Policy Custom values file: `occm_network_policy_custom_values_24.3.0.yaml`

c. **Definitions**: Definitions folder contains the CNE Compatibility and definition files:
   - `occm_cne_compatibility.yaml`
   - `occm.yaml`

d. TOSCA-Metadata: `TOSCA.meta`

2. Verify the checksum of tar balls mentioned in the Readme.txt.
3. Run the following command to push the Docker images to docker registry:

   ```
   docker load --input <image_file_name.tar>
   ```

4. Run the following command to push the docker files to docker registry:

   ```
   docker tag <image-name>:<image-tag> <docker-repo>/<image-name>:<image-tag>
   docker push <docker_repo>/<image_name>:<image-tag>
   ```

5. Run the following command to check if all the images are loaded:

   ```
   docker images
   ```

6. Run the following command to push the Helm charts to Helm repository:

   ```
   helm cm-push --force <chart name>.tgz <helm repo>
   ```

   For example:

   ```
   helm cm-push --force occm-24.3.0.tgz occm-helm-repo
   ```

2.2.1.3 Verifying and Creating Namespace

This section explains how to verify or create a namespace in the system. If the namespace does not exist, the user must create it.

To verify and create OCCM namespace, perform the following steps:

1. Run the following command to verify if the required namespace already exists in the system:

   ```
   $ kubectl get namespace
   ```

2. In the output of the above command, check if the required namespace is available. If not available, create the namespace using the following command:

   ```
   $ kubectl create namespace <required namespace>
   ```

   For example, the following kubectl command creates the namespace `occm-ns`:

   ```
   $ kubectl create namespace occm-ns
   ```

Naming Convention for Namespaces

The namespace must meet the following requirements:

- start and end with an alphanumeric character
- contain 63 characters or less
- contain only alphanumeric characters or '-'

> **Note**
>
> It is recommended to avoid using the prefix `kube-` when creating a namespace. The prefix is reserved for Kubernetes system namespaces.

2.2.1.4 Creating Service Account, Role, and Rolebinding

2.2.1.4.1 Global Service Account Configuration

This section is optional and it describes how to manually create a service account, role, and rolebinding.

A custom service account can be provided for OCCM deployment in global.serviceAccountName of `occm_custom_values_<version>.yaml`.

A custom service account can be provided for helm in global.serviceAccountName:

```
global:
  dockerRegistry: cgbu-occncc-dev-docker.dockerhub-phx.oci.oraclecorp.com
  serviceAccountName: ""
```

Configuring Global Service Account to Manage NF Certificates with OCCM and NF in the Same Namespace

A sample OCCM service account yaml file to create custom service account is as follows:

```
## Service account yaml file for occm-sa
apiVersion: v1
kind: ServiceAccount
metadata:
  name: occm-sa
  namespace: occm
  annotations: {}
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: occm-role
  namespace: occm
rules:
- apiGroups:
  - "" # "" indicates the core API group
  resources:
  - services
  - configmaps
  - pods
  - secrets
  - endpoints
  - namespaces
```

```
  verbs:
  - get
  - watch
  - list
  - create
  - delete
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: occm-rolebinding
  namespace: occm
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: occm-role
subjects:
- kind: ServiceAccount
  name: occm-sa
  namespace: occm
```

Configuring Global Service Account to Manage NF Certificates with OCCM and NF in Separate Namespaces

OCCM provides support for key and certificate management in multiple namespaces.

In this deployment model, OCCM is deployed in namespace different from the components' namespaces managed by it. It needs privileges to read, write, and delete Kubernetes secrets in the managed namespaces.

This is achieved by creating multiple namespace specific roles and binding them to the service account for OCCM.

- **AUTOMATIC Service Account Configuration:** Roles and role bindings are created for each namespace specified using the `occmAccessedNamespaces` field in `occm_custom_values.yaml`. A service account for OCCM is created automatically and the roles created are assigned using the corresponding role binding.
  Namespaces managed by OCCM service account:

  ```
  occmAccessedNamespaces:
    - ns1
    - ns2
  ```

  > **Note**
  >
  > Automatic Service Account Configuration is applicable for Single Namespace Management as well

- **Custom Service Account Configuration:** A custom service account can also be configured against the `serviceAccountName` field in occm_custom_values.yaml. If this is provided, automatic service account creation doesn't get triggered. The `occmManagedNamespaces` field doesn't need to be configured.

A sample OCCM service account yaml file for creating a custom service account is as follows:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: occm-sa
  namespace: occm
  annotations: {}
---

apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: ns1
  name: occm-secret-writer-role
rules:
- apiGroups:
  - "" # "" indicates the core API group
  resources:
  - secrets
  - namespaces
  verbs:
  - get
  - watch
  - list
  - create
  - update
  - delete
---

apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: ns2
  name: occm-secret-writer-role
rules:
- apiGroups:
  - "" # "" indicates the core API group
  resources:
  - secrets
  - namespaces
  verbs:
  - get
  - watch
  - list
  - create
  - update
  - delete

---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: occm-secret-writer-rolebinding
  namespace: ns1
```

```
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: occm-secret-writer-role
subjects:
- kind: ServiceAccount
  name: occm-sa
  namespace: occm
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: occm-secret-writer-rolebinding
  namespace: ns2
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: occm-secret-writer-role
subjects:
- kind: ServiceAccount
  name: occm-sa
  namespace: occm

---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: occm-role
  namespace: occm
rules:
- apiGroups:
  - "" # "" indicates the core API group
  resources:
  - services
  - configmaps
  - pods
  - secrets
  - endpoints
  - namespaces
  verbs:
  - get
  - watch
  - list
  - create
  - delete
  - update
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: occm-rolebinding
  namespace: occm
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: occm-role
```

```
  subjects:
  - kind: ServiceAccount
    name: occm-sa
    namespace: occm
```

2.2.1.4.2 Helm Test Service Account Configuration

helmTestServiceAccountName is an optional field in the `occm_custom_values_<version>.yaml` file. Custom service account can be provided for helm in `global.helmTestServiceAccountName:`:

```
global:
  helmTestServiceAccountName: occm-helmtest-serviceaccount
```

A sample helm test service account yaml file is as follows:

```
helm test service account apiVersion: v1
kind: ServiceAccount
metadata:
  name: occm-helmtest-serviceaccount
  namespace: occm
  annotations: {}
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: occm-helmtest-role
  namespace: occm
rules:
- apiGroups:
  - "" # "" indicates the core API group
  resources:
  - services
  - configmaps
  - pods
  - secrets
  - endpoints
  - serviceaccounts
  - namespaces
  verbs:
  - get
  - watch
  - list
- apiGroups:
  - policy
  resources:
  - poddisruptionbudgets
  verbs:
  - get
  - watch
  - list
  - update
- apiGroups:
  - apps
  resources:
```

```
  - deployments
  - statefulsets
  verbs:
  - get
  - watch
  - list
  - update
- apiGroups:
  - autoscaling
  resources:
  - horizontalpodautoscalers
  verbs:
  - get
  - watch
  - list
  - update
- apiGroups:
  - rbac.authorization.k8s.io
  resources:
  - roles
  - rolebindings
  verbs:
  - get
  - watch
  - list
  - update
- apiGroups:
  - monitoring.coreos.com
  resources:
  - prometheusrules
  verbs:
  - get
  - watch
  - list
  - update

---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: occm-helmtest-rolebinding
  namespace: occm
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: occm-helmtest-role
subjects:
- kind: ServiceAccount
  name: occm-helmtest-serviceaccount
  namespace: occm
```

2.2.1.5 Configuring Network Policies

Kubernetes network policies allow you to define ingress or egress rules based on Kubernetes resources such as Pod, Namespace, IP, and Port. These rules are selected based on

Kubernetes labels in the application. These network policies enforce access restrictions for all the applicable data flows except communication from Kubernetes node to pod for invoking container probe.

Note: Configuring network policies is an optional step. Based on the security requirements, network policies may or may not be configured.

For more information on network policies, see https://kubernetes.io/docs/concepts/services-networking/network-policies/.

Configuring Network Policies

Following are the various operations that can be performed for network policies:

2.2.1.5.1 Installing Network Policies

Prerequisite

Network Policies are implemented by using the network plug-in. To use network policies, you must be using a networking solution that supports Network Policy.

Note:

For a fresh installation, it is recommended to install Network Policies before installing OCCM. However, if OCCM is already installed, you can still install the Network Policies.

To install network policy

1. Open the occm_network_policy_custom_values_24.3.0.yaml file provided in the release package zip file. For downloading the file, see Downloading the OCCM Package and Pushing the Images to Customer Docker Registry.
2. The file is provided with the default network policies. If required, update the occm_network_policy_custom_values_24.3.0.yaml file. For more information on the parameters, see the Configuration Parameters for network policy parameter table
   - To connect with CNC Console, update the below parameter in the `allow-ingress-from-console` network policy in the `occm_network_policies_custom_values_24.3.0.yaml`file:
     - kubernetes.io/metadata.name: <namespace in which CNCC is deployed>
   - In `allow-ingress-prometheus` policy, kubernetes.io/metadata.name parameter must contain the value for the namespace where Prometheus is deployed, and app.kubernetes.io/name parameter value should match the label from Prometheus pod.
3. Run the following command to install the network policies:

   ```
   helm install <release_name> -f
   <occm_network_policy_custom_values_<version>.yaml> --namespace
         <namespace> <chartpath>./<chart>.tgz
   ```

   For example:

   ```
   helm install occm-network-policy -f
   occm_network_policy_custom_values_24.3.0.yaml --namespace occm occm-
   network-policy-24.3.0.tgz
   ```

   Where,

- `helm-release-name`: occm-network-policy Helm release name.
- `custom-value-file`: occm-network-policy custom value file.
- `namespace`: OCCM namespace.
- `network-policy`: location where the network-policy package is stored.

Note:

- Connections that were created before installing network policy and still persist are not impacted by the new network policy. Only the new connections would be impacted.

2.2.1.5.2 Upgrading Network Policies

To add, delete, or update network policy

1. Modify the `occm_network_policy_custom_values_24.3.0.yaml` file to update, add, or delete the network policy.
2. Run the following command to upgrade the network policies:

   ```
   helm upgrade <release_name> -f
   occm_network_policy_custom_values_<version>.yaml --namespace
         <namespace> <chartpath>./<chart>.tgz
   ```

   For example:

   ```
   helm upgrade occm-network-policy -f
   occm_network_policy_custom_values_24.3.0.yaml --namespace occm occm-
   network-policy-24.3.0.tgz
   ```

   Where,

   - `helm-release-name`: occm-network-policy Helm release name.
   - `custom-value-file`: occm-network-policy custom value file.
   - `namespace`: OCCM namespace.
   - `network-policy`: location where the network-policy package is stored.

2.2.1.5.3 Verifying Network Policies

Run the following command to verify if the network policies are deployed successfully:

```
kubectl get networkpolicy -n <namespace>
```

For Example:

```
kubectl get networkpolicy -n occm
```

Where,

- `helm-release-name`: occm-network-policy Helm release name.
- `namespace`: OCCM namespace.

2.2.1.5.4 Uninstalling Network Policies

Run the following command to uninstall all the network policies:

```
helm uninstall <release_name> --namespace <namespace>
```

For Example:

```
helm uninstall occm-network-policy --namespace occm
```

2.2.1.5.5 Configuration Parameters for Network Policies

Table 2-4 Supported Kubernetes Resource for Configuring Network Policies

Parameter	Description	Details
apiVersion	This is a mandatory parameter. Specifies the Kubernetes version for access control. Note: This is the supported api version for network policy. This is a read-only parameter.	Data Type: string Default Value: `networking.k8s.i o/v1`
kind	This is a mandatory parameter. Represents the REST resource this object represents. Note: This is a read-only parameter.	Data Type: string Default Value: NetworkPolicy

Table 2-5 Supported parameters for Configuring Network Policies

Parameter	Description	Details
metadata.name	This is a mandatory parameter. Specifies a unique name for the network policy.	{{ .metadata.name }}
spec.{}	This is a mandatory parameter. This consists of all the information needed to define a particular network policy in the given namespace. **Note**: NRF supports the spec parameters defined in Kubernetes Resource Category.	Default Value: NA

For more information about this functionality, see the Network Policies section in *Oracle Communications Cloud Native Core, Certificate Management User Guide*

2.2.2 Installation Tasks

This section explains how to install Oracle Communications Cloud Native Core, Certificate Management.

> **Note**
>
> - Before installing OCCM, you must complete the Prerequisites and Preinstallation Tasks.
> - In a georedundant deployment, perform the steps explained in this section on all the georedundant sites.

2.2.2.1 Installing OCCM Package

This section includes information about OCCM deployment.

OCCM Deployment on Kubernetes

To install OCCM using Comand Line Interface (CLI):

1. Execute the following command to check the version of the helm chart installation:

   ```
   helm search repo <release_name> -l
   ```

   For example:

   ```
   helm search repo occm -l
   ```

   ```
   NAME                            CHART VERSION   APP VERSION     DESCRIPTION
   cgbu-occm-dev-helm/occm            24.3.0         24.3.0           A helm
   chart for OCCM deployment
   ```

2. Prepare occm_custom_values_24.3.0.yaml file with the required parameter information. For more information on these parameters, see Configuration Options.

   > **Note**
   >
   > **Metrics scrapping on CNE**
   > To enable metrics scrapping at CNE, add the following annotation under `global.nonlbDeployments` section:
   >
   > ```
   > nonlbDeployments:
   >       labels: {}
   >       annotations:
   >         oracle.com/cnc: "true"
   > ```

> **Note**
>
> **CNE LB VM Support**
> To support the CNE LB VM selection based on destination IP address with OCCM, add the following annotation under `global.nonlbDeployments` section:
>
> ```
> nonlbDeployments:
>       labels: {}
>       annotations:
>         oracle.com.cnc/egress-network: oam
> ```

> **Note**
>
> **Traffic Segregation**
> To enable egress network attachment for OCCM to CA communication, add the following annotation with the appropriate value:
>
> ```
> annotations:
>   k8s.v1.cni.cncf.io/networks: ""
> ```
>
> Where **k8s.v1.cni.cncf.io/networks**: contains the network attachment information that the pod uses for network segregation. For more information about *Traffic Segregation*, see *Oracle Communications Cloud Native Core, Certificate Management User Guide*.
>
> ```
> Example:
>   nonlbDeployments:
>       labels: {}
>       annotations:
>         k8s.v1.cni.cncf.io/networks: default/nf-oam-egr1@nf-oam-egr1
> ```

3. **Deploy OCCM:**
   a. **Verify Deployment**:
   To verify the deployment status, open a new terminal and run the following command:

   ```
   kubectl get pods -n <namespace_name> -w
   ```

   For example:

   ```
   kubectl get pods -n occm-ns -w
   ```

   The pod status gets updated on a regular interval. When helm install command is run and exits with the status, you may stop watching the status of kubernetes pods.

   b. **Installation using helm tar**:

To install using Helm tar, run the following command:

```
helm install <release_name> -f occm_custom_values_<version>.yaml -n
<namespace_name>
    occm-<version>.tgz
```

For example:

```
helm install occm -f occm_custom_values_24.3.0.yaml --namespace occm-ns
occm-24.3.0.tgz
```

Sample Output

```
# Example:
NAME: occm
LAST DEPLOYED: Wed Nov 15 10:11:17 2023
NAMESPACE: occm-ns
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

c. **Installation using helm repository**:
To install using Helm repository, run the following command:

```
helm install <release_name> <helm-repo> -f
occm_custom_values_<version>.yaml --namespace <namespace_name> --
version <helm_version>
```

For example:

```
helm install cgbu-occm-dev-helm/occm -f occm_custom_values_24.3.0.yaml
--namespace occm-ns --version 24.3.0
```

where,

`release_name` and `namespace_name`: depends on customer configuration

`helm-repo`: is the repository name where the helm images, charts are stored

`values`: is the helm configuration file which needs to be updated based on the docker registry

4. Run following command to check the deployment status:

```
helm status <release_name>
```

5. Run the following command to check if all the services are deployed and running:

```
kubectl get svc -n occm-ns
```

For example:

```
NAME           TYPE         CLUSTER-IP      EXTERNAL-IP    PORT(S)
AGE
service/occm   ClusterIP    10.233.26.68    <none>         8989/TCP
21m
```

6. Run the following command to check if the pods are up and running:

   ```
   kubectl get po -n occm-ns
   ```

   For example:

   ```
   NAME                                READY   STATUS    RESTARTS      AGE
   pod/occm-occm-5fb5557f75-mjhcbh     1/1     Running   0             21m
   ```

2.2.3 Postinstallation Tasks

This section explains the postinstallation tasks for OCCM.

2.2.3.1 Verifying Installation

To verify if OCCM is installed:

1. Run the following command to verify the installation status:

   ```
   <helm-release> -n <namespace>
   ```

   For example:

   ```
   helm status occm -n occm
   ```

   In the output, if `STATUS` is showing as `deployed`, then the installation is successful.

2. Run the following command to verify if the pods are up and active:

   ```
   kubectl get jobs,pods -n release_namespace
   ```

   For example:

   ```
   kubectl get pods -n occm
   ```

3. Run the following command to verify if the services are deployed and active:

   ```
   kubectl get services -n release_namespace
   ```

   For example:

   ```
   kubectl get services -n occm
   ```

> **Note**
>
> Take a backup of the following files that are required during fault recovery:
>
> - Updated `occm_custom_values_24.3.0.yaml` file.
>
> If the installation is not successful or you do not see the status as Running for all the pods, perform the troubleshooting steps provided in the *Oracle Communications cloud Native Core, Certificate Management Troubleshooting Guide.*

2.2.3.2 Performing Helm Test

Helm Test is a feature which will validate OCCM Successful Installation along with readiness (Readiness probe url configured will be checked for success) of the pod. The pod to be checked will be based on the namespace and label selector configured for the helm test configurations.

> **Note**
>
> Helm3 is mandatory for the Helm Test feature to work.

You must follow the following instructions to run the Helm test functionality. The configurations mentioned in the following step must be done before executing the helm install command.

1. Configure the helm test configurations that are under global sections in values.yaml file.

```
global:
  # Helm test related configurations
  test:
    nfName: occm
    image:
      name: occm/nf_test
      tag: <version>
      imagePullPolicy: IfNotPresent
    config:
      logLevel: WARN
      timeout: 240
```

2. Run the following helm test command:

```
helm test <helm_release_name> -n <k8s namespace>
```

For example:

```
helm test occm -n occmdemo
Pod occm-test pending
Pod occm-test pending
Pod occm-test pending
Pod occm-test pending
Pod occm-test running
Pod occm-test succeeded
```

```
NAME: occm
LAST DEPLOYED: Wed Nov 08 02:38:25 2023
NAMESPACE: occmdemo
STATUS: deployed
REVISION: 1
TEST SUITE:     occm-test
Last Started:   Wed Nov 08 02:38:25 2023
Last Completed: Wed Nov 08 02:38:25 2023
Phase:          Succeeded
NOTES:
# Copyright 2020 (C), Oracle and/or its affiliates. All rights reserved.

Thank you for installing occm.

Your release is named occm , Release Revision: 1.
To learn more about the release, try:

  $ helm status occm
  $ helm get occm
```

3. Wait for the helm test job to complete. Check if the test job is successful in the output.

2.2.3.3 CNC Console Configuration

CNC Console instance configuration must be updated to enable access of OCCM.

For information on how to enable OCCM, see *Oracle Communications Cloud Native Configuration Console Installation, Upgrade, and Fault Recovery Guide*.

2.2.3.4 OCCM Configuration

For information on the features supported by OCCM, issuer and certificate configurations, see OCCM Supported Features in the *Oracle Communications Cloud Native Core, Certificate Management User Guide.*

For OCCM REST API details, see *Oracle Communications Cloud Native Core, Certificate Management REST Specifications Guide.*

3
Customizing OCCM

This chapter provides information about customizing OCCM deployment in a cloud native environment.

The OCCM deployment is customized by overriding the default values of various configurable parameters in the `occm_custom_values_<version>.yaml` file.

Perform the following steps to customize the custom yaml files :

1. Use the custom values and templates delivered as part of the package. For more information on how to download the package from MOS, see Downloading the OCCM Package section.
2. Customize the appropriate custom value file.
3. Save the updated files.

> **Note**
>
> - All parameters mentioned as mandatory must be present in custom-values.yaml file.
> - All fixed value parameters listed must be present in the custom values yaml file with the exact values as specified in this section.
> - For installing OCCM in an existing NF deployment, see the 'Introducing OCCM in an Existing NF Deployment' section in the *Oracle Communications Cloud Native Core, Certificate Management User Guide.*

3.1 Configuration Options

Table 3-1 Configuration Options

Parameter	Description	Details
global.dockerRegistry	This is a mandatory parameter. Here, user provides the registry that contains OCCM images. It comprises of <registry-url>	**Data Type:** String **Range:** It may contain lowercase letters, digits, and separators. A separator is defined as a period, one or two underscores, or one or more dashes. **Default Value:** cgbu-occm-dev-docker.dockerhub-iad.oci.oraclecorp.com

Table 3-1 (Cont.) Configuration Options

Parameter	Description	Details
global.serviceAccountName	This is an optional parameter. Name of service account. If this field is kept empty, then a default service account with release name will be auto created. If any value is provided, then a custom service account has to be created manually before deployment.	**Data Type:** String **Range:** Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods, and dashes. An image name may not start with a period or a dash and may contain a maximum of 128 characters.
global.occmAccessedNamespaces	This is an optional field. In case of OCCM multiple namespace support, namespaces are listed here for automatic service account creation.	**Data Type:** List (String) **Default Value:** NA **Range:** Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods, and dashes. An image name may not start with a period or a dash and may contain a maximum of 128 characters.
global.customExtension	This is an optional field. Custom extension to include custom labels and annotation.	**Data Type:** String **Default Value:** NA **Range:** Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods, and dashes. An image name may not start with a period or a dash and may contain a maximum of 128 characters.
global.customExtension.allResources.labels	This is an optional parameter. This can be used to add custom label(s) to all Kubernetes resources that will be created by OCCM helm chart.	**Data Type:** String **Range:** Custom labels that need to be added to all the OCCM Kubernetes resources.
global.customExtension.allResources.annotations	This is an optional parameter. This can be used to add custom annotation(s) to all Kubernetes resources that will be created by OCCM helm chart.	**Data Type:** String **Range:** Custom annotations that need to be added to all the OCCM Kubernetes resources.
global.customExtension.nonlbServices.labels	This is an optional parameter. This can be used to add custom label(s) to all non-Load Balancer Type Services that will be created by OCCM helm chart.	**Data Type:** String **Range:** Custom labels that need to be added to OCCM that are considered as not Load Balancer type.
global.customExtension.nonlbServices.annotations	This is an optional parameter. This can be used to add custom annotation(s) to all non-Load Balancer Type Services that will be created by OCCM helm chart.	**Data Type:** String **Range:** Custom annotations that need to be added to OCCM that are considered as not Load Balancer type.

Table 3-1 (Cont.) Configuration Options

Parameter	Description	Details
global.customExtension.nonlbDeployments.labels	This is an optional parameter. This can be used to add custom label(s) to all Deployments that will be created by OCCM helm chart which are associated to a Service which if not of Load Balancer Type.	**Data Type:** String **Range:** Custom labels that need to be added to OCCM Deployments that are associated to a service which is not of Load Balancer type.
global.customExtension.nonlbDeployments.annotations	This is an optional parameter. This can be used to add custom annotation(s) to all Deployments that will be created by OCCM helm chart which are associated to a Service which if not of Load Balancer Type. For example: oracle.com/cnc: "true"` oracle.com.cnc/egress-network: oam	**Data Type:** String **Range:** Custom annotations that need to be added to OCCM Deployments that are associated to a service which is not of Load Balancer type.
global.ephemeralStorage.limits.containersLogStorage	This is a mandatory parameter. Set value for Ephemeral Storage Limits.	**Data Type:** Integer **Range:** It can take values in integer that is further used in MBs. **Default Value:** 1000
global.ephemeralStorage.limits.containersCriticalStorage	This is a mandatory parameter. Set value for Ephemeral Storage Limits.	**Data Type:** Integer **Range:** It can take values in integer that is further used in MBs. **Default Value:** 2
global.ephemeralStorage.requests.containersLogStorage	This is a mandatory parameter. Set value for Ephemeral Storage Requests.	**Data Type:** Integer **Range:** It can take values in integer that is further used in MBs. **Default Value:** 50
global.ephemeralStorage.requests.containersCriticalStorage	This is a mandatory parameter. Set value for Ephemeral Storage Requests.	**Data Type:** Integer **Range:** It can take values in integer that is further used in MBs. **Default Value:** 2
global.hookJobResources.limit.cpu	This is an optional parameter. It limits the number of CPUs to be used by the helm test pod.	**Data Type:** Integer **Range:** Valid Integer value allowed. **Default Value:** 0.5
global.hookJobResources.limit.memory	This is an optional parameter. It limits the memory to be used by the helm test pod.	**Data Type:** Integer **Range:** Valid Integer value followed by Mi/Gi etc. **Default Value:** 0.5Gi

Table 3-1 (Cont.) Configuration Options

Parameter	Description	Details
global.hookJobResources.limit.logStorage	This is an optional parameter. It limits the logStorage (ephemeral storage) to be used by the helm test pod.	**Data Type:** Integer **Range:** Values will be set by global.ephemeralStorage.requests.containerLogStorage. **Default Value:** 50Mi
global.hookJobResources.limit.criticalStorage	This is an optional parameter. It limits the criticalStorage (ephemeral storage) to be used by the helm test pod.	**Data Type:** Integer **Range:** Values will be set by global.ephemeralStorage.limits.containers CrititcalStorage. **Default Value:** 2
global.hookJobResources.request.cpu	This is an optional parameter. It requests the number of CPUs to be used by the helm test pod.	**Data Type:** Integer **Range:** Valid Integer value allowed. **Default Value:** 0.5
global.hookJobResources.request.memory	This is an optional parameter. It requests the memory to be used by the helm test pod.	**Data Type:** Integer **Range:** Valid Integer value followed by Mi/Gi etc. **Default Value:** 0.5Gi
global.hookJobResources.request.logStorage	This is an optional parameter. It requests the logStorage (ephemeral storage) to be used by the helm test pod.	**Data Type:** Integer **Range:** Values will be set by global.ephemeralStorage.requests.containerLogStorage. **Default Value:** 50Mi
global.hookJobResources.request.criticlStorage	This is an optional parameter. It requests the criticlStorage (ephemeral storage) to be used by the helm test pod.	**Data Type:** Integer **Range:** Values will be set by global.ephemeralStorage.limits.containers CrititcalStorage. **Default Value:** 2
global.k8sResource.container.prefix	This is an optional parameter. This value will be used to prefix to all the container names of OCCM.	**Data Type:** String **Range:** Value that will be prefixed to all the container names of Ingress Gateway.
global.k8sResource.container.suffix	This is an optional parameter. This value will be used to suffix to all the container names of OCCM.	**Data Type:** String **Range:** Value that will be suffixed to all the container names of Ingress Gateway.
global.helmTestServiceAccountName	This is an optional parameter. For helm test execution, preference goes to global.helmTestServiceAccountName first. If this is not available then global.serviceAccountName will be referred. If both of these are missing, then default service account will be created and used.	**Data Type:** String **Range:** Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods, and dashes. An image name may not start with a period or a dash and may contain a maximum of 128 characters.

Table 3-1 (Cont.) Configuration Options

Parameter	Description	Details
global.test.nfName	This is a mandatory parameter. Name of deployment for which helm test is done.	**Data Type:** String **Range:** NF Name **Default Value:** OCCM
global.test.image.name	This is a mandatory parameter. Image name for the helm test container image.	**Data Type:** String **Range:** Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods, and dashes. An image name may not start with a period or a dash and may contain a maximum of 128 characters. **Default Value:** OCCM
global.test.image.tag	This is a mandatory parameter. Image version tag for helm test.	**Data Type:** String **Range:** Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods. and dashes. An image name may not start with a period or a dash and may contain a maximum of 128 characters.
global.test.image.imagePullPolicy	This is an optional parameter. Pull Policy decides from where to pull the image.	**Data Type:** String **Range:** It can take a value from the following: IfNotPresent, Always, Never IfNotPresent is the default pullPolicy
global.test.config.logLevel	This is a mandatory parameter. Pull Policy decides from where to pull the image.	**Data Type:** String **Range:** WARN, DEBUG, INFO, etc. **Default Value:** Info
global.test.config.timeout	This is a mandatory parameter. Timeout value for the helm test operation. If exceeded, helm test will be considered as failure.	**Data Type:** String **Range:** 1-300 seconds **Default Value:** 240
global.test.resources	This is a mandatory parameter. The mentioned Kubernetes resources are logged in Helm test.	**Data Type:**(List) String **Range:** It takes resources and its version in the form of <resource_name>/<max_version_supportedbyNF> - horizontalpodautoscalers/v1 - deployments/v1 - serviceaccounts/v1 - roles/v1 - services/v1 - rolebindings/v1
global.test.complianceEnable	This is a mandatory parameter. It will enable or disable helm test resource logging.	**Data Type:** Boolean **Range:** True or False **Default Value:** True

Table 3-1 (Cont.) Configuration Options

Parameter	Description	Details
global.extraContainers	This is a mandatory parameter. To enable or disable the debug tools container.	**Data Type:** enum **Range:**DISABLED, ENABLED **Default Value:** DISABLED
global.debugToolContainerMemoryLimit	This is a mandatory parameter. Debug tool container memory limit.	**Data Type:** String **Range:** Valid Integer value followed by Mi/Gi etc. **Default Value:** debug-tools-dir
global.extraContainersVolumesTpl	This is a mandatory parameter. Debug tool extra container volume details.	**Data Type:** String **Range:** It may contain lowercase letters, digits, and separators. A separator is defined as a period, one or two underscores, or one or more dashes. **Default Value:** 4Gi
global.extraContainersTpl	This is a mandatory parameter. Debug tool extra container command details.	**Data Type:** String **Range:** It may contain lowercase letters, digits, and separators. A separator is defined as a period, one or two underscores, or one or more dashes.
image.tag	This is a mandatory parameter. Image Tag to be used for OCCM.	**Data Type:** enum **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods, and dashes. A tag name may not start with a period or a dash and may contain a maximum of 128 characters. **Default Value:** DISABLED
image.name	This is a mandatory parameter. It is the image name of the OCCM.	**Data Type:** String **Range:**Valid ASCII and may contain lowercase and uppercase letters, digits, underscores, periods, and dashes. A tag name may not start with a period or a dash and may contain a maximum of 128 characters.
image.pullPolicy	This is an optional parameter. Pull Policy decides from where to pull the image.	**Data Type:** String **Range:** It can take a value from the following: IfNotPresent, Always, Never IfNotPresent is the default pullPolicy
ports.containerPort	This is a mandatory parameter. It is the http port of the container for the OCCM.	**Data Type:** Integer **Range:** 0-65535 **Default value:** 8989
ports.actuatorPort	This is a mandatory parameter. It is the actuator port of the container for the OCCM.	**Data Type:** Integer **Range:**0-65535 **Default value:** 9000

Table 3-1 (Cont.) Configuration Options

Parameter	Description	Details
ports.servicePort	This is a mandatory parameter. It is the service port of the container for the OCCM.	**Data Type:** Integer **Range:**0-65535 **Default value:** 8989
deployment.livenessProbe.initialDelaySeconds	This is an optional parameter. It specifies that the kubelet should perform a liveness probe every xx seconds.	**Data Type:** Integer **Range:**0-65535 **Default value:** 60
deployment.livenessProbe.periodSeconds	This is an optional parameter. It specifies that the kubelet should perform a liveness probe every xx seconds.	**Data Type:** Integer **Range:** 0-65535 **Default value:** 3
deployment.livenessProbe.timeoutSeconds	This is an optional parameter. It is the number of seconds after which the probe times out.	**Data Type:** Integer **Range:**0-65535 **Default value:** 15
deployment.livenessProbe.successThreshold	This is an optional parameter. Minimum consecutive successes for the probe to be considered successful after having failed.	**Data Type:** Integer **Range:**0-65535 **Default value:** 1
deployment.livenessProbe.failureThreshold	This is an optional parameter. When a Pod starts and the probe fails, Kubernetes will try failureThreshold times before giving up.	**Data Type:** Integer **Range:**0-65535 **Default value:** 3
deployment.readinessProbe.initialDelaySeconds	This is an optional parameter. It tells the kubelet that it should wait second before performing the first probe.	**Data Type:** Integer **Range:**0-65535 **Default value:** 20
deployment.readinessProbe.timeoutSeconds	This is an optional parameter. It is the number of seconds after which the probe times out.	**Data Type:** Integer **Range:**0-65535 **Default value:** 3
deployment.readinessProbe.periodSeconds	This is an optional parameter. It specifies that the kubelet should perform a liveness probe every xx seconds.	**Data Type:** Integer **Range:**0-65535 **Default value:** 10
deployment.readinessProbe.successThreshold	This is an optional parameter. Minimum consecutive successes for the probe to be considered successful after having failed.	**Data Type:** Integer **Range:**0-65535 **Default value:** 1
deployment.readinessProbe.failureThreshold	This is an optional parameter. When a Pod starts and the probe fails, Kubernetes will try failureThreshold times before giving up.	**Data Type:** Integer **Range:**0-65535 **Default value:** 3

Table 3-1 (Cont.) Configuration Options

Parameter	Description	Details
resources.limits.cpu	This is an optional parameter. It limits the number of CPUs to be used by the OCCM.	**Data Type:** Float **Range:**Valid floating point value between 0 and 1 **Default Value:** 2
resources.limits.memory	This is an optional parameter. It limits the memory utilization by the microservice.	**Data Type:** String **Range:**Valid Integer value followed by Mi/Gi etc. **Default value:** 2Gi
resources.limits.logStorage	This is a mandatory parameter. It limits the logStorage (ephemeral storage) to be used by the helm test pod.	**Data Type:** Integer **Range:**Values will be set by global.ephemeralStorage.limits.containers LogStorage. **Default value:** 1000
resources.limits.criticalStorage	This is a mandatory parameter. It limits the criticalStorage (ephemeral storage) to be used by the helm test pod.	**Data Type:** Integer **Range:**Values will be set by global.ephemeralStorage.limits.containers CrititcalStorage. **Default value:** 2
resources.requests.cpu	This is a mandatory parameter. The minimum amount of CPUs required.	**Data Type:** String **Range:**Valid floating point value between 0 and 1. **Default value:** 1
resources.requests.memory	This is a mandatory parameter. The minimum amount of memory required.	**Data Type:** String **Range:** Valid Integer value followed by Mi/Gi etc. **Default value:** 1Gi
resources.requests.logStorage	This is a mandatory parameter. The minimum amount of logStorage (ephemeral storage).	**Data Type:** Integer **Range:** Values will be set by global.ephemeralStorage.requests.containerLogStorage. **Default value:** 50
resources.requests.criticalStorage	This is a mandatory parameter. The minimum amount of criticalStorage (ephemeral storage)	**Data Type:** Integer **Range:**Values will be set by global.ephemeralStorage.requests.containerCrititcalStorage. **Default value:** 2
log.level.occm	This is a mandatory parameter. It is the level at which the user wants to see application level logs.	**Data Type:** String **Range:** WARN, DEBUG, INFO, TRACE etc. **Default value:** INFO
log.level.root	This is a mandatory parameter. It is the level at which user wants to see root level logs.	**Data Type:** String **Default value:** ERROR
log.level.helidonFramework	This is a mandatory parameter. It is the level at which user wants to see helidon framework level logs.	**Data Type:** String **Default value:** ERROR

Table 3-1 (Cont.) Configuration Options

Parameter	Description	Details
occmConfig.cmp.config.useOccmCertSignForKur	This field, when set true, specifies that OCCM key and certificate will be used to sign the CMP request message. When set to false, old certificate is used as the signer certtificate.	**Data Type:** boolean **Default Value:** false **Range:** True or false
occmConfig.cmp.config.extractCertChainFromCmpResponse	This field, when set to true, specifies that certificate chain will be extracted from CA's CMP response message. In case, the CA doesn't send the chain, operator has the flexibility to manually configure it after setting the field to false.	**Data Type:** boolean **Default Value:** true **Range:** True or false
occmConfig.cmp.config.tls.enableX509StrictCheck	This is an optional parameter. This field, when set to true, "-x509_strict" will be included in openssl cmp cmd for strict checking of the X.509 certificates.	**Data Type:** boolean **Default Value:** true **Range:** True or false
occmConfig.cmp.config.tls.ignoreCriticalExtensionsCheck	This is an optional parameter. This field, when set to true, "-ignore_critical" will be included in openssl cmp cmd for checking of X.509 certificate critical extensions.	**Data Type:** boolean **Default Value:** false **Range:** True or false
occmConfig.cmp.config.tls.minProtocol	This is an optional field. This fields sets the minimum supported TLS version.	**Data Type:** String **Default Value:** TLSv1.2
occmConfig.cmp.config.tls.tlsNamedGroups	This is an optional field. This is equivalent to Groups in openssl. This sets the supported groups. For clients, the groups are set using the supported groups extension. The value must be colon separated groups.	**Data Type:** String **Default Value:** P-256:P-384:P-521:X25519:X448
occmConfig.cmp.config.tls.cipherStrings	This is an optional field. This field sets the available ciphers for TLSv1.2 and below. The value should be colon separated ciphers.	**Data Type:** String **Default Value:** ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-GCM-SHA256:\ ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:\ ECDHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES256-CCM:ECDHE-ECDSA-AES128-CCM

Table 3-1 (Cont.) Configuration Options

Parameter	Description	Details
occmConfig.cmp.config.tls.cipherSuites	This is an optional field. This field sets the available cipher suites for TLSv1.3. The value should be colon separated ciphers.	**Data Type:** String **Default Value:** TLS_AES_128_GCM_SHA256:TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_CCM_SHA256
occmConfig.cmp.config.tls.clientSignatureSchemes	This is an optional field. This is equivalent to SignatureAlgorithms in openssl. This sets the supported signature algorithms for TLSv1.2 and TLSv1.3. The value should be colon separated signature schemes.	**Data Type:** String **Default Value:** ecdsa_secp384r1_sha384:ecdsa_secp256r1_sha256:ed448:ed25519:rsa_pss_rsae_sha512:\ rsa_pss_rsae_sha384:rsa_pss_rsae_sha256:rsa_pss_pss_sha512:rsa_pss_pss_sha384:\ rsa_pss_pss_sha256:rsa_pkcs1_sha512:rsa_pkcs1_sha384:rsa_pkcs1_sha256
occmConfig.cmp.config.extractCertChainFromCmpResponse	This is an optional parameter. This field, when set true, specifies that the certficate chain will be extracted from CA's CMP response message.	**Data Type:** boolean **Default Value:** true
occmConfig.k8sSecretMonitoring	This is an optional parameter. This field, when set true, specifies that secret monitoring feature is enabled.	**Data Type:** boolean **Default Value:** true

4 Upgrading OCCM

> **Note**
>
> It is advisable to back up the OCCM configmap before performing an upgrade or rollback. During an upgrade, the existing configurations or certificates are preserved. However, in the event of a rollback, any configurations or certificates created during the upgrade will be lost. Therefore, having a backup allows you to reapply these configurations or certificates in future upgrades.
>
> To back up the OCCM configuration, see OCCM Configuration Backup . For restoring the configuration from a backup, see OCCM Configuration Restore.

This chapter provides information about upgrading the OCCM deployment to the latest release. It is recommended to perform OCCM upgrade in a specific order. For more information about the upgrade order, see Oracle Communications Cloud Native Core, Solution Upgrade Guide

Following are the steps to perform upgrade:

1. Take OCCM configuration backup by taking backup OCCM configmap.
2. Upgrade OCCM.

> **Note**
>
> Before proceeding with OCCM helm upgrade, you must backup the latest OCCM configmap and store the backup file in a location from where it can be easily restored.

Overview

You can use the Helm upgrade feature to upgrade OCCM from its current version to the latest version.

You can upgrade or Rollback OCCM from a source release to a target release using CLI procedures as outlined in the following table:

Upgrade/Rollback Task	References	Supported for CLI
OCCM Configuration Backup	NA	Yes
OCCM Upgrade	See *Oracle Communications Cloud Native Core, CD Control Server User Guide*	Yes

4.1 Supported Upgrade Paths

The following table lists the supported upgrade paths for OCCM:

Table 4-1 Supported Upgrade Paths

Source Release	Target Release	Upgrade Sequence	Comments
24.2.x or 24.1.x	24.3.x	• CNC Console • OCCM • NF • cnDBTier	• CNC Console should be upgraded before OCCM. • Helm upgrade OCCM using existing release name.

4.2 Preupgrade Tasks

This section provides information about preupgrade tasks to be performed before upgrading OCCM.

⚠ Caution

- No configuration should be performed during upgrade.
- Do not exit from helm upgrade command manually. After running the helm upgrade command, it takes some amount of time (depending upon number of PODs to upgrade) to upgrade all of the services. In the meantime, you must not press "ctrl+c" to come out from helm upgrade command. It may lead to anomalous behavior.

ⓘ Note

1. Keep current occm_custom_values_<version>.yaml file as backup, that is occm_custom_values_<version to be upgraded>.yaml.
2. Update the new custom_values.yaml defined for target OCCM release. See OCCM Parameters section in Configuration Options for more details about helm configurable parameters.
3. OCCM certificate configuration from version 24.1.0 onwards supports ecCurve SECP256k1 as it is a few bits weaker than SECP256r1. If there are any certificate configurations with ecCurve SECP256k1, the certificate has to be deleted before performing upgrade.
4. OCCM 24.1.0 onwards issuer configuration supports only unique server URLs. If there are any issuer configurations with same server URLs, then those issuer and corresponding certificates should be deleted before upgrade.

Preupgrade steps:

1. Keep current `occm_custom_values_<version>.yaml` file as backup, that is `occm_custom_values_24.2.0.yaml`.
2. Update the new `custom_values.yaml` defined for target OCCM release. For more details about Helm configurable parameters, see Configuration Options.

4.2.1 OCCM Configuration Backup

OCCM configurations are stored in configmap. You must take a backup before performing upgrade. These backups can be used to restore the OCCM configuration data.

To perform the backup, run the following command to create a temporary directory and save the json file containing configmap data. This can be used if there is any issue with the configmap data.

1. Log in to the deployment cluster and run the following command to take the data backup of the configmap:

   ```
   kubectl get cm <configmap name> -n <namespace> -o json >
   <backup_filen_name>_backup.json
   ```

2. Run the following command to verify the OCCM configmap name:

   ```
   $ kubectl get cm <configmap name> -n <namespace>
   ```

   For example:

   ```
   $ kubectl get cm occm-occm -n occm
   ```

3. Run the following command to backup OCCM configmap:

   ```
   $ kubectl get cm <configmap name> -o json -n <namespace>  > occm-config-
   map_<version>_backup.json
   ```

   For example:

   ```
   $ kubectl get cm occm-occm -o json -n occm > occm-config-
   map_24.2.0_backup.json
   ```

> **Note**
>
> You are recommended to take a periodic backup of configmap.

4.3 Upgrade Tasks

This section describes the procedure to upgrade OCCM to the latest release.

1. Prepare `occm_custom_values_<version>.yaml` file for upgrade.
2. Run the following command to upgrade OCCM using existing release release name:

   ```
   $ helm upgrade <occm_release_name> <helm_chart> -f
   <occm_custom_values_<version>.yaml> -n <namespace>
   ```

For example:

```
$ helm upgrade occm ocspf-helm-repo/occm -f
occm_custom_values_<version>.yaml -n occm
```

3. Run the following command to check the status of upgrade:

```
$ helm status <occm_release_name> -n <namespace>
```

For example:

```
$ helm status occm -n occm
```

> **Note**
>
> You must use the existing release name for upgrade or rollback.

4.4 Post Upgrade Tasks

This section provides the steps to be performed after upgrading OCCM.

For information on verifying OCCM Installation, see Verifying Installation

4.4.1 Parameters and Definitions During OCCM Upgrade

Table 4-2 OCCM Upgrade Parameters and Definitions

Parameters	Definitions
<release_name>	OCCM Helm release deployed. It could be found in the output of 'helm list' command
<namespace>	OCCM namespace in which release deployed
<helm_chart>	OCCM helm chart
<chart_version>	OCCM helm chart version in case helm charts are referred from helm repo
<helm_repo>	OCCM helm repo
<occm_custom_values_<version>.yaml>	OCCM customized values.yaml for target release

5
Rolling Back OCCM

> **Note**
>
> It is advisable to back up the OCCM configmap before performing an upgrade or rollback. However, in the event of a rollback, any configurations or certificates created during the upgrade will be lost. Therefore, having a backup allows you to reapply these configurations or certificates in future upgrades. During an upgrade, the existing configurations or certificates are preserved.
> To back up the OCCM configuration, see OCCM Configuration Backup . For restoring the configuration from a backup, see OCCM Configuration Restore.

This chapter provides information about rolling back the OCCM deployment to the previous release.

You can rollback OCCM from a source release to a target release using CLI procedures as outlined in the following table:

Upgrade /Rollback Task	References	Supported for CLI
OCCM Rollback		Yes

Perform the following steps to rollback OCCM:

1. Perform OCCM Configuration Restore.
   To restore the OCCM configuration, log in to to the deployment cluster and run the following command to restore the configmap:

   ```
   kubectl apply -f occm-config-map_24.2.0_backup.json
   ```

2. OCCM Rollback

5.1 Supported Rollback Paths

This section describes the supported rollback paths for OCCM:

Table 5-1 OCCM Rollback Sequence

Source Version	Target Version	Rollback Sequence	Comments
24.3.x	24.2.x or 24.1.x	• cnDBTier • NF • OCCM • CNC Console	• OCCM must be rolled back first followed by CNC Console rollback. • Helm rollback OCCM using existing release name. This will roll back OCCM, if enabled.

5.2 Prerollback Tasks

> **Note**
>
> - No configuration should be performed during rollback.
> - Do not exit from helm rollback command manually. After running the Helm rollback command, it takes some time (depending upon number of PODs to rollback) to roll back all of the services. In the meantime, you must not press "ctrl+c" to come out from helm rollback command. It may lead to anomalous behavior.

OCCM Configuration Restore

To restore the OCCM configuration:

1. Run the following command to get the configmap:

   ```
   $ kubectl get cm -n <namespace>
   ```

   Example:

   ```
   $ kubectl get cm occm-occm -n occm
   ```

2. Run the following command to delete the configmap:

   ```
   kubectl delete cm <configmap name> -n <namespace>
   ```

   Example:

   ```
   $ kubectl delete cm occm-occm -n occm
   ```

3. Run the following command to apply the backup configmap:

   ```
   kubectl apply -f occm-config-map_24.2.0_backup.json -n occm
   ```

5.3 Rollback Tasks

This section describes the procedure to roll back OCCM:

> **Note**
>
> You must use the existing release name for rollback.

1. Run the following command to check which revision you need to roll back:

   ```
   $ helm history <release_name> --namespace <release_namespace>
   ```

   For example:

   ```
   $ helm history occm --namespace namspace1
   ```

2. Run the following command to roll back to the required revision:

   ```
   $ helm rollback <release_name> <revision_number> --namespace
       <release_namespace>
   ```

   For example:

   ```
   $ helm rollback occm 1 --namespace namespace1
   ```

6
Uninstalling OCCM

This chapter provides information on how to uninstall OCCM. When you uninstall a Helm chart from the OCCM deployment, it removes only the Kubernetes objects created during the installation.

6.1 Uninstalling OCCM Using Helm

This chapter describes how to uninstall OCCM using Helm.

To uninstall OCCM using Helm 3, run the following command:

```
helm uninstall <release_name> --namespace <namespace_name>
```

For example:

```
helm uninstall occm--namespace occm-ns
```

> **Note**
>
> OCCM helm uninstallation does not remove the configmap entry. It is retained to store the state of OCCM configurations. For complete clean up, configmap must be manually deleted. For more information, see OCCM Cleanup

6.2 OCCM Cleanup

To clean up jobs when uninstalling OCCM:

1. Run the following command to check if any jobs are present after uninstalling OCCM:

   ```
   $ kubectl get jobs -n <namespace_name>
   ```

   For Example:

   ```
   $ kubectl get jobs -n occm-ns
   ```

2. Run the following command to delete any jobs that are present:

   ```
   $ kubectl delete jobs --all -n <namespace_name>
   ```

   For example:

   ```
   $ kubectl delete jobs --all -n occm-ns
   ```

3. Run the following command to delete the configmap created to persist the application data:

   ```
   $ kubectl delete configmap <occm configmap name> -n <namespace_name>
   ```

   For example:

   ```
   $ kubectl delete configmap occm-occm -n occm-ns
   ```

   Warning

   You can't restore the OCCM configuration data if the OCCM configMap is deleted.

7 Fault Recovery

This chapter provides information about fault recovery for Oracle Communications Cloud Native Core, Certificate Management deployment.

7.1 Impacted Areas

The following table shares information about the impacted areas during OCCM fault recovery:

Table 7-1 Fault Recovery Impacted Areas

Scenario	Details	Requires Fault Recovery or reinatallation of CNE	Requires Fault Recovery or reinatallation of OCCM	Comments
1	Complete site failure (due to infrastructure failure e.g. hardware/CNE etc)	Yes	Yes	NA
2	Corruption in OCCM deployment	No	Yes	Only helm uninstallation and installation is done.

7.2 Prerequisites

Before performing any fault recovery procedure, ensure that the following prerequisites are met:

- Docker images used during the last installation or upgrade must be retained in the external data source.
- Custom values file used at the time of OCCM deployment is retained. If the `custom_values.yaml` file is not retained, then regenerate it manually. This task increases the overall Fault Recovery time.

7.3 Fault Recovery Scenarios

This section describes the fault recovery procedures for various scenarios.

7.3.1 Scenario 1: Full Site Failure

Single, Multiple, or all Site Failure

This scenario applies when one, more that one, or all sites have failed and there is a requirement to perform fault recovery.

To recover the failed sites:

1. Run the Fault Recovery procedure to install Kubernetes cluster
2. Install OCCM helm charts. For more information about installing OCCM, see the Installing OCCM chapter in the *Oracle Communications Cloud Native Core, Certificate Management Installation, Upgrade, and Fault Recovery Guide.*

7.3.2 Scenario 2: Corruption in OCCM Deployment

This section describes how to recover when the OCCM deployment is corrupted.

Scenario 2a: OCCM deployment is corrupted

To recover the corrupted deployment:

1. Run the following commands to uninstall the corrupted OCCM deployment if needed:

   ```
   helm uninstall <deployment name> --namespace <deployment namespace>
   ```

   For example:

   ```
   helm uninstall occm --namespace occm
   ```

2. Use the backed up copy of the custom-values.yaml file to install the OCCM. For more information about installing OCCM, see the Installing OCCM chapter in the *Oracle Communications Cloud Native Core, Certificate Management Installation, Upgrade, and Fault Recovery Guide.*

Scenario 2b: OCCM configuration data is corrupted

To recover the configuration data:

1. Run the following commands to uninstall the corrupted OCCM deployment if needed:

   ```
   helm uninstall <deployment name> --namespace <deployment namespace>
   ```

   For example:

   ```
   helm uninstall occm --namespace occm
   ```

2. Run the following command to restore OCCM configuration using the backup copy of occm-config-map:

   ```
   kubectl apply occm-config-map_<version>_backup.json
   ```

   For more information, see OCCM Configuration Backup .
3. Use the backed up copy of the custom-values.yaml file to install the OCCM. For more information about installing OCCM, see the Installing OCCM chapter in the *Oracle Communications Cloud Native Core, Certificate Management Installation, Upgrade, and Fault Recovery Guide.*